nature machine intelligence

Article

https://doi.org/10.1038/s42256-023-00691-9

Self-play reinforcement learning guides protein engineering

Received: 30 November 2022

Accepted: 20 June 2023

Published online: 20 July 2023

Check for updates

Yi Wang[1], Hui Tang[1], Lichao Huang[1], Lulu Pan[2], Lixiang Yang[1], Huanming Yang[3,4], Feng Mu[1] ✉ & Meng Yang[1] ✉

Designing protein sequences towards desired properties is a fundamental goal of protein engineering, with applications in drug discovery and enzymatic engineering. Machine learning-guided directed evolution has shown success in expediting the optimization cycle and reducing experimental burden. However, efficient sampling in the vast design space remains a challenge. To address this, we propose EvoPlay, a self-play reinforcement learning framework based on the single-player version of AlphaZero. In this work, we mutate a single-site residue as an action to optimize protein sequences, analogous to playing pieces on a chessboard. A policy-value neural network reciprocally interacts with look-ahead Monte Carlo tree search to guide the optimization agent with breadth and depth. We extensively evaluate EvoPlay on a suite of in silico directed evolution tasks over full-length sequences or combinatorial sites using functional surrogates. EvoPlay also supports AlphaFold2 as a structural surrogate to design peptide binders with high affinities, validated by binding assays. Moreover, we harness EvoPlay to prospectively engineer luciferase, resulting in the discovery of variants with 7.8-fold bioluminescence improvement beyond wild type. In sum, EvoPlay holds great promise for facilitating protein design to tackle unmet academic, industrial and clinical needs.

Protein engineering seeks to discover highly functional sequences by searching the complex high-dimensional surface of the fitness landscape, which characterizes the mapping between protein sequence and the desired property of interest. For a 100-amino-acid target protein, the resulting design space comprises 20^{100} possibilities, which is larger than the number of atoms in the universe. Directed evolution (DE)[1], inspired by natural evolution, employs iterative protocols of random mutation and selection, and scores the candidates using high-throughput functional assays. DE navigates the fitness landscape by acquiring beneficial sequences in a greedy hill-climbing manner, often leading to local optima and discarding enormous mutations. Recently, machine learning surrogate models have been introduced to guide DE (MLDE)[2,3] and accelerate design cycles[4,5]. The model is trained on labelled variants, and predicts protein properties and proposes new candidates for functional characterization, reducing the wet-lab test burden and improving sampling efficiency[6,7]. Cluster learning-assisted directed evolution (CLADE)[8] has improved the hit rate over random-sampling-based focused training MLDE[6] by putting a fitness model on top of hierarchical clusters, and labelled designs from multiple rounds can be iteratively added to refine the surrogate. The input encoding format can vary, ranging from classical one-hot encoding of the 20 amino acids to physicochemical encoding (AAindex[9], Georgiev[10,11]) and neural network-derived embeddings, particularly from large self-supervised masked language models[12–14].

In model-based optimization, a common practice is to use greedy selections to acquire top-ranked predictions. However, this approach may limit the diversity. To address this, AdaLead[15] introduced recombination of mutations with an adaptive greedy search.

[1]MGI, Shenzhen, China. [2]MGI-QingDao, Qingdao, China. [3]Hangzhou Institute of Medicine, Chinese Academy of Sciences, Hangzhou, China. [4]James D. Watson Institute of Genome Sciences, Hangzhou, China. ✉e-mail: yangmeng1@mgi-tech.com; mufeng@mgi-tech.com

eUniRep[16] employed Markov chain Monte Carlo (MCMC) algorithms with simulated annealing to stochastically sample candidates. Another method is proximal exploration, used by PEX[17] to locally search for lower-order mutations around the wild type (WT). Recently, MCMC has also been used for structure-based protein design through hallucinating sequences with valid folds[18] or iteratively querying a language-model-based structure predictor[19] to design unnatural proteins[20] using hierarchical programming[21]. Bayesian optimization (BO) is an alternative to greedy acquisition[22], which estimates model uncertainty and balances exploration and exploitation through upper confidence bound acquisition[23]. Gaussian process (GP)[24,25] regression is a popular surrogate for BO, but its inference cost scales exponentially, which can be intractable for large design space. Utilizing model ensembles to quantify prediction variance can also help in selecting the most reliable models[26]. Regularized generative models can be used to adaptively sample in the latent space, known as latent space optimization. The variational autoencoder-based Dbas[27], and an advanced version, Cbas[28], which penalizes samples in the distribution tail, are some examples. More recently, ReLSO[29] uses a Transformer autoencoder and optimizes sequences by gradient ascent.

Reinforcement learning (RL) enables an intelligent agent to learn how to perform actions by interacting with its environment and maximizing a reward function. DeepMind has successfully combined Monte Carlo tree search (MCTS)[30] with deep RL to master a suite of chess games, evolving from AlphaGo[31] and AlphaGo Zero[32] to AlphaZero[33]. Beyond games, RL has also shown promise in solving combinatorial optimization problems in a range of domains, including chip placement[34], nuclear fusion control[35], flow battery design[36], matrix multiplication acceleration[37] and autonomous driving[38]. In protein engineering, DyNA-PPO[39] was a pioneer in using model-based RL with proximal policy optimization and adaptive surrogate selections. DyNA-PPO formulates the design problem as a Markov decision process, where a blank sequence is initialized and residues are generated from left to right autoregressively. However, the design can only be evaluated at the end of each episode, leading to sparse rewards and increased risk of generating invalid sequences with misfolds. Recently, Baker's laboratory developed an MCTS-based 'top-down' approach that uses fragments to assemble protein nanomaterials for a prespecified design purpose[40], although they have not incorporated neural networks to guide the search.

We propose that the principles of AlphaZero's self-play RL can be adapted to a single-player optimization problem, such as goal-directed protein design. To facilitate DE, we introduce EvoPlay as an in silico mutating agent. In this paper, we demonstrate EvoPlay's effectiveness through four tasks. Task 1 involves in silico benchmarking of protein engineering in the full-length design space, evaluated by a simulated oracle. Task 2 is in silico DE of peptide binders using AlphaFold2 (AF2) as a surrogate to incorporate structure reward at the protein–peptide interface. Task 3 involves MLDE for a four-site combinatorial library of GB1 and PhoQ in an active learning setting. Finally, task 4 involves prospective enzymatic engineering to improve the activity of *Gaussia* luciferase (GLuc)[41], validated by a bioluminescence assay.

Results

Mutating amino acids in a similar way to playing pieces on a chessboard

EvoPlay leverages a neural network-guided MCTS to design proteins, resembling playing on a chessboard (Fig. 1 and Methods). The initial sequence of EvoPlay can be arbitrary, ranging from WT to any other sequences that the user wishes to improve upon. EvoPlay can optimize on a single sequence or a pool of sequences. Additionally, users have the option to specify a full-length input to mimic random mutagenesis, or to specify several positions to narrow down the design space (Fig. 1a).

The optimization process of EvoPlay is a chain of sequential play episodes. The environmental state *s* is represented by a binary matrix of size $20 \times L$, where 20 represents the number of types of amino acid and L is the length of the design space. A play move in this context involves changing the residue type in a single column of the state matrix, that is, a single-site mutation. In each play episode, EvoPlay's RL agent keeps playing move π_t $(0 < t < T)$ of mutations until this episode reaches the end condition (Fig. 1b and Methods). Before playing each actual mutation move π_t, a series of simulations are performed to iteratively execute four steps: selection, expansion, evaluation and backup. In each simulation, EvoPlay traverses the MCTS tree from the root node (initial state, s_t) towards a leaf node where nodes of the tree represent different sequences. Given the current node state s, the selection of the next node depends on the maximum sum value of $Q_{s,a}$ and $U_{s,a}$, both of which are determined by the corresponding edges that lead out of s (Methods). When a leaf node with no child is encountered, it is expanded and evaluated on the basis of $\mathbf{p}$ and v, which are outputs of the policy-value network. When $Q_{s,a}$ updates, the visit count $N_{s,a}$ of the edges on the traversal path is incremented by 1 (equations (2) and (3)). A play move π_t is sampled and is proportional to the exponentiated visit counts at time-step t (Fig. 1b).

EvoPlay implements an iterative look-ahead MCTS procedure (simulations) to generate samples for training a policy-value neural network, which is then used to guide the search. The improved MCTS operator selects a move and queries the surrogate model to retrieve environmental reward r. The neural network is updated by matching the policy vector $\mathbf{p}$ with MCTS execution probabilities $\boldsymbol{\pi}$ and minimizing the error between predicted value v and surrogate score r. In this case, the policy head of the neural network confines the search to high-probability moves, while the value head evaluates mutants during the tree search process. The surrogate can be a machine learning model that maps sequences to functions or structural predictors or other simulators (Fig. 1c). EvoPlay starts from a complete sequence and mutates selected positions to certain residue types as actions, which better mimics natural evolution or laboratory mutagenesis and has stronger interpretability than DyNA-PPO and latent space optimization. The mutating agent can stop at any time during each episode and deliver valid sequences with dense rewards. We will show that EvoPlay is robust to surrogate models. Top designed sequences proposed by EvoPlay can be further validated by wet-lab functional assays or molecular dynamics (MD) simulations (Fig. 1d).

Competitiveness in evolving full-length proteins of high fitness

We first test EvoPlay on two proteins, green fluorescent protein (GFP)[42] from *Aequorea victoria* and poly(A)-binding protein (PAB1)[43]. We consider the full length of GFP and the RNA recognition motif of PAB1 as design spaces, with 20^{237} and 20^{44} possible variants, respectively. Our objective is to optimize GFP for higher fluorescence intensity and PAB1 for higher binding fitness. Following previous works[17,44], we use TAPE[45] as an in silico oracle to simulate the ground-truth landscape and compare EvoPlay with four baseline algorithms: classical BO, regularized generative model Cbas, evolutionary search-based AdaLead and RL-based DyNA-PPO, representing four major paradigms. As naive RL often converges in a single mode with poor diversity, DyNA-PPO adds a diversity-promoting reward to penalize repeating candidates. We also implement Soft Actor-Critic (SAC)[46], an off-policy maximum entropy RL method with stochastic policies to diversify explorations.

We perform five repeats for each starting seed, sampling five different seed sequences from the bottom 30% of observed fitness. To ensure a fair comparison, we restrict the total number of surrogate predictions to 4,000 in one repeat, providing the same surrogate-query budget to each method. We use a convolutional neural network (CNN)[47] architecture (detailed in Supplementary Table 13) for the surrogate after evaluating several other architectures (recurrent neural network-based UniRep[48], MuFacNet[17], GP[25] and hybrid[49] in

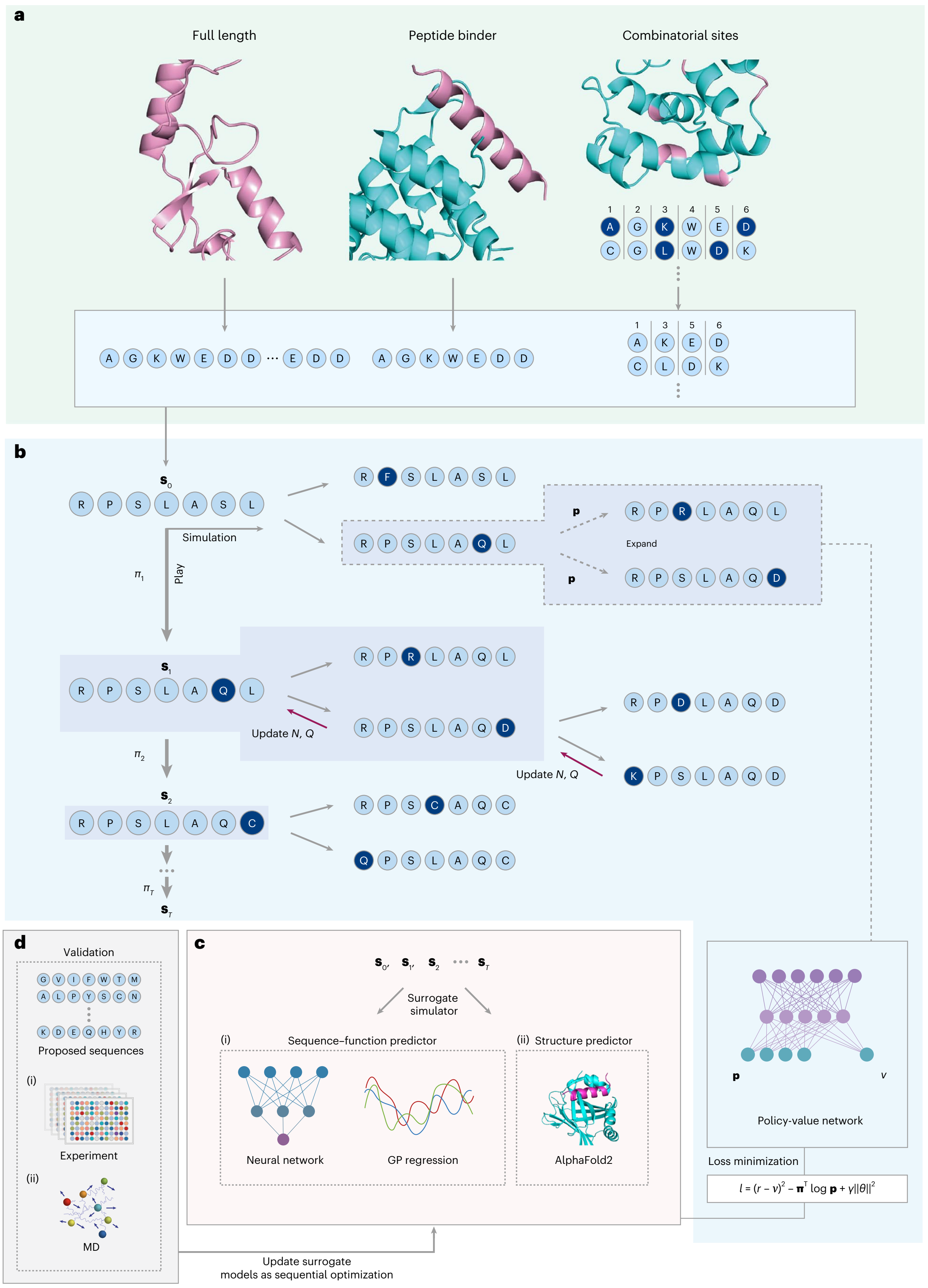


Fig. 1 | Overview of EvoPlay. a, EvoPlay can accept different types of input, such as full-length proteins, peptide binders and combinatorial sites. **b**, The play moves and MCTS simulations generate training samples for the policy-value network, which guides the MCTS search. **c**, Surrogate models can be functional predictors with different architectures, such as CNNs or GP regressors, or structure-based predictors, such as AF2. **d**, The generated sequences can be validated using wet-lab measurements or MD simulations (Methods).

Supplementary Table 12) on observed GFP datasets. The network architectures of the policy-value network are given in Supplementary Table 11. We use the open-source FLEXS environment[15] to implement benchmarking methods and gradually increase the training samples for the surrogate by 10% in each round to prevent overfitting. In every repeat, we conduct ten rounds, each calling the surrogate 400 times, and record all proposed sequences to track the cumulative maximum fitness scored by the TAPE oracle. As depicted in Fig. 2a and Supplementary Figs. 1 and 2, EvoPlay outperforms other algorithms and produces higher maximal fitness sequences within prespecified optimization steps for both datasets (listed in Supplementary Tables 1 and 2). EvoPlay also demonstrates the best sampling efficiency, evaluated by mean fitness in most starting cases for PAB1. As for GFP, AdaLead is the only comparable method, while others are inferior. Diversity and novelty metrics are calculated on the basis of the top 100 and all generated designs by RL-based methods, as presented in Supplementary Tables 3 and 4. Across all five seeds, EvoPlay explores more novel and functional space than other RL methods. AdaLead's exploration diversity, driven by recombination, does not contribute proportionally to fitness improvement, as shown in Fig. 2b(ii),(iv). We count the number of sequences exceeding functional thresholds (3.5 for GFP and 0.5 for PAB1) for all methods (Fig. 2b(i),(iii) and Supplementary Tables 16 and 17). The results demonstrate that EvoPlay achieves a well balanced trade-off between exploration and exploitation, generating more high-quality sequences.

For PAB1, we perform MD simulations on randomly selected mutants from the top 1% of designs (see the mutants in Supplementary Table 19) together with the WT, to validate our results. After 200 ns of MD simulation, the structure of EvoPlay designs superimposes well with WT (Fig. 2d), and quickly enters an equilibrium state (Fig. 2d(iii)). Overall, the root-mean-square deviation (RMSD) fluctuations of the complexes formed by EvoPlay designs are smaller than those of the WT and AdaLead designs (Fig. 2d(iv)), indicating that EvoPlay generates more stable complexes. The binding free energies of EvoPlay designs are lower than those of WT and AdaLead designs, supporting the stronger binding affinity of EvoPlay designs with RNA (a detailed energy decomposition is presented in Supplementary Table 5). Taken together, these MD simulations (see Supplementary Fig. 13 for GFP) further justify EvoPlay's superior performance and its ability to capture some intrinsic rationalities regarding the structure–function mapping.

Recently, language models trained on millions of protein sequences (ESM-1b[13], ESM-1v[50]) have enabled zero-shot prediction of mutational effects on function. In this study, we hypothesize that EvoPlay designs towards extrinsic fitness could be correlated with the intrinsic patterns learned by these models. To test this, we select one ESM-1b model and five ESM-1v models—six in total—to calculate the continuous information entropy of top EvoPlay designs and natural ones. Our results, shown in Fig. 2c, reveal that EvoPlay converges to natural positional variability with high correlation for both the GFP and PAB1 datasets (Spearman's $r_s = 0.967$, $P < 1.31 \times 10^{-14}$, for GFP and Spearman's $r_s = 0.896$, $P < 2.41 \times 10^{-27}$, for PAB1). To further improve EvoPlay for fitness enhancement, we explore whether using prior natural evolutionary information can help constrain its action space. As an example, we narrow down the full-length space of GFP from 237 to a 35-site subspace by selecting amino acids on the basis of corresponding entropy obtained from the same evolutionary-scale modelling (ESM) models. We find that this approach (EvoPlay-ESM) produces a higher cumulative fitness value more robustly, with a lower variance in five runs (Fig. 2a(iii)). We also compare these 35 positions selected by the language models with the 30 sites having the highest information entropy indicated by the position-specific scoring matrix and find 12 overlapping positions. In addition, we find 24 overlapping positions with the disordered regions predicted by DisEMBL[51] (see Supplementary Table 18 for more details).

Designing peptide binders using AF2 as a surrogate

AF2 has revolutionized reliable structure prediction of protein monomers[52] and multimers[53] as well as peptide–protein complexes[54]. AF2 has also been utilized for fixed-backbone sequence design as an inverse problem[55]. Wicky et al.[56] leveraged Monte Carlo search coupled with AF2 prediction to design cyclic homo-oligomers from a random sequence. In this study, we focus on designing peptide binders using AF2 with the concatenation of the target receptor MSA (multiple sequence alignment) and the peptide sequence as input. Although the monomeric version of AF2 was trained on single chains, it has been proved to achieve high-resolution structure prediction for 12 out of 96 peptide–protein complexes resulting from a search against the Protein Data Bank (PDB) using peptide–protein binding constraints[54]. This approach outperforms traditional docking protocols and achieves performance approaching that of crystallography experiments. ProteinMPNN[57,58] has successfully utilized the backbone structure to generate binders with a success rate comparable to that of Rosetta. EvoBind[59] introduced a per-residue local distance difference test (pLDDT)-correlated loss term to optimize receptor–peptide interfaces. However, EvoBind's random mutation sampling with greedy search is susceptible to becoming trapped in local optima. Additionally, by using AF2 with eight recycles, EvoBind consumes an average of 12 h 46 min for one peptide on NVIDIA A100 Tensor Core graphics processing units with 40 GB of random-access memory, which is not computationally efficient.

We aim to investigate whether EvoPlay's look-ahead search can efficiently generate high-quality binders. We select four representative peptides, including 1ssc and 2cnz, for which AF2 gave the best interface structure predictions (with the lowest interface RMSD < 0.5 Å), as well as two additional targets (3r7g and 6seo in Supplementary Fig. 7). In the experiment, EvoPlay initializes five random seeds using a Gumbel distribution of all 20 amino acids and evaluates 1,000 design rounds, repeated five times for each starting seed. During optimization, EvoPlay uses the reciprocal of EvoBind's structural loss terms as the reward to score each explored state (that is, a peptide) and a replay buffer to store accumulated states, moves and rewards, which are then used to train the neural network to continuously guide MCTS with policy improvement. To improve generation efficiency, sequences from both simulation moves and play moves are taken into account. We also modify AF2's Evoformer module (v2.0) to generate reliable predictions even at recycle 0[60] (Supplementary Notes and Supplementary Fig. 8) and add a 'jump out' option to reduce EvoPlay's sensitivity to the initial state, helping it evade local optima by adding randomness. Specifically, this option randomly mutates one to three sites on the current best sequence as the starting state for the next iteration when no reward improvement is detected for six consecutive episodes. In addition to EvoBind, we include MCMC with simulated annealing[18] as a benchmarking algorithm. To illustrate, we performed five repetitions from five random seeds for 1ssc, each consisting of 1,000 iterations, resulting in a total of 25,000 sequences across 25 groups of runs (see Fig. 3a for starting sequence 4 and Supplementary Fig. 9 for others). The results indicate that EvoPlay achieves a significantly higher hit rate (Kruskal–Wallis test, $P = 0.046$) at a loss threshold of 0.01 when compared with EvoBind and MCMC (Fig. 3b). Even at more stringent thresholds of 0.005 and 0.001, EvoPlay shows more hit counts than EvoBind and MCMC (Fig. 3c; the results for 2cnz are available in Supplementary Figs. 10–12). Distance-preserving scaling plots suggest that EvoPlay can explore with greater novelty and diversity than greedy EvoBind (Fig. 3d; results for other starting seeds can be found in Supplementary Fig. 5 and Supplementary Table 6).

We analyse the top 1% of designs extracted from all runs and visualize contact sequence logos[61]. The frequency distribution of amino acids in the designed peptides converged to a pattern similar to that of the native structure (Fig. 3h). We conduct MD simulations (200 ns) for the peptide–protein complexes designed by EvoPlay and EvoBind, as well as the native complex. The average RMSD values were 0.260 ± 0.037 nm,

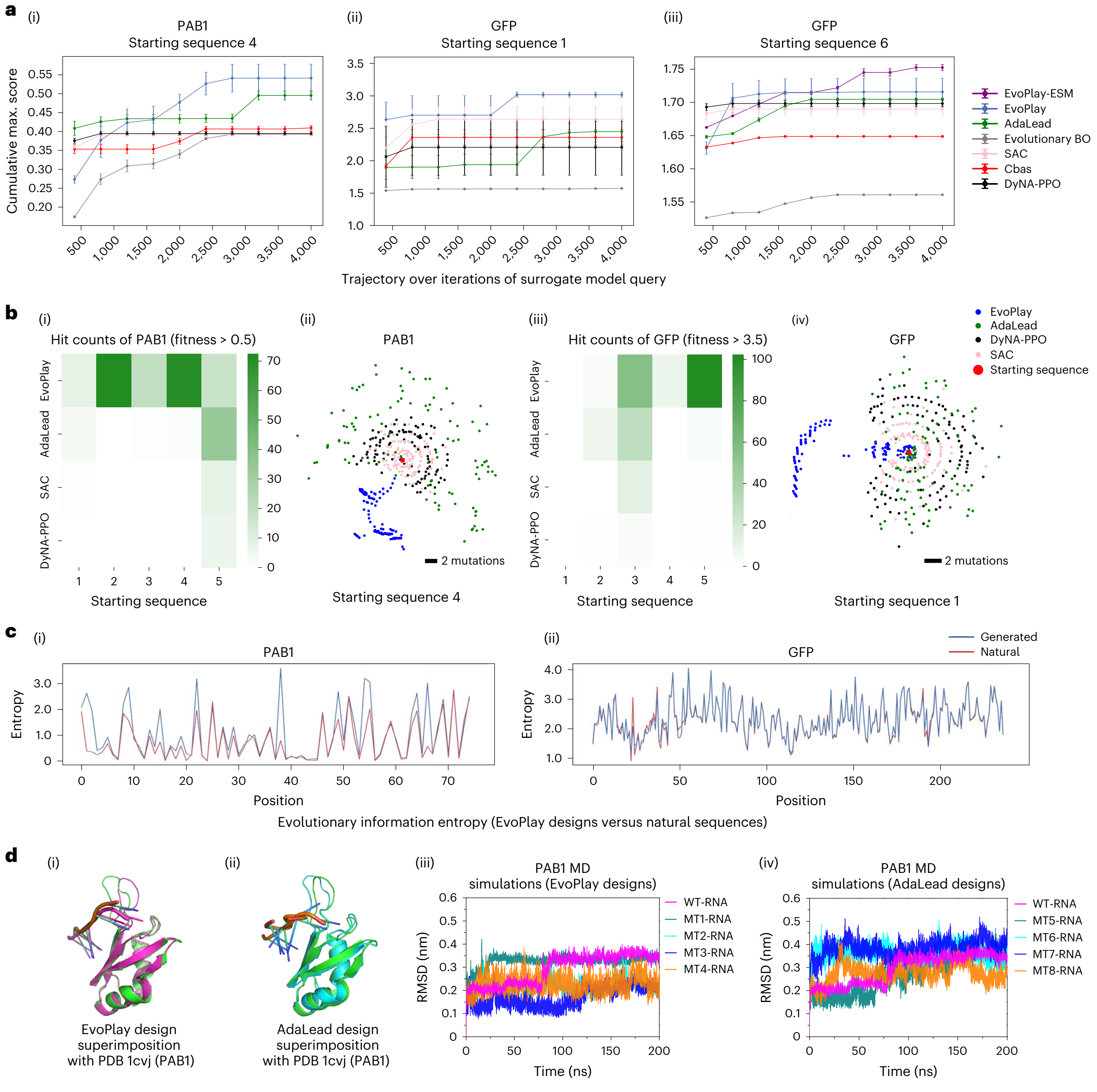


Fig. 2 | PAB1 and GFP design evaluated by TAPE oracle. a, Cumulative maximum results of five repeats using a single starting sequence on PAB1 and GFP datasets, comparing EvoPlay against other methods, including AdaLead, evolutionary BO, Cbas, SAC and DyNA-PPO. The solid line represents the average of five repeats, and the error bar shows the s.d. (Supplementary Tables 1 and 2). EvoPlay-ESM, an evolutionary confined version of EvoPlay, is compared against six other models evaluated on the GFP dataset. All scores are obtained using the TAPE oracle. **b**, Heatmaps showing the average count of sequences generated by EvoPlay and other benchmark methods, with fitness scores exceeding the threshold (3.5 on the GFP dataset and 0.5 on the PAB1 dataset), in five replicates of each starting sequence, and distance-preserving scaling plots showing the top 100 sequences generated by EvoPlay, AdaLead, DyNA-PPO and SAC (see Supplementary Figs. 3 and 4 for the other four starting sequences). **c**, Information entropy (bits) of EvoPlay-designed sequences when compared with natural sequences (Methods). **d**, Native PAB1 (green) superimposed with EvoPlay-designed mutant (magenta) and AdaLead-designed mutant (cyan) after 200 ns MD simulation and the RMSD trends of PAB1 WT (WT-RNA) and EvoPlay-designed mutants (MT1-RNA, MT2-RNA, MT3-RNA and MT4-RNA) and AdaLead-designed mutants (MT5-RNA, MT6-RNA, MT7-RNA and MT8-RNA) during 200 ns MD simulations. The average RMSD values of the WT-RNA, MT1-RNA, MT2-RNA, MT3-RNA, MT4-RNA, MT5-RNA, MT6-RNA, MT7-RNA and MT8-RNA complexes are 0.30 ± 0.06 nm, 0.33 ± 0.05 nm, 0.21 ± 0.03 nm, 0.17 ± 0.04 nm, 0.23 ± 0.04 nm, 0.27 ± 0.08 nm, 0.36 ± 0.03 nm, 0.38 ± 0.003 nm and 0.27 ± 0.04 nm, respectively.

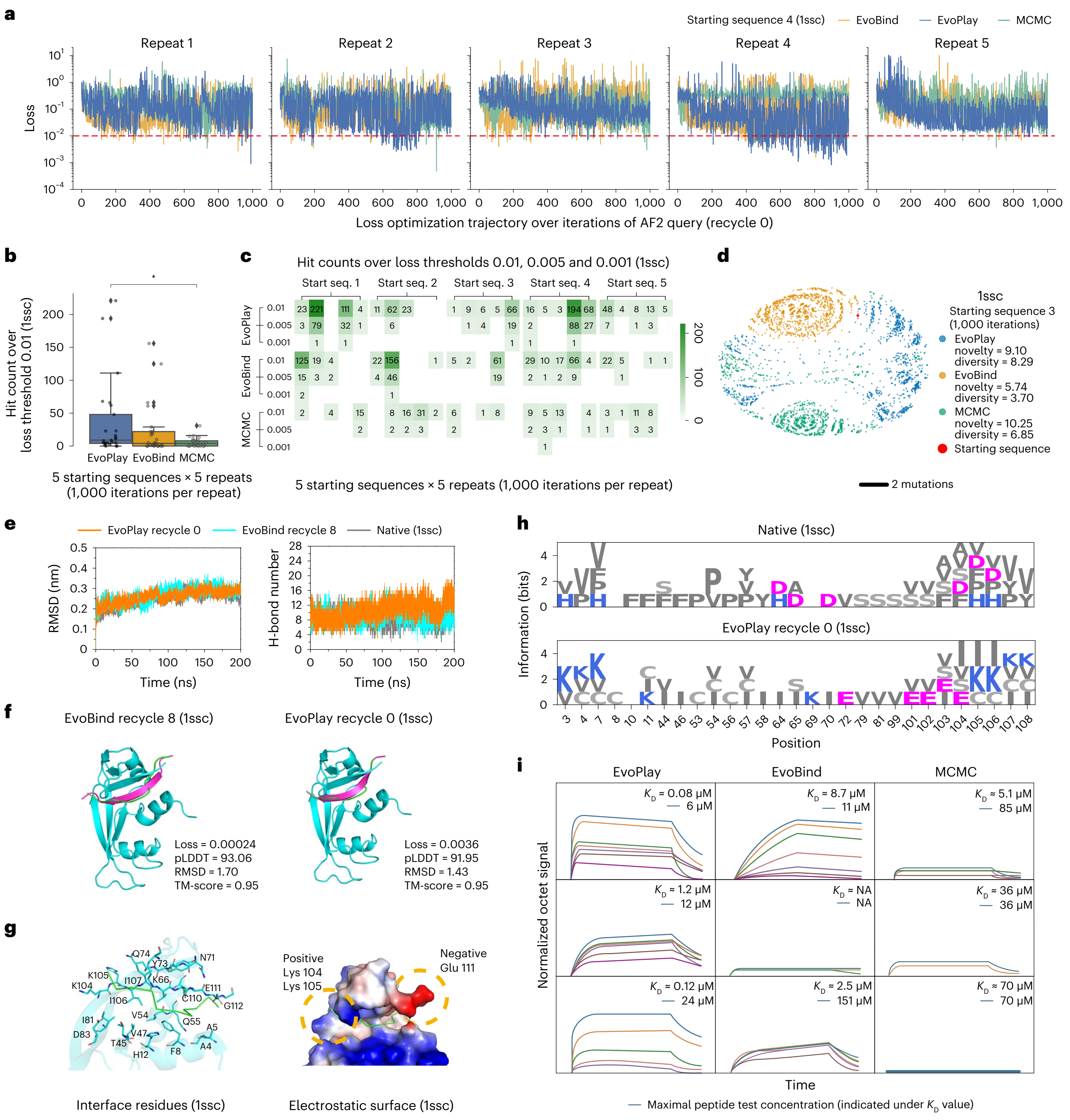


Fig. 3 | High-quality peptide binder designed by EvoPlay. a, Performance comparison of EvoPlay, EvoBind and MCMC on 1ssc (PDB) with the same starting sequence. The *x* axis represents iterations; the *y* axis represents the optimized loss. The dashed line represents the threshold 0.01. **b**, Box plot displaying the hit rates among benchmarking algorithms for 1ssc ($n = 25$ independent repeated experiments for each algorithm). The box plots show the median, the first quartile and the third quartile. The whiskers extend to points that lie within 1.5 times the interquartile ranges of the lower and upper quartiles, and then observations that fall outside this range are displayed independently. A two-sided Kruskal–Wallis test is used for statistics analysis, $P = 0.046$, $*P < 0.05$. **c**, Heatmap displaying hit counts for reaching three thresholds (0.01, 0.005 and 0.001). **d**, Distance-preserving scaling plot visualizing the novelty and diversity of generated sequences. **e**, RMSD changes and hydrogen bonds formed during 200 ns MD simulations of the peptide–protein complex. **f**, Superposition comparison of the designed peptide–protein complex (receptor in cyan, native peptide in green and designed peptide in magenta). TM-score, template modelling score. **g**, Left: interface residues (<8 Å) of the 1ssc protein (cyan) and peptide (green) in stick mode. Right: electrostatic surface of the native 1ssc protein. **h**, Contact sequence logos for the native (upper) and top 1% of EvoPlay-designed (lower) peptides. **i**, Binding affinity of designed peptides to RNase1 targets characterized by biolayer interferometry. The octet signal is normalized by the maximum signal of the cognate binder–target pair. 'K_D ≈ NA' indicates undetectable binding affinity.

https://doi.org/10.1038/s42256-023-00691-9

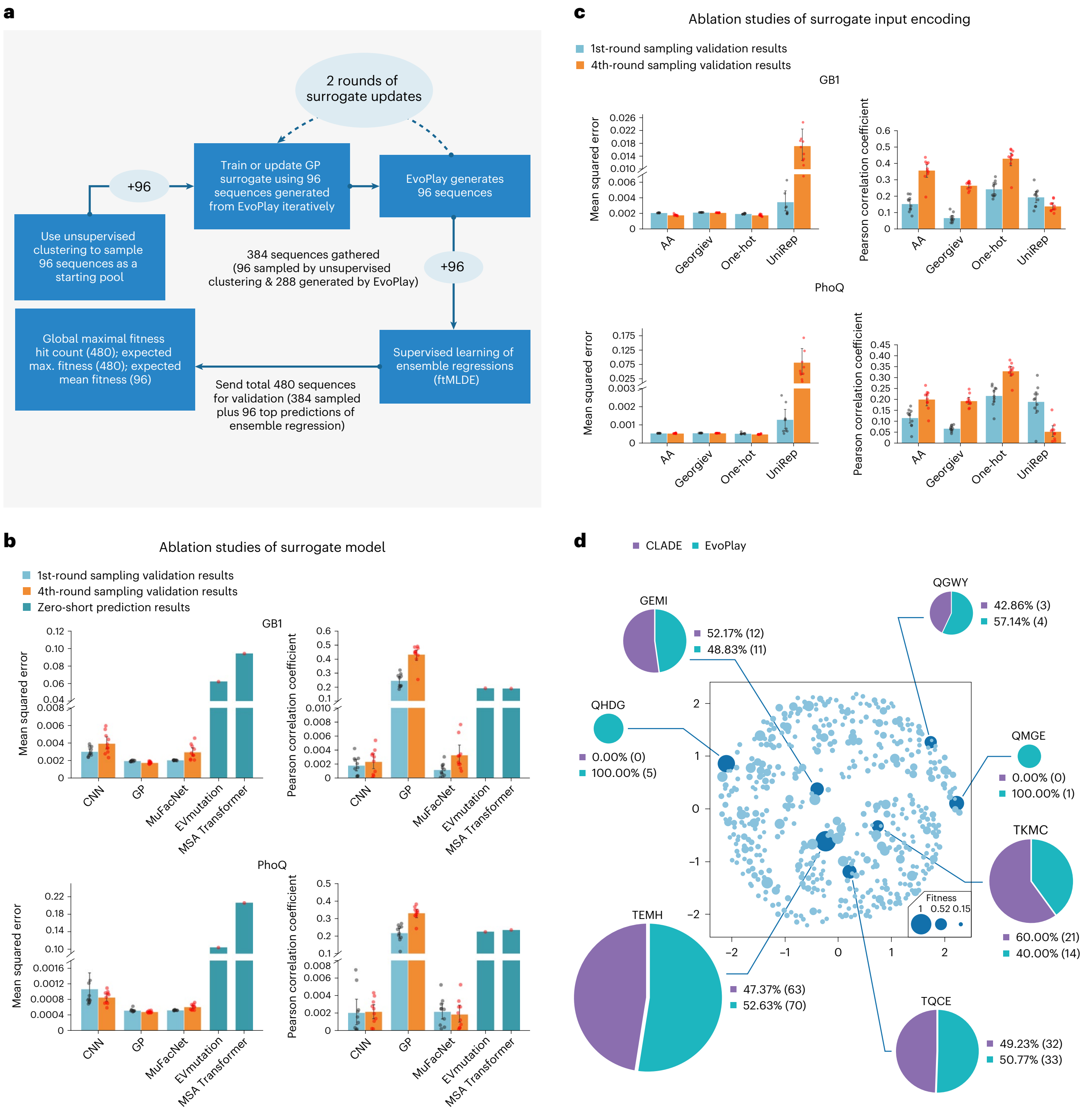


Fig. 4 | EvoPlay-assisted MLDE. a, Illustration of two-stage mutant sampling to evaluate EvoPlay-assisted MLDE. The first stage involves sampling 96 mutants using *K*-means clustering and random in-cluster sampling, followed by sampling 288 mutants using EvoPlay-agent, and prioritizing another 96 mutants using focused training MLDE (ftMLDE) ensemble models in the second stage. **b**, Ablation studies of different surrogate models (CNN, GP, MuFacNet, EVmutation and MSA Transformer). Each bar plot represents the values of mean squared error and Pearson correlation coefficients on GB1 (149,361 sequences excluding 96 or 384 training samples) and PhoQ (160,000 sequences excluding 96 or 384 training samples) test sets and the error bars show the 95% confidence interval. **c**, Ablation studies of different encoding schemes (AAindex, Georgiev, one-hot encoding, UniRep embeddings) for the GP regression model. Each bar plot represents corresponding values as in **b** and the error bars show the 95% confidence interval. In **b** and **c** there are ten independent repeats for each bar. Black dots represent results of each repeat after first-round sampling (96 samples), and red dots represent results after four rounds of sampling (384 samples). **d**, Comparison of top hit counts using one-hot encodings of the 500 highest-fitness sequences from the PhoQ dataset, transformed to a two-dimensional array through DMS dimensionality reduction. Circle size represents experimental fitness value, and the pie chart depicts hit counts of the top seven highest-fitness mutants discovered by EvoPlay and CLADE. Mean squared error and Pearson correlation coefficient are calculated using the Python libraries sklearn and NumPy, respectively.

0.267 ± 0.038 nm and 0.240 ± 0.037 nm, respectively, indicating consistent behaviour (Fig. 3e left). We also count the number of hydrogen bonds formed in the complexes designed by EvoPlay (9.96 ± 1.61), EvoBind (9.25 ± 1.91) and the native complex (7.97 ± 1.61) (Fig. 3e right). Additionally, we estimate the binding free energy (Supplementary Table 9) using the MM/PBSA approach[62]. When compared with the native complex (−74.86 ± 5.79 kcal mol^{-1}) and EvoBind designs (−87.43 ± 5.42 kcal mol^{-1}), EvoPlay (−96.13 ± 5.45 kcal mol^{-1}) achieves 28.41% and 9.95% improvements in stabilization, respectively, which rationalizes its superiority. EvoPlay with recycle 0 (of AF2) even shows comparable performance to EvoBind with recycle 8 (Fig. 3f). The structure of interfacial residues and electrostatic interactions of the EvoPlay design are shown in Fig. 3g. Results for 3r7g and 6seo (recycle 3) are available in Supplementary Fig. 7 and Supplementary Tables 8 and 9.

We further validate EvoPlay's design performance with wet-lab experiments. For each starting sequence used for three benchmarking algorithms, the peptides with the highest pLDDT value are chosen among the top five designed sequences ranked by loss (Supplementary Table 15). Plasmid construction, expression and purification of receptor protein as well as peptide synthesis are described in Supplementary Notes. Subsequently, biolayer interferometry (Methods) is performed to measure the binding interactions between peptide candidates and the RNase1 (PDB 1ssc) protein. The results show that all five peptides designed by EvoPlay demonstrate outstanding binding affinity targeting the RNase1 protein (Fig. 3i and Supplementary Fig. 15). Three out of the five peptides exceed WT, and one of them achieves the highest affinity (K_D = 80 nM), in the nanomolar range. In contrast, MCMC and EvoBind designs show lower binding affinity, with only one of MCMC's designs exceeding the WT, and the overall affinity performance only at the micromolar level. Therefore, the wet-lab validation results provide strong evidence supporting the superior performance of EvoPlay over EvoBind and MCMC.

Designing mutants across four-site combinatorial libraries

The goal of this task is to improve the efficiency of MLDE. The original MLDE approach trains ensemble regressors using random sampling in sequential rounds and acquires samples greedily. However, exploring low-fitness regions may not provide useful landscape knowledge. To address this, CLADE[8] introduces a two-stage protocol (Supplementary Notes). In the first stage, an unsupervised hierarchical clustering guides the exploration of the fitness landscape to iteratively select 384 informative training variants. In the second stage, supervised ensembled regression models[6] are trained to evaluate the entire library and prioritize the top 96 predicted variants. The main challenge is to reduce the inclusion of minimally informative holes in the training set to improve the surrogate. CLADE uses GP-type sampling with upper-confidence-bound acquisition, which has been shown to be the most effective sampling method. Additionally, the model is iteratively updated by adding labelled samples in an active learning manner.

In this section, we evaluate EvoPlay on two four-site combinatorial libraries, GB1 and PhoQ, using the same setting as CLADE (depicted in Fig. 4a). EvoPlay generates a starting sequence pool of 96 variants using *K*-means clustering and then selects one sample from the pool to initialize the optimization. In the generation process, EvoPlay follows the same method as in other tasks, with the only difference being that if initial sequences do not change during ten episodes the agent will select a new initial sequence from the pool (see Methods for more details). Then, the agent proposes 96 variants for three rounds (a total of 288 variants) iteratively and the GP regressor scores each move and rewards the agent. During optimization, we progressively add labelled 96 mutants from the previous round to update the GP regressor of the current round. We evaluate various surrogates (including CNN[47], MuFacNet[17] and zero-shot predictors EVmutation[63] and MSA Transformer[14]) with different input formats (AAindex[9], Georgiev[11] and UniRep[48] embedding) using mean squared error and Pearson

Table 1 | Evaluation of EvoPlay and CLADE over DE tasks (GB1, PhoQ)

		Predicted max. fitness	Predicted mean fitness	Global maximal fitness hit count (max.: 500)
GB1	CLADE	7.24 (max.: 8.761)	2.79	83
	EvoPlay	7.38 (max.: 8.761)	3.21	88
PhoQ	CLADE	64.41 (max.: 133)	9.77	63
	EvoPlay	64.90 (max.: 133)	11.27	70

All statistics for both methods are obtained from 500 independent repeats. The predicted maximum fitness and global maximal fitness hit count are evaluated on the union of the first-stage 384 sampled mutants and the second-stage top 96 predictions by the ensemble regression model. The predicted mean fitness is evaluated only on the top 96 predictions by the ensemble regression model. One-hot encoding is used for both methods.

coefficient as metrics, and find that GP with one-hot encoding performs the best for our small-scale training samples selected from a large unseen landscape. The ablation results (ten independent validations for GB1 and PhoQ datasets respectively) support our choice of GP as surrogate and one-hot encoding as the input encoding method (Fig. 4b,c).

We conduct 500 independent runs for both EvoPlay and CLADE to compare their performances. In each run, 384 variants are sampled from the combinatorial space (which has a size of around 160,000) in the first stage, and another 96 mutants are selected in the second stage. We evaluate the global maximal fitness hit count and predict maximal fitness across all 480 variants. In addition, we assess the predicted mean fitness of the top 96 variants predicted by the second-stage machine learning model ensembles (results summarized in Table 1). For both GB1 and PhoQ datasets, EvoPlay achieves slightly better maximal fitness on the union set while notably higher mean fitness of top 96 machine learning-predicted variants in the second stage, with around 15% relative improvement. Furthermore, EvoPlay outperforms CLADE in hitting global maximal fitness counts by 5 and 7 for GB1 and PhoQ, respectively. For the PhoQ dataset, we visualize the 500 highest-fitness variants from the library (Fig. 4d). EvoPlay can hit all top seven variants, while CLADE fails to discover two local peaks, that is, QHDG and QMGE, which rank second and third among all mutants. For the highest-fitness variant (TEMH), EvoPlay hits it more frequently than does CLADE (70 versus 63). In addition, it is worth noting that CLADE's approach of inferring over the entire design space can be computationally challenging for datasets such as GFP and PAB1 with much larger design space, and even more intractable for sophisticated surrogates such as AF2. On the other hand, EvoPlay eliminates the need for enumeration and exhaustive ranking, making it more manageable for handling more complex surrogates, as demonstrated in the peptide-binder section. By accepting prespecified sites as input, EvoPlay provides protein engineers with the flexibility to customize their design windows, acting as a more practical and versatile tool for MLDE tasks.

Prospective engineering of improved luciferase

To evaluate EvoPlay prospectively, we attempt to optimize the bioluminescence of GLuc[41] on the basis of our in-house pool (Fig. 5a). GLuc (weight-averaged molecular mass, M_w = 18.2 kDa, excluding secretion tag) is isolated from a marine copepod, *Gaussia princeps*, and is used to catalyse the bioluminescence reaction, emitting bright blue light by oxidizing coelenterazine (CTZ) without any other co-factors except oxygen. Despite its small molecular mass, the luminescence intensity driven by GLuc is stronger than that of most other luciferases, making it an attractive tool for evaluating biological events[64,65].

Here, we provide a brief description of how we construct our in-house variant pool. First, we utilize MSA search (DeepMSA v1

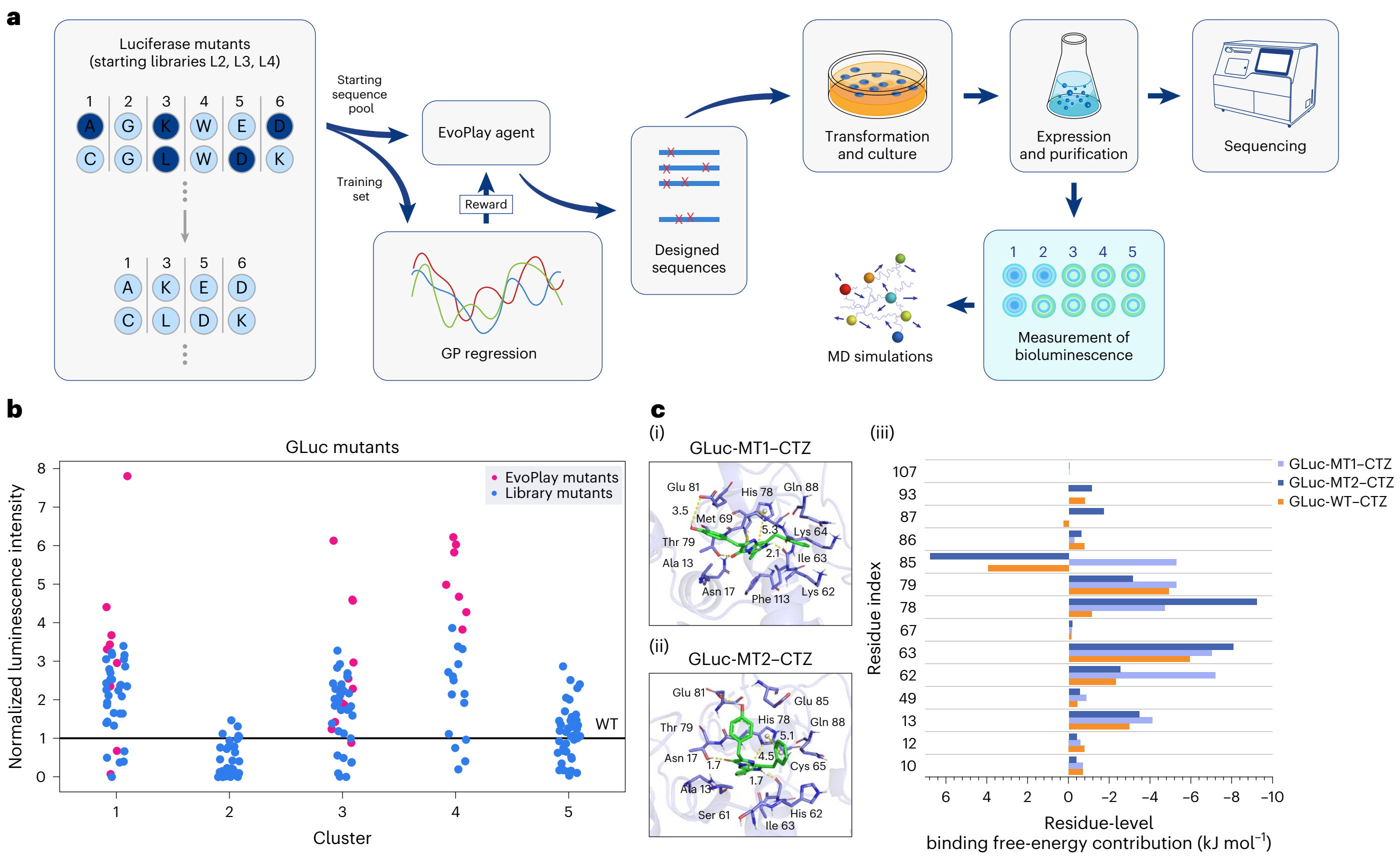


Fig. 5 | Prospective GLuc engineering. a, Workflow of GLuc engineering. EvoPlay designs mutants based on a starting sequence pool and validates the proposed designs by bioluminescence measurement and MD simulations. **b**, Luminescence intensities of 29 EvoPlay designs and 164 starting mutants. Among the 29 EvoPlay designs, 11 exceed the maximum value of the starting library and 4 achieve a sixfold improvement beyond the WT level. The maximum luminescence intensity of the EvoPlay designs reaches eight times that of WT, which is twice the maximum value in the starting library. **c(i)**,**(ii)**, MD simulations show similar binding patterns of GLuc-MT1 (V12A, H62K, P67L, E85S, S86T, A87G, E93P, L107M, V121E) and GLuc-MT2 (N10S, V12A) with CTZ. In detail, CTZ can form hydrogen-bond interactions with the Ile 63, Glu 81 and Thr 79 residues of GLuc-MT1 and GLuc-MT2, and the benzene ring of CTZ can form strong π–π conjugate interaction with the imidazolyl of His 78. **(iii)**, Comparison of residue-level energy contributions of GLuc-WT, GLuc-MT1 and GLuc-MT2. The Glu 85 of GLuc-WT and GLuc-MT2 are positively charged, exerting a certain repulsive effect on CTZ molecules and unfavourable for the GLuc–CTZ binding. Ser 85 contributes notably to binding CTZ. His 62, Ile 63, His 78, Thr 79 and Ala 87 residues contribute greatly to CTZ binding affinity. Asn 10 and Val 12 are mutated into Ser 10 and Ala 12, increasing the cavity adaptability.

pipeline[66]) to identify non-conserved regions (MSA sequences are presented in Supplementary Data 1, and the resulting logo plot is shown in Extended Data Fig. 1). Next, we create three random mutant libraries—L2, L3 and L4—that encompass residues in these regions. Specifically, L2 includes the following residues: Phe 9, Asn 10, Val 12, Ala 13, Ser 16, Ala 19, Thr 20 and Leu 23. L3 contains the following residues: Lys 49, His 62, Pro 67, Glu 85, Glu 93, Asp 106, Leu 107 and Val 121. L4 comprises the residues Gln 135, Gln 146, Thr 150, Ser 153, Gly 157, Gln 158, Lys 161, Ala 165 and Gly 166. In addition to these regions, we introduce other empirical sites from the GLuc cavity (Ala 13, Val 14, Asn 17, His 78, Thr 79 and Ile 114), the entrance loop (Ser 86, Ala 87, Gln 88, Gly 89, Gly 92 and Ile 95) and PCR errors (Ala 15, Pro 36, Lys 71, Glu 81 and Glu 111) to build mutant combinations. Finally, we construct 200 combinational mutants into plasmids, transform them into bacteria, induce protein expression and measure luminescence (Methods). On the basis of these experimental results, we further remove mutants in the loosely packed moiety region (sites 150–168)[67] and obtain a pool of 164 variants (Supplementary Data 2). This variant pool serves as both the seed candidates for EvoPlay and the training dataset for the surrogate GP regressor.

This is a combinatorial-sites design, in which we first calculate the mutation frequencies of each site from the 164-variant pool and identify 32 positions that mutate at least once (GLuc input data preprocessing for surrogate model). These 32 sites are then combined to form a new input space, resulting in a 20 × 32 matrix that represents the state environment. Starting sequences are reselected from the seed pool when no reward improvement is detected for ten episodes. In each independent run, we collect 150 sequences as outputs from EvoPlay's play moves, resulting in a total of 1,047 candidates after deduplication across ten replicates. We further implement *K*-means clustering of all candidates and select top mutants within each cluster ranked by surrogate score. Ultimately, we choose 36 variants for wet-lab validation (Supplementary Fig. 14). We include one positive control (B6 from the 164-variant pool) and a WT control. Plasmids carrying mutants are transformed into *Escherichia coli* strain Origami B (DE3) for protein expression (Methods). Subsequently, we successfully construct and purify 29 mutants (Supplementary Data 3) using the His-tag approach. Bioluminescence signals are measured immediately after adding CTZ (Methods).

We normalize the signal intensity of 164 in-house variants and 29 EvoPlay designs by using WT fitness as reference (which is equal to 1, Fig. 5b). Among the 164 in-house variants, the maximum fitness is 3.86 times that of WT, and the B6 positive control is 3.05 times that of WT. The results reveal that 26 out of 36 variants (including seven variants that are not successfully purified) emit stronger bioluminescence than WT; among them, 4 variants achieve a sixfold improvement over WT, and 11 designs outperform all in-house variants, with the maximal design achieving a 7.8-fold improvement over WT (twofold improvement over the best in-house variant). Moreover, MD simulations show that two proposed mutants, GLuc-MT1 and GLuc-MT2, form hydrogen bonds, π–π conjugates and hydrophobic interactions with the CTZ compound to improve binding affinity and stability. In particular, Glu 85 of WT and GLuc-MT2 contribute positive energy for binding (3.93 and 6.76 kJ mol^{-1}, respectively; Fig. 5c(iii)) while the energy contribution of Ser 85 in GLuc-MT1 becomes negative (−0.223 kJ mol^{-1}), resulting in a notable increase in GLuc-MT1's binding affinity with CTZ (−122.76 ± −23.41 kJ mol^{-1}). The His 62 to Lys 62 mutation shows an improved energy contribution to substrate binding (Fig. 5c(iii)). Additionally, the combinatorial mutations (E85S, S86T, A87G, E93P, L107M) promote hydrophobic interactions with CTZ in the loop region (85–107) (Fig. 5c(i)). More discussions of the MD analysis result for GLuc-MT2 are provided in Supplementary Notes (Fig. 5c(ii)). In summary, EvoPlay is effective in identifying substantially higher-fitness mutants beyond the starting library and can be integrated into existing engineering workflows.

Discussion

In this study, we comprehensively demonstrate the merits of self-play RL to facilitate protein design towards functional or structural properties. Neural network-guided MCTS has been proven powerful for exploring complex landscapes, outperforming a host of algorithms including in silico optimization of full-length proteins, peptide-binder design using AF2-derived structural rewards, and assisting DE sampling over empirical combinatorial libraries with sequential surrogate refinements. Additionally, EvoPlay can substantially improve luciferase activity on our in-house pool. We highlight several key advantages of EvoPlay. (1) EvoPlay explicitly defines an interpretable action space that scores each visited state and avoids receiving sparse rewards only from end-states as in other RL models. It can accept full-length or prespecified sites. (2) EvoPlay can either use only play states as output or incorporate simulation states as additional output sequences. EvoPlay requires only a limited number of surrogate evaluations, while BO's acquisition process requires exhaustive scoring and prioritizing in the whole design space, which becomes computationally intractable for more complex libraries or full-length design. Furthermore, when the surrogate is a structural predictor or has a more sophisticated architecture, look-ahead MCTS shows greater advantages in sampling efficiency and substantially reducing the query budget. (3) EvoPlay effectively addresses the exploration–exploitation trade-off, generating candidate sequences with higher fitness. (4) EvoPlay shows robustness to surrogate models, preferring computational speed over nominally more accurate predictive models. Even with less accurate surrogates trained on limited datasets, EvoPlay extrapolates well and proposes high-quality variants.

EvoPlay can also benefit from the use of zero-shot predictors to exclude possible zero- or low-fitness mutants. Self-supervised learning approaches such as MutCompute2[68], ProteinBiRNN[69] and ESM-IF[70] can be introduced to provide evolutionary or structural constraints and thus reduce the search space. In drug discovery or enzyme development, where optimizing multiple aspects is necessary, improving one specified function may harm others. EvoPlay can synthesize a combination of properties into a single reward and search for a Pareto-optimal set[36,71]. Though in silico DE cannot replace experimental validation, it can substantially expedite the screening of large-scale libraries driven by high-throughput microfluids[72,73] or automated robotics[74,75]. EvoPlay's efficient sequence generation module can fit well into a stepwise optimization workflow to facilitate the design cycle. EvoPlay is also scalable through employing multiple workers in parallel to process batched designs starting from independent seeds. As the cost of automation and microfluid-based profiling decreases, it becomes increasingly important to balance resource consumption between dry-lab and wet-lab experiments. EvoPlay's sampling efficiency under a controlled budget well supports the synergy.

Several improvements can be made to EvoPlay. (1) Structural constraints can be incorporated into MCTS moves. Inspired by ProteinMPNN's sequence–structure cogeneration[57,76], residue probability can be evaluated on the fly on the basis of backbone coordinates and orientations[58], which injects structural information into EvoPlay. (2) EvoPlay can be customized to scaffold functional sites. Energy interaction terms and motif RMSD constraints, as in RFDesign[77], can be integrated to hallucinate scaffolds. (3) The enzyme conformational landscape[78] can be incorporated into the surrogate to aid biocatalytic design of targeted conformation, leading to tailored activity or selectivity. (4) EvoPlay can extend the mutating action from the residue level to the fragment level. Helices or loops can be substituted under geometric constraints. To sum up, EvoPlay highlights the great promise of deploying self-play RL for protein engineering.

Methods

EvoPlay workflow

The environmental state of EvoPlay is represented by a binary matrix of size 20 × L with columns indicating positions (L) and rows indicating amino acids (20) and the action space is correspondingly a flattened vector of size 20 × L. When a move is performed, one element from the action space vector is selected, resulting in a single-site mutation on the state sequence. The element indicated by the move in the state matrix is changed to 1 while the other elements (residue types) of the same column (position in protein) are all set to 0. An episode contains a series of moves with the corresponding states denoted by s_t $(0 < t < T)$, and the rewards denoted by r_t $(0 < t < T)$. The state is updated iteratively until the episode reaches the end condition.

(1) The surrogate score r_t of the current state is lower than that of the previous state r_{t-1}.
(2) The current move is invalid, which means when the state sequence remains unchanged or changes to a previously generated sequence.

During MCTS simulations, a policy-value neural network f_θ guides the process. Each simulation corresponds to an episode and travels from the root node s_{root} towards a leaf node s_L in the MCTS tree, where each node represents a different state (sequence). To reach the next state s', an action a (simulation move) is performed on the current state s, and the edge between s and s' is denoted as (s, a). The selection of the next node during the simulation is based on the maximal value of $Q_{s,a} + U_{s,a}$, where $Q_{s,a}$ is evaluated and updated by the output v of the neural network and $N_{s,a}$. $Q_{s,a}$ is the action value of edge (s, a). $U_{s,a}$ is determined from

$$U_{s,a} = C_{\text{puct}} P_{s,a} \frac{\sqrt{N_{s,a}^{\text{parent}}}}{1 + N_{s,a}} \tag{1}$$

where C_{puct} is an exploration–exploitation tuning constant. $P_{s,a}$ represents the prior probability of visiting child nodes from parent node s, and is obtained by putting the parent's state into the neural network. When the search algorithm encounters a leaf node s_L, it expands the tree by creating new child nodes. Finally, the tree is updated in a backward manner, starting from the leaf node towards the root:

$$N_{s,a} = N_{s,a} + 1 \tag{2}$$

$$Q_{s,a} = \frac{Q_{s,a}\left(N_{s,a} - 1\right) + v}{Q_{s,a}} = Q_{s,a} + \frac{v - Q_{s,a}}{Q_{s,a}} \tag{3}$$

where v is evaluated from

$$(\mathbf{p}(s_L), v) = f_\theta(s_L). \tag{4}$$

When the simulations on a root node s_t are completed, a play move π_t is sampled and is proportional to the exponentiated visit counts at time-step t, $\pi_t \propto N(s_t, a)^{1/\tau}$, where τ is a tunable temperature parameter. r_t for each move is evaluated using a surrogate model. The data (s_t, π_t, r_t) of the episodes are stored in a buffer for RL. The neural network ($\mathbf{p}$, v) = $f_\theta(s)$ is trained using a combination of loss terms as

$$l = (r - v)^2 - \pi^{\mathrm{T}} \log \mathbf{p} + \gamma \|\theta\|^2 \tag{5}$$

where γ is the regularization parameter of θ. $\mathbf{p}$ and v are computed by the neural network, representing the mutation probabilities and state value, respectively. An episode ends if the reward of the current move is lower than that of the previous move, and a new episode starts with the sequence that achieved the highest reward in the previous episode for the PAB1/GFP and peptide-binder tasks. In the EvoPlay-assisted DE task and GLuc design task, a new starting sequence of an episode can be alternatively selected from a sequence pool if no fitness improvement is observed for ten consecutive episodes

Policy-value neural network. The input to the policy-value neural network is a state matrix with a shape of 20 × L, where L is the length of the input variable sites and 20 represents the length of one-hot encoded residue types. The model architecture consists of a stack of one-dimensional convolutional layers, followed by a policy head and value head (Supplementary Table 11).

Sequence design evaluated by TAPE oracle

Dataset. We utilize two datasets in this task: PAB1 (UniProt P04147) and GFP (UniProt P42212). The fitness of PAB1[43] represents its binding ability. The PAB1 dataset contains 36,523 mutants with a length of 75, and we adopt the XY Enrichment score as the fitness label. The GFP dataset[42] includes 51,715 mutant sequences with fitness representing the fluorescence intensity. We remove the first residue of all GFP sequences and keep all sequences of equal length (237). Our goal is to design mutants with high fitness scored by the oracle.

Surrogate model. The surrogate model takes a full-length protein in a one-hot encoded format as input. We use CNN[47] as the baseline model and compare it with four other models (UniRep[48], GP regression[25], MuFacNet[17], hybrid[49]), with ablation results listed in Supplementary Table 12 and a detailed CNN architecture provided in Supplementary Table 13. To avoid overfitting, we gradually increase the number of training data by 10% to update each surrogate model (more details in Benchmark methods in Supplementary Notes).

Oracle and metrics. In this task, we use TAPE[45] as the oracle, which is fine-tuned in PEX[17] to orthogonally evaluate generated sequences in silico. We calculate the cumulative maximum score (fitness) of all the sequences generated by each method. To ensure a fair benchmark, we maintain consistent query costs across all methods by aligning the querying times.

Novelty and diversity. We calculate novelty and diversity[44] metrics for sequences generated by three RL methods: EvoPlay, DyNA-PPO[39] and SAC[46]. Novelty is calculated from

$$\text{novelty}(G) = \frac{\sum_{i=1}^{n} \text{levenshtein}(s_i, s_{\text{start}})(s_i \in G)}{|G|} \tag{6}$$

where G denotes the generated sequences, $|G|$ denotes the number of generated sequences, s_i denotes the ith starting sequence and levenshtein(,) is the Levenshtein distance between the ith and jth sequences. Diversity is defined by the following equation:

$$\text{diversity}(G) = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{levenshtein}(s_i, s_j)(s_i, s_j \in G)}{|G|\,(|G| - 1)}. \tag{7}$$

Restricting search space via zero-shot predictors. Inspired by the language-model-guided antibody evolution[79], we used a zero-shot language model to select positions with high information entropy. The generated sequences were one-hot encoded and input into six ESM models (esm1b, esm1v1, esm1v2, esm1v3, esm1v4, esm1v5)[13,50]. The output represents the mutation frequency value for each site. The continuous information entropy of each position in the sequence was calculated using equation (8) with the six ESM models, and the average information entropy was calculated using equation (9). We filtered highly variable positions where Shannon entropy was greater than 3, resulting in 35 selected sites for the GFP dataset benchmark task (positions 13, 23, 37, 44, 45, 46, 55, 64, 67, 68, 71, 75, 76, 78, 79, 86, 92, 137, 146, 151, 155, 167, 175, 178, 179, 181, 182, 190, 197, 202, 215, 216, 229, 231, 232). We restricted EvoPlay to start from and search only over the design space defined by these 35 selected sites. The total number of surrogate queries was set to 4,000, and we compared the generated sequences with those obtained using full-length searching.

$$H(X_j) = -\int_x p(x_i) \log p(x_i)\, \mathrm{d}x_i \tag{8}$$

$$A_j = \frac{1}{M} \sum_{1}^{M} H(X_j) \tag{9}$$

where $H(X_j)$ denotes the Shannon entropy of position j. The analysis is performed with M language models from the Meta ESM library (esm1b, esm1v1, esm1v2, esm1v3, esm1v4, esm1v5). The method of assessing the similarity between the natural and generated sequences of PAB1 and GFP is described in Supplementary Notes.

Protein–RNA complex model preparation for MD. The structures of PAB1 and GFP were obtained from UniProt (http://www.uniprot.org/, P04147, P42212). The PyMOL2.1 package[80] was used to plot the structure of the 126–200 region (RNA recognition motifs). To build the PAB1–RNA complex structure, we superimposed it on the crystal structure of the human PAB1 (PDB 1cvj) and optimized it using MD simulation. UniDesign tools[81] were used to construct the three-dimensional structures of PAB1 mutants.

Peptide binder design using AF2 as surrogate

Receptor and peptide structure datasets. The protein–peptide complexes (PDB 1ssc, 2cnz, 3r7g, 6seo) were chosen from a group of 96 peptide–protein complexes on the basis of specific criteria such as the length of the peptide chain, distance between the peptide chain and receptor chain, and other factors. These proteins were then used in a recent study[54], where their interface structures were predicted with high accuracy using AF2.

Receptor sequence, peptide sequence, binding pocket residues. We retrieve the sequences of the receptor and peptide from the SEQRES lines of the PDB files[54,59], and then search for the receptor MSA using HHblits[82]. We select the binding pocket residues as those with at least one backbone atom located within 8.0 Å of a peptide backbone atom, using PyMOL74[80] v2.1.

AF2 as surrogate model. We use AF2[52] as the structure-based surrogate. The input is the concatenation of the receptor MSA and the single peptide sequence, containing only sequence information. We adopt AF2 v2.0, model 1, ensemble number 1, with recycle 0 for 1ssc and recycle 3 for 2cnz, 3r7g and 6seo (Supplementary Fig. 7). The pLDDT score of model 1 is used to calculate the reward function and evaluate the designed sequences. No additional refinement is performed on the models, and no structural templates are used.

EvoPlay-designed peptide binder and evaluation. In EvoPlay, the design process starts from a single sequence as the initial state s_0, and then mutates one position in one EvoPlay move. Assuming that the current state after a certain number of moves is s_t, the current move π_t (at a simulation move) will change the state from s_t to s_{t+1}. Newly generated sequences from MCTS simulations during the expansion stage are also considered as design outputs. The reward for an episode of sequential moves is calculated according to equation (10). If r_t is lower than r_{t-1}, the episode ends. A new episode starts with the highest rewarded state from the previous episode. If the same state remains as the starting sequence more than E times (E is a hyperparameter), the sequence will be randomly mutated with one to three amino acids. Additionally, the play move π_t will avoid hitting already-generated sequences. r is calculated using the following formula:

$$r = \frac{\mathrm{pLDDT}_{\mathrm{peptide}}}{(d_{\mathrm{peptide}} + d_{\mathrm{receptor}})\,\Delta\mathrm{COM}} C \qquad (10)$$

where $\mathrm{pLDDT}_{\mathrm{peptide}}$ is the average pLDDT of the AF2 output for the peptide, and d_{peptide}, d_{receptor} and ΔCOM are defined as in EvoBind[59], where the sum of d_{peptide} and d_{receptor} amounts to the distance of the binder–receptor interface between peptide and protein, and the distance between the centroids (alpha carbon centre of peptide) of the predicted peptide and the native peptide is denoted as ΔCOM, which ensures that the designed peptide does not drift too far from the receptor residues. C is a balancing constant set to 0.002. We performed five repetitions for each of the five starting sequences to benchmark against EvoBind and MCMC. A Gumbel distribution was used to generate random starting sequences and we collected 1,000 generated designs for comparison.

Binding characterization using biolayer interferometry. Kinetic parameters were measured using Gator Prime (Gatorbio). The recombinant protein was immobilized using Ni-NTA (nitrilotriacetic acid) biosensors, which were hydrated in a 96-well MAX plate (catalogue 06-0101) for 10 min. Wells containing 200 µl of Q buffer (10 mM PBS + 0.02% Tween 20 + 0.2% BSA) were used for peptide dilution. The binding assay was initiated by placing hydrated biosensors on the Gator Prime instrument and incubating them in a 250 µl black assay plate containing 200 µl of Q buffer. The biosensor was then shifted to a second position, where the wells contained 200 µl of recombinant protein diluted using Q buffer (10 µg ml^{-1}). After the loading step, the biosensor was shifted to Q-buffer-containing wells for baselining, followed by monitoring of the association of the complexes by transferring the biosensors into wells containing peptides. The dissociation of the complexes was then measured by transferring the biosensor back into buffer. Biosensors were regenerated after each measurement. The sensors specifically bind to the recombinant RNase1 (PDB 1ssc) but not to peptides. We determined the optimal concentration of the protein and peptides by twofold dilution of the peptides to concentrations ranging from 320 µM to 5 µM in Q buffer. Finally, biosensors were hydrated, and the interactions were measured. The data were analysed using the Gator Prime analysing module, and the K_D value was calculated. Biolayer interferometry experiments were conducted by AlpalifeBio.

EvoPlay-assisted DE for GB1 and PhoQ

Datasets. We utilize two datasets, namely GB1 (UniProt P19909) and PhoQ (UniProt P23837), for the MLDE task. These datasets consist of combinatorial libraries of four sites, resulting in a design space of 20^4, which equates to 160,000 possible variants. The GB1 dataset comprises 149,361 variants (V39, D40, G41 and V54) that have been experimentally labelled with fitness defined as binding affinity to IgG-Fc. The PhoQ dataset contains 140,517 variants (A284, V285, S288 and T289), with fitness defined as fluorescence levels related to the signalling activity of kinase PhoQ interacting with PhoP in a regulatory system[83,84]. Fitness values for both datasets have been normalized to a range of [0, 1], with 92% of the variants having fitness values lower than 0.01 and over 99% with fitness values lower than 0.3. Although the PhoQ dataset has fewer high-fitness variants, it exhibits a greater diversity of high-fitness sequences.

Surrogate model and episode starting sequence pool. The input consists of four-site sequences encoded in one-hot format. We utilize GP as a surrogate model to score mutants generated by play moves of EvoPlay. The GP surrogate is trained in an active learning manner, gradually updating the model by adding 96 generated mutants in each round. In the first round, 96 variants are generated through K-means clustering sampling, while a subsequent 288 variants are generated through EvoPlay sampling. If the same sequence is used in more than ten episodes, a new sequence from the starting sequence pool will be selected as the new episode start. Otherwise, the next episode will start with the sequence that scored the highest in the previous episode. Simulation times are set to 30. An ablation study comparing GP[25], CNN[47], MuFacNet[17], EVmutation[85] (zero shot) and MSA Transformer[14] (zero shot) as surrogate models is included. For EVmutation, predictions of the combinatorial libraries are calculated using the EVcouplings webapp[63], and MSA Transformer predictions are calculated using the naive probability using mask-filling protocols[14].

Evaluation and metrics. We sequentially selected 480 sequences from the library, with 384 sequences generated by the first-stage sampling, including 288 sequences from EvoPlay sampling and 96 sequences from K-means clustering as the MCTS starting sequence pool, and 96 sequences selected from the predictions of the second-stage supervised learning. In 500 independent repeats for both methods without cherry-picking, we evaluated the predicted maximum fitness and global maximal fitness hit count on the union of the 384 sequences generated in the first stage and the 96 sequences predicted in the second stage. Predicted mean fitness was evaluated only on the top 96 sequences from the supervised ensemble models in the second stage.

GLuc designed by EvoPlay

Materials. The protein sequence of the WT GLuc gene (UniProt Q9BLZ2) excluding the secretion tag (17 residues) is KPTENNEDFNIVAVASNFATTDLDADRGKLPGKKLPLEVLKEMEANARKAGCTRGCLICLSHIKCTPKMKKFIPGRCHTYEGDKESAQGGIGEAIVDIPEIPGFKDLEPMEQFIAQVDLCVDCTTGCLKGLANVQCSDLLKKWLPQRCATFASKIQGQVDKIKGAGGD.

The standard chemicals were purchased from BBI, while the enzymes for standard cloning procedures were purchased from TOYOBO, Thermo Fisher Scientific and New England Biolabs (NEB). The Plasmid Mini Kit II was bought from Omega, and the pCold plasmid was obtained from Takara. The Origami B (DE3) host cells were supplied by Shanghai Weidi Biotechnology, and the CTZ was synthesized by MGI.

Library construction for GLuc mutants. We used the online DeepMSA v1 pipeline[66] service (https://zhanglab.ccmb.med.umich.edu/DeepMSA/) to obtain a 30-sequence MSA, including the WT sequence. However, the small number of detected MSAs resulted in a normalized number of effective sequences, N_f, of only 0.86. We aligned the MSAs using Clustal Omega and generated a sequence logo using the conservation analysis tool WebLogo[86] (https://weblogo.berkeley.edu/logo.cgi). The sequence logo represents each column of the alignment as a stack of letters, where the height of each letter is proportional to its frequency, and the overall height of each stack is proportional to the sequence conservation, which is defined by the entropy of the letter distribution[87] (refer to Supplementary Data 1 for the generated MSAs and Extended Data Fig. 1 for the sequence logo).

Amino acids at non-conserved regions were selected to build three oligonucleotide pools (synthesized by BGI), namely L2 (Phe 9, Asn 10, Val 12, Ala 13, Ser 16, Ala 19, Thr 20, Leu 23), L3 (Lys 49, His 62, Pro 67, Glu 85, Glu 93, Asp 106, Leu 107, Val 121) and L4 (Gln 135, Gln 146, Thr 150, Ser 153, Gly 157, Gln 158, Lys 161, Ala 165, Gly 166). The luciferase mutant libraries for L2, L3 and L4 were amplified in DH5α cells and expressed in Origami B (DE3) using recombinant plasmids constructed from PCR products of oligonucleotide pools and pCold vector using Gibson Assembly (NEB) (primer list 1). Mutants were also obtained through combination of the three libraries and the sites mentioned below. For instance, a catalytic cavity was hypothesized by heteronuclear NMR[67], which was made up of 19 residues: Asn 10, Val 12, Ala 13, Val 14, Ser 16, Asn 17, Phe 18, Leu 60, Ser 61, Ile 63, Lys 64, Cys 65, Arg 76, Cys 77, His 78, Thr 79, Phe 113, Ile 114, Val 117. Of these, nine residues (Asn 10, Val 12, Ala 13, Val 14, Ser 16, Asn 17, His 78, Thr 79, Ile 114) were selected, along with six residues (Ser 86, Ala 87, Gln 88, Gly 89, Gly 92, Ile 95) from the loop related to the entrance to the cavity. However, due to PCR errors, additional changes (Ala 15, Pro 36, Lys 71, Glu 81, Glu 111) were introduced during the construction of the mutant library. Furthermore, an active mutant B6 was isolated (H62K, P67L, E85S, S86T, A87G, E93P, L107M, V121E), from which a dozen mutants were generated by combining B6 with other sites. Site-directed mutants were also generated on the basis of the WT luciferases (Asn 10, Val 12, His 78, Thr 79, Ala 87, Gln 88, Gly 89, Ile 114). After removing mutants with the loosely packed moiety (sites 150–168), a pool of 164 mutants was obtained for surrogate model training (Supplementary Data 2).

GLuc input data preprocessing for surrogate model. After aligning the mutation sites of the 164 mutants, the design space was narrowed down to 32 residue positions, namely positions 9, 10, 12–17, 19, 20, 23, 36, 49, 62, 67, 71, 78, 79, 81, 85–89, 92, 93, 95, 106, 107, 111, 114 and 121. These 32 sites were combined to form the input sequence, and the 164 sequences were used to train the surrogate model. During the design process, a single sequence in the current state was represented by a 20 × 32 matrix after one-hot encoding. Therefore, the environment state of this task is a one-hot 20 × 32 matrix, where a single-site mutation on the matrix corresponds to one play move.

Surrogate model and starting sequence pool. One-hot encoding is used to encode recombination sequences. We choose GP as the surrogate model to score EvoPlay moves. When no better scored sequences are generated and the same sequence starts for more than ten episodes, the next episode will randomly select one restarting seed from those 164 recombination sequences. The EvoPlay simulation times are set to 30. Only sequences generated by EvoPlay play moves are kept for downstream evaluation.

GLuc mutant screening and protein purification. All mutants from the three libraries were amplified by PCR, followed by digestion with DpnI (NEB) at 37 °C for 1 h. The resulting PCR products were then transformed into DH5α and cultured in lysogeny broth plates overnight. After single clones were picked and sequenced, plasmids were extracted and transformed into Origami B (DE3) host cells. Single colonies were then inoculated in 3 ml of lysogeny broth medium (5 g of yeast extract, 10 g of tryptone and 10 g of NaCl per litre) with ampicillin (100 mg l^{-1}) at 37 °C and 250 r.p.m. for 3–4 h. Isopropyl-β-D-thiogalactopyranoside was added to a final concentration of 0.01 mM, and the temperature was lowered to 16 °C when the optical density at 600 nm reached 0.5. After 16 h of incubation, cells were collected by centrifugation (10,000 *g*, 4 °C for 10 min) and resuspended in buffer (50 mM Tris-HCl, pH 7.5; 250 mM NaCl, 1 mM phenylmethyl sulfonyl fluoride). The ratio of cell pellet to buffer was 1:10 (1 g of cell pellet to 10 ml of buffer). The cells were then sonicated in a cold-water bath for 20 min to rupture the cell membranes, followed by centrifugation at 10,000 *g*, 4 °C for 30 min to separate the soluble fraction (supernatant) and insoluble fraction (pellet). Protein in the supernatant was purified using a His MultiTrap FF 96-well filter plate (Cytiva GE), washed with buffer (50 mM Tris, pH 8.0, 250 mM NaCl, 10 mM imidazole) three to five times (500 µl each time) and eluted with 100 µl of 300 mM imidazole (50 mM Tris, pH 8.0, 250 mM NaCl, 10 mM imidazole). The 300 mM imidazole buffer was then removed using 1×PBS through an Ultracel-3 regenerated cellulose membrane (Millipore). The total protein was quantified using Bradford reagent (Bio-Rad) and analysed using 12% SDS–PAGE followed by Coomassie brilliant blue staining.

Bioluminescence measurement. The bioluminescence signals from different mutants were measured using a BioTek Synergy H1 microplate reader. To begin with, 10 µl of protein solution (10 µg ml^{-1}) was added to a flat-bottom 96-well black polystyrene plate (Thermo Fisher Scientific) and diluted with luciferase dilution buffer (50 mM Tris-HCl, 100 mM NaCl, 0.1% (v/v) Tween-20, pH 8.0). The bioluminescence signals were detected immediately after adding 90 µl of CTZ (50 µM, MGI) to each well at room temperature using the sample injection probe of the Synergy H1.

GLuc MD model preparation. The GLuc used in this study was obtained from the RCSB PDB (https://www.rcsb.org/, 7d2o). To determine the binding orientation of the top hit compound in different docking poses, GLuc was docked with CTZ using Autodock4.2 (ref. 88). Polar hydrogens were added, and the Lamarckian algorithm was applied. The grid size dimensions were set to 40 × 40 × 40 *xyz* points with a grid spacing of 0.375 Å. The docking centre coordinates (*x*, *y* and *z*), 3.282, −0.061, −7.119, were obtained from key interacting residues (Arg 76, His 78, Thr 79)[67]. To obtain a stable complex model of GLuc–CTZ, the top five poses with the lowest binding energy or higher binding affinity were extracted and aligned for MD simulation. Three-dimensional structures of GLuc mutants (MT1—V12S, K49R, H62K, P67A, E85S, S86T, A87G, E93P, L107M, V121E; MT2—N10S, V12A) were constructed using UniDesign tools.

MD simulations
All MD simulations for the target systems are performed using the Gromacs[89–96]. The tools and settings of MD simulations are detailed in Supplementary Notes.

Reporting summary
Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability
Datasets used in all benchmark studies have been published previously. The PAB1 (UniProt P04147) and GFP (UniProt P42212) datasets used for EvoPlay surrogate training have been archived[97]. The four-site library GB1 (UniProt P19909) and PhoQ (UniProt P23837) datasets have been provided[98]. Peptide and receptor sequences of the WT for 1ssc, 2cnz, 3r7g and 6seo (PDB) have been referenced[99]. The MSA and logo used in GLuc design are presented in Supplementary Data 1 and Extended Data Fig. 1. Our in-house GLuc library can be found in Supplementary Data 2, and experimentally validated variants proposed by EvoPlay are presented in Supplementary Data 3. Source data are provided with this paper[100].

Code availability
EvoPlay is written in Python (v3.8.10) using the PyTorch (v1.12.1) library. The source code is available on GitHub (https://github.com/melobio/EvoPlay) under the GPLv3 license. The doi of the GitHub repository for EvoPlay is provided with the Zenodo link[101]. The Code Ocean capsule for EvoPlay is also published[102].

References

1. Romero, P. A. & Arnold, F. H. Exploring protein fitness landscapes by directed evolution. *Nat. Rev. Mol. Cell Biol.* **10**, 866–876 (2009).
2. Wu, Z., Kan, S. J., Lewis, R. D., Wittmann, B. J. & Arnold, F. H. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proc. Natl Acad. Sci. USA* **116**, 8852–8858 (2019).
3. Yang, K. K., Wu, Z. & Arnold, F. H. Machine-learning-guided directed evolution for protein engineering. *Nat. Methods* **16**, 687–694 (2019).
4. Luo, Y. et al. ECNet is an evolutionary context-integrated deep learning framework for protein engineering. *Nat. Commun.* **12**, 5743–5756 (2021).
5. Greenhalgh, J. C., Fahlberg, S. A., Pfleger, B. F. & Romero, P. A. Machine learning-guided acyl-ACP reductase engineering for improved in vivo fatty alcohol production. *Nat. Commun.* **12**, 5825–5834 (2021).
6. Wittmann, B. J., Yue, Y. & Arnold, F. H. Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell Syst.* **12**, 1026–1045 (2021).
7. Hie, B. L. & Yang, K. K. Adaptive machine learning for protein engineering. *Curr. Opin. Struct. Biol.* **72**, 145–152 (2022).
8. Qiu, Y., Hu, J. & Wei, G.-W. Cluster learning-assisted directed evolution. *Nat. Comput. Sci.* **1**, 809–818 (2021).
9. Kawashima, S. & Kanehisa, M. AAindex: amino acid index database. *Nucleic Acids Res.* **28**, 374 (2000).
10. Ofer, D. & Linial, M. ProFET: feature engineering captures high-level protein functions. *Bioinformatics* **31**, 3429–3436 (2015).
11. Georgiev, A. G. Interpretable numerical descriptors of amino acid space. *J. Comput. Biol.* **16**, 703–723 (2009).
12. Elnaggar, A. et al. ProtTrans: toward understanding the language of life through self-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 7112–7127 (2022).
13. Rives, A. et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl Acad. Sci. USA* **118**, e2016239118 (2021).
14. Rao, R. M. et al. MSA Transformer. *Proc. Mach. Learning Res.* **139**, 8844–8856 (2021).
15. Sinai, S. et al. AdaLead: a simple and robust adaptive greedy search algorithm for sequence design. Preprint at https://arxiv.org/abs/2010.02141 (2020).
16. Biswas, S., Khimulya, G., Alley, E. C., Esvelt, K. M. & Church, G. M. Low-N protein engineering with data-efficient deep learning. *Nat. Methods* **18**, 389–396 (2021).
17. Ren, Z. et al. Proximal exploration for model-guided protein sequence design. *Proc. Mach. Learning Res.* **162**, 18520–18536 (2022).
18. Anishchenko, I. et al. De novo protein design by deep network hallucination. *Nature* **600**, 547–552 (2021).
19. Zeming, L. et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).
20. Verkuil, R. et al. Language models generalize beyond natural proteins. Preprint at *bioRxiv* https://doi.org/10.1101/2022.12.21.521521 (2022).
21. Hie, B. et al. A high-level programming language for generative protein design. Preprint at *bioRxiv* https://doi.org/10.1101/2022.12.21.521526 (2022).
22. González, J. et al. Batch Bayesian optimization via local penalization. *Proc. Mach. Learning Res.* **51**, 648–657 (2016).
23. Hie, B., Bryson, B. D. & Berger, B. Leveraging uncertainty in machine learning accelerates biological discovery and design. *Cell Syst.* **11**, 461–477 (2020).
24. Williams, C. K. & Rasmussen, C. E. *Gaussian Processes for Machine Learning* (MIT Press, 2006).
25. Romero, P. A., Krause, A. & Arnold, F. H. Navigating the protein fitness landscape with Gaussian processes. *Proc. Natl Acad. Sci. USA* **110**, E193–E201 (2013).
26. Bryant, D. H. et al. Deep diversification of an AAV capsid protein by machine learning. *Nat. Biotechnol.* **39**, 691–696 (2021).
27. Brookes, D. H. & Listgarten, J. Design by adaptive sampling. Preprint at https://arxiv.org/abs/1810.03714 (2018).
28. Brookes, D., Park, H. & Listgarten, J. Conditioning by adaptive sampling for robust design. *Proc. Mach. Learning Res.* **97**, 773–782 (2019).
29. Castro, E. et al. Transformer-based protein generation with regularized latent space optimization. *Nat. Mach. Intell.* **4**, 840–851 (2022).
30. Browne, C. B. et al. A survey of Monte Carlo tree search methods. *IEEE Trans. Comput. Intell. AI Games* **4**, 1–43 (2012).
31. Silver, D. et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).
32. Silver, D. et al. Mastering the game of Go without human knowledge. *Nature* **550**, 354–359 (2017).
33. Silver, D. et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* **362**, 1140–1144 (2018).
34. Mirhoseini, A. et al. A graph placement methodology for fast chip design. *Nature* **594**, 207–212 (2021).
35. Degrave, J. et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature* **602**, 414–419 (2022).
36. Shree Sowndarya, S. V. et al. Multi-objective goal-directed optimization of de novo stable organic radicals for aqueous redox flow batteries. *Nat. Mach. Intell.* **4**, 720–730 (2022).
37. Fawzi, A. et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature* **610**, 47–53 (2022).
38. Feng, S. et al. Dense reinforcement learning for safety validation of autonomous vehicles. *Nature* **615**, 620–627 (2023).
39. Angermueller, C. et al. Model-based reinforcement learning for biological sequence design. In *International Conference on Learning Representations* (eds A. Rush), April 1–23 (ICLR, 2020).
40. Isaac, I. D. et al. Top-down design of protein architectures with reinforcement learning. *Science* **380**, 266–273 (2023).
41. Nakatsu, T. et al. Structural basis for the spectral difference in luciferase bioluminescence. *Nature* **440**, 372–376 (2006).
42. Sarkisyan, K. S. et al. Local fitness landscape of the green fluorescent protein. *Nature* **533**, 397–401 (2016).
43. Melamed, D., Young, D. L., Gamble, C. E., Miller, C. R. & Fields, S. Deep mutational scanning of an RRM domain of the *Saccharomyces cerevisiae* poly(A)-binding protein. *RNA* **19**, 1537–1551 (2013).
44. Jain, M. et al. Biological sequence design with GFlowNets. *Proc. Mach. Learning Res.* **162**, 9786–9801 (2022).
45. Rao, R. et al. Evaluating protein transfer learning with TAPE. *Adv. Neural Inf. Process Syst.* **32**, 9689–9701 (2019).
46. Haarnoja, T., Zhou, A., Abbeel, P. & Levine, S. Soft Actor-Critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. *Proc. Mach. Learning Res.* **80**, 1861–1870 (2018).
47. Shanehsazzadeh, A., Belanger, D. & Dohan, D. Is transfer learning necessary for protein landscape prediction? Preprint at https://arxiv.org/abs/2011.03443 (2020).
48. Alley, E. C., Khimulya, G., Biswas, S., AlQuraishi, M. & Church, G. M. Unified rational protein engineering with sequence-based deep representation learning. *Nat. Methods* **16**, 1315–1322 (2019).
49. Illig, A.-M., Siedhoff, N. E., Schwaneberg, U. & Davari, M. D. A hybrid model combining evolutionary probability and machine learning leverages data-driven protein engineering. Preprint at *bioRxiv* https://doi.org/10.1101/2022.06.07.495081 (2022).

50. Meier, J. et al. Language models enable zero-shot prediction of the effects of mutations on protein function. In *Advances in Neural Information Processing Systems 34* (eds M. Ranzato), 29287–29303 (NeurIPS, 2021).
51. Linding, R. et al. Protein disorder prediction: implications for structural proteomics. *Structure* **11**, 1453–1459 (2003).
52. Jumper, J. et al. Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (2021).
53. Evans, R. et al. Protein complex prediction with AlphaFold-Multimer. Preprint at *bioRxiv* https://doi.org/10.1101/2021.10.04.463034 (2022).
54. Tsaban, T. et al. Harnessing protein folding neural networks for peptide–protein docking. *Nat. Commun.* **13**, 176 (2022).
55. Jendrusch, M., Korbel, J. O. & Sadiq, S. K. AlphaDesign: a de novo protein design framework based on AlphaFold. Preprint at *bioRxiv* https://doi.org/10.1101/2021.10.11.463937 (2021).
56. Wicky, B. et al. Hallucinating symmetric protein assemblies. *Science* **378**, 56–61 (2022).
57. Dauparas, J. et al. Robust deep learning-based protein sequence design using ProteinMPNN. *Science* **378**, 49–56 (2022).
58. Bennett, N. R. et al. Improving de novo protein binder design with deep learning. *Nat. Commun.* **14**, 2625–2633 (2023).
59. Bryant, P. & Elofsson, A. EvoBind: in silico directed evolution of peptide binders with AlphaFold. Preprint at *bioRxiv* https://doi.org/10.1101/2022.07.23.501214 (2022).
60. Mirdita, M. et al. ColabFold: making protein folding accessible to all. *Nat. Methods* **19**, 679–682 (2022).
61. Tareen, A. & Kinney, J. B. Logomaker: beautiful sequence logos in Python. *Bioinformatics* **36**, 2272–2274 (2020).
62. Miller, B. R. III et al. MMPBSA.py: an efficient program for end-state free energy calculations. *J. Chem. Theory Comput.* **8**, 3314–3321 (2012).
63. Hopf, T. A. et al. The EVcouplings Python framework for coevolutionary sequence analysis. *Bioinformatics* **35**, 1582–1584 (2019).
64. Welsh, J. P., Patel, K. G., Manthiram, K. & Swartz, J. R. Multiply mutated *Gaussia* luciferases provide prolonged and intense bioluminescence. *Biochem. Biophys. Res. Commun.* **389**, 563–568 (2009).
65. Kim, S. B., Suzuki, H., Sato, M. & Tao, H. Superluminescent variants of marine luciferases for bioassays. *Anal. Chem.* **83**, 8732–8740 (2011).
66. Zhang, C., Zheng, W., Mortuza, S., Li, Y. & Zhang, Y. DeepMSA: constructing deep multiple sequence alignment to improve contact prediction and fold-recognition for distant-homology proteins. *Bioinformatics* **36**, 2105–2112 (2020).
67. Wu, N. et al. Solution structure of *Gaussia* luciferase with five disulfide bonds and identification of a putative coelenterazine binding cavity by heteronuclear NMR. *Sci. Rep.* **10**, 20069 (2020).
68. Lu, H. et al. Machine learning-aided engineering of hydrolases for PET depolymerization. *Nature* **604**, 662–667 (2022).
69. Norn, C. et al. Protein sequence design by conformational landscape optimization. *Proc. Natl Acad. Sci. USA* **118**, e2017228118 (2021).
70. Hsu, C. et al. Learning inverse folding from millions of predicted structures. *Proc. Mach. Learning Res.* **162**, 8946–8970 (2022).
71. Makowski, E. K. et al. Co-optimization of therapeutic antibody affinity and specificity using machine learning models that generalize to novel mutational space. *Nat. Commun.* **13**, 3788 (2022).
72. Markel, U. et al. Advances in ultrahigh-throughput screening for directed enzyme evolution. *Chem. Soc. Rev.* **49**, 233–262 (2020).
73. Gérard, A. et al. High-throughput single-cell activity-based screening and sequencing of antibodies using droplet microfluidics. *Nat. Biotechnol.* **38**, 715–721 (2020).
74. Dörr, M. et al. Fully automatized high-throughput enzyme library screening using a robotic platform. *Appl. Biochem. Biotechnol.* **113**, 1421–1432 (2016).
75. Wittmann, B. J. et al. evSeq: cost-effective amplicon sequencing of every variant in a protein library. *ACS Synth. Biol.* **11**, 1313–1324 (2022).
76. Ingraham, J., Garg, V., Barzilay, R. & Jaakkola, T. Generative models for graph-based protein design. In *Advances in Neural Information Processing Systems 32* (eds H. Wallach et al.) 15820–15831 (NeurIPS, 2019).
77. Wang, J. et al. Scaffolding protein functional sites using deep learning. *Science* **377**, 387–394 (2022).
78. Bell, E. L. et al. Biocatalysis. *Nat. Rev. Methods Primers* **1**, 45 (2021).
79. Hie, B. L. et al. Efficient evolution of human antibodies from general protein language models and sequence information alone. *Nat. Biotechnol.* https://doi.org/10.1038/s41587-023-01763-2 (2023).
80. The PyMOL Molecular Graphics System v.1.2 r3pre (Schrödinger, 2011).
81. Huang, X., Pearce, R. & Zhang, Y. EvoEF2: accurate and fast energy function for computational protein design. *Bioinformatics* **36**, 1135–1142 (2020).
82. Steinegger, M. et al. HH-suite3 for fast remote homology detection and deep protein annotation. *BMC Bioinform.* **20**, 473 (2019).
83. Podgornaia, A. I. & Laub, M. T. Pervasive degeneracy and epistasis in a protein–protein interface. *Science* **347**, 673–677 (2015).
84. Bergstra, J., Yamins, D. & Cox, D. Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. *Proc. Mach. Learning Res.* **28**, 115–123 (2013).
85. Hopf, T. A. et al. Mutation effects predicted from sequence co-variation. *Nat. Biotechnol.* **35**, 128–135 (2017).
86. Crooks, G. E., Hon, G., Chandonia, J.-M. & Brenner, S. E. WebLogo: a sequence logo generator. *Genome Res.* **14**, 1188–1190 (2004).
87. Schneider, T. D. & Stephens, R. M. Sequence logos: a new way to display consensus sequences. *Nucleic Acids Res.* **18**, 6097–6100 (1990).
88. Morris, G. et al. AutoDock4 and AutoDockTools4: automated docking with selective receptor flexibility. *J. Comput. Chem.* **30**, 2785–2791 (2009).
89. Van Der Spoel, D. et al. GROMACS: fast, flexible, and free. *J. Comput. Chem.* **26**, 1701–1718 (2005).
90. Lindorff-Larsen, K. et al. Improved side-chain torsion potentials for the Amber ff99SB protein force field. *Proteins* **78**, 1950–1958 (2010).
91. Lu, T. Sobtop v.1.0 (dev3.1) http://sobereva.com/soft/Sobtop (2022).
92. Neese, F. Software update: the ORCA program system—Version 5.0. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **12**, e1606 (2022).
93. Jorgensen, W. L., Chandrasekhar, J., Madura, J. D., Impey, R. W. & Klein, M. L. Comparison of simple potential functions for simulating liquid water. *J. Chem. Phys.* **79**, 926–935 (1983).
94. Darden, T., York, D. & Pedersen, L. Particle mesh Ewald: an *N*·log(*N*) method for Ewald sums in large systems. *J. Chem. Phys.* **98**, 10089–10092 (1993).
95. Bussi, G., Donadio, D. & Parrinello, M. Canonical sampling through velocity rescaling. *J. Chem. Phys.* **126**, 014101 (2007).
96. Parrinello, M. & Rahman, A. Polymorphic transitions in single crystals: a new molecular dynamics method. *J. Appl. Phys.* **52**, 7182–7190 (1981).
97. Huang, L. GFP & PAB1 training data of EvoPlay *Figshare* https://doi.org/10.6084/m9.figshare.23498195 (2023).
98. Huang, L GB1 & PhoQ data of EvoPlay *Figshare* https://doi.org/10.6084/m9.figshare.21767369.v3 (2023).

99. Huang, L. Peptide and receptor sequences of the wild type for 1ssc, 2cnz, 3r7g and 6seo *Figshare* https://doi.org/10.6084/m9.figshare.23375666.v1 (2023).
100. Huang, L. EvoPlay Figs. 2–5 Source Data *Figshare* https://doi.org/10.6084/m9.figshare.23437295.v1 (2023).
101. melobio. (2023). melobio/EvoPlay: v1.0.0 (v1.0.0) *Zenodo* https://doi.org/10.5281/zenodo.8059425 (2023).
102. Meng, Y. Self-play reinforcement learning guides protein engineering *Code Ocean* https://doi.org/10.24433/CO.1846781.v2 (2023).

Acknowledgements
This research is supported by the Ministry of Science and Technology of the People's Republic of China's programme titled 'National Key Research and Development Program of China' (2022YFF1202200, 2022YFF1202203) and the Science, Technology and Innovation Commission of Shenzhen Municipality under grant JSGGZD20220822095802006.

Author contributions
M.Y. conceived the problem and designed all detailed studies. Y.W., H.T., L.H. and L.Y. performed analysis. L.P. conducted luciferase validation experiments. F.M. and H.Y. provided strategic guidance. M.Y. wrote the manuscript and Y.W. made modifications.

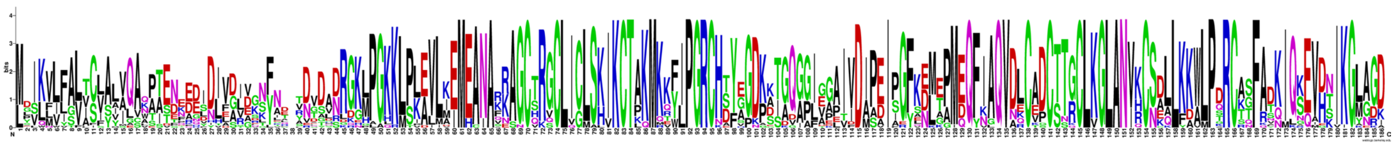
Competing interests
29 newly designed and validated GLuc mutants are undergoing patent filing (PCT/CN2023/087445). F.M. declares stock holdings in MGI. The remaining authors declare no competing interests.

Additional information
Extended data is available for this paper at https://doi.org/10.1038/s42256-023-00691-9.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42256-023-00691-9.

Correspondence and requests for materials should be addressed to Meng Yang or Feng Mu.

Peer review information *Nature Machine Intelligence* thanks Christian Dallago, Dominik Niopek and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.



Extended Data Fig. 1 | Logo indicating the frequencies of amino acids from 30 MSAs of GLuc. The sequence logo represents each column of the MSA alignment as a stack of letters, where the height of each letter is proportional to its frequency, and the overall height of each stack is proportional to the entropy of the letter distribution.

nature portfolio

Corresponding author(s):	YANG MENG
Last updated by author(s):	Jun 10, 2023

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed	
☐	☒	The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☐	☒	A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒	The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐	A description of all covariates tested
☐	☒	A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒	A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☐	☒	For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☐	☒	For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐	For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☐	☒	Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	We used Python(3.8) to process and collect the data in this study.
Data analysis	We have made the code available on GitHub. The repository can be found at https://github.com/melobio/EvoPlay The trained weights of TAPE Oracle are obtained from https://github.com/HeliXonProtein/proximal-exploration. The trained weights of ESM (1b,1v) models for preselecting action sites are obtained from https://github.com/facebookresearch/esm. The implementation of benchmarking methods(Evolutionary-BO, Cbas, DyNA-PPO, Adalead, SAC) are based on FLEXS library https://github.com/samsinai/FLEXS. We use AlphaFold2(2.0) as surrogate predictor for the design of peptide-receptor interface structure. The modification of EvoBind with recycle 0 and 3 are based on https://github.com/patrickbryant1/EvoBind. The implementation of MCMC in peptide design are based on the repository https://github.com/gjoni/trDesign. The benchmarking method of CLADE for generating 384 training sequences for ftMLDE supervised learning is as the implementation of https://github.com/WeilabMSU/CLADE. For generating the 96 sequences of 2nd stage of EvoPlay-assisted MLDE, we use the supervised learning of ensemble models from https://github.com/fhalab/MLDE. We utilize DeepMSA v1 pipeline service (https://zhanglab.ccmb.med.umich.edu/DeepMSA/) to obtain a 30-sequence multiple sequence alignment (MSA) and use Weblogo 2.8.2(https://weblogo.berkeley.edu/logo.cgi) to generate sequence logo for Gaussia luciferase (GLuc). The implementation of UniRep in ablation experiments of surrogate models are based on the repository https://github.com/churchlab/UniRep .

The implementation of Hybrid in ablation experiments of surrogate models are based on the repository https://github.com/Protein-Engineering-Framework/Hybrid_Model.
The implementation of MuFacNet in ablation experiments of surrogate models are based on the repository https://github.com/HeliXonProtein/proximal-exploration.

We adopt the following software for Molecular Dynamic Simulation and analysis: Gromacs (2020.1),Sobtop (dev3.1),ORCA (5.0),PyMOL(2.1), Autodock (4.2).

The main algorithms related to molecular simulation include the following:
Lamarckian algorithm,
Particle Mesh Ewald (PME) algorithm
Steepest descent algorithm
Velocity rescaling algorithm
Parrinello-Rahman barostat algorithm
Linear constraint solver (LINCS) algorithm
UniDesign

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our policy

Datasets used in all benchmark studies were published previously.

The PAB1 (UniProt ID: P04147) and GFP (UniProt ID: P42212) datasets was obtained from https://doi.org/10.6084/m9.figshare.23498195.

Peptide and receptor sequences of the wild type for 1ssc,2cnz,3r7g and 6seo (PDB id) can be referred to FigShare (doi: https://doi.org/10.6084/m9.figshare.23375666.v1)

GB1 (UniProt ID: P19909) and PhoQ (UniProt ID: P23837) datasets were obtained from https://doi.org/10.6084/m9.figshare.21767369.v3.

The MSA and logo used in GLuc design is presented in supplementary data 1 and Extended Data Fig. 1.

Our in-house GLuc library can be referred to Supplementary Data 2.

Experimentally validated variants proposed by EvoPlay are presented in Supplementary Data 3

Source Data are provided by the link: https://doi.org/10.6084/m9.figshare.23437295.v1

Human research participants

Policy information about studies involving human research participants and Sex and Gender in Research.

Reporting on sex and gender	n/a
Population characteristics	n/a
Recruitment	n/a
Ethics oversight	n/a

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	The sample sizes were sufficient for analysis, and no statistical methods were used to predetermine sample size.
Data exclusions	1) For GLuc experiment, samples with mutating sites within the loosely packed moiety (sites 150-168) of GLuc were excluded during in-house pool construction. Samples that were not successfully constructed or purified were excluded. 2) For bio-layer interferometry validation test, no data were excluded.
Replication	1)For the GLuc experiments, results were reproduced twice independently using different batches of purified proteins on separated days, and for each independent experiment, bioluminescence measurements were repeated three times successfully with similar results. 2)Biolayer interferometry experiment were independently repeated at least twice, and data were collected in triplicate over several days. All attempts were successful. The protein–peptide interaction kinetics were assessed at various concentrations. All data were reproducible.
Randomization	Samples were randomly separated into experimental groups.
Blinding	All experiments were performed by researchers unaware of the identities of the samples(researchers unaware of the predicted fitness score predicted by models and even which model a sample was generated).

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study	
☒	☐	Antibodies
☒	☐	Eukaryotic cell lines
☒	☐	Palaeontology and archaeology
☒	☐	Animals and other organisms
☒	☐	Clinical data
☒	☐	Dual use research of concern

Methods

n/a	Involved in the study	
☒	☐	ChIP-seq
☒	☐	Flow cytometry
☒	☐	MRI-based neuroimaging