PLOS ONE

RESEARCH ARTICLE

Rosetta:MSF:NN: Boosting performance of multi-state computational protein design with a neural network

Julian Nazet, Elmar Lang, Rainer Merkl*

Institute of Biophysics and Physical Biochemistry, University of Regensburg, Regensburg, Germany

* rainer.merkl@ur.de

OPEN ACCESS

Citation: Nazet J, Lang E, Merkl R (2021) Rosetta:MSF:NN: Boosting performance of multi-state computational protein design with a neural network. PLoS ONE 16(8): e0256691. https://doi.org/10.1371/journal.pone.0256691

Editor: Yang Zhang, University of Michigan, UNITED STATES

Received: December 23, 2020

Accepted: August 12, 2021

Published: August 26, 2021

Data Availability Statement: Relevant data are within the paper and its Supporting Information files. Data of the benchmark can be found here: https://github.com/JulianNazet/Benchmark_Rosetta-MSF-NN.

Funding: RM, ME 2259/4-1 Deutsche Forschungsgemeinschaft https://www.dfg.de/. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

Abstract

Rational protein design aims at the targeted modification of existing proteins. To reach this goal, software suites like Rosetta propose sequences to introduce the desired properties. Challenging design problems necessitate the representation of a protein by means of a structural ensemble. Thus, Rosetta multi-state design (MSD) protocols have been developed wherein each state represents one protein conformation. Computational demands of MSD protocols are high, because for each of the candidate sequences a costly three-dimensional (3D) model has to be created and assessed for all states. Each of these scores contributes one data point to a complex, design-specific energy landscape. As neural networks (NN) proved well-suited to learn such solution spaces, we integrated one into the framework `Rosetta:MSF` instead of the so far used genetic algorithm with the aim to reduce computational costs. As its predecessor, `Rosetta:MSF:NN` administers a set of candidate sequences and their scores and scans sequence space iteratively. During each iteration, the union of all candidate sequences and their Rosetta scores are used to re-train NNs that possess a design-specific architecture. The enormous speed of the NNs allows an extensive assessment of alternative sequences, which are ranked on the scores predicted by the NN. Costly 3D models are computed only for a small fraction of best-scoring sequences; these and the corresponding 3D-based scores replace half of the candidate sequences during each iteration. The analysis of two sets of candidate sequences generated for a specific design problem by means of a genetic algorithm confirmed that the NN predicted 3D-based scores quite well; the Pearson correlation coefficient was at least 0.95. Applying `Rosetta:MSF:NN:enzdes` to a benchmark consisting of 16 ligand-binding problems showed that this protocol converges ten-times faster than the genetic algorithm and finds sequences with comparable scores.

Introduction

Computational protein design has become an important tool in molecular biology [1]. Different approaches and protocols have proven their reliability for a broad range of applications. To name just a few design problems, for a protein under study, the informed user can increase

thermostability [2, 3], alter the binding of ligands [4, 5], redesign interactions with other proteins [6, 7] or design novel catalytic sites [8]. Moreover, the *de novo* design of catalytically active proteins is feasible [9, 10] as well as antibody redesign [11, 12].

For challenging design problems that require the modelling of structural flexibility, the traditional, single-state design (SSD) strategy that is optimal for finding a sequence for one structurally fixed backbone is not sufficient. For example, enzymes adopt different conformations during a catalytic cycle and more generally, all important biological effects are best represented by an ensemble of conformational states; for a review see [13]. This is why multi-state design (MSD) protocols have been introduced, which score a single sequence with respect to conformationally different backbones modelled as states [14–24]. Moreover, MSD allows for negative design, i.e., the computation of sequences that destabilize certain states related to misfolded conformations or an undesired binding interaction [25]. However, the more precise MSD approach has its price due to the demands for higher computational efforts needed for the identification of appropriate sequences: Considering m states requires scoring each candidate sequence m times and combining these scores to a global "fitness" value in order to identify sequences that are optimal for all states. MSD approaches have demonstrated their superiority in applications like the prediction of mutational tolerance in enzymes [26], the understanding of thermal adaptation of enzymes [27], the design of influenza antibodies [28], multi-specific interfaces [29], and multi-substrate enzymes [30].

A well-proven and highly flexible software suite supporting highly diverse problems of protein design is Rosetta [31] and several Rosetta-based MSD protocols have been implemented [14, 15, 28]. To determine the fitness of a sequence during the search phase, the 3D residue positions, whose occupancy can be varied by the software protocol, are decorated with the considered amino acid side chains. A key element of this assessment is to find an optimal combination of side chain orientations [32]. Building these optimized 3D ($3D_{opt}$) models consumes most of the computational time needed for the whole protein design protocol. If the occupancy of n residue positions is unconstrained for a design task, optimal rotamer combinations have to be found and assessed for 20^n different $3D_{opt}$ models in an SSD and for $m \times 20^n$ $3D_{opt}$ models in an MSD experiment. Thus, even for design problems of moderate complexity, a hybrid method that does not require the computation of a costly $3D_{opt}$ model to score each of the candidate sequences might drastically reduce computation time of Rosetta's protein design protocols.

This search for optimal sequences can be considered a problem of multi-dimensional regression, were every combination of amino acid residues yields one data point of the design-specific energy landscape. Due to the superior runtime, the usage of a simple regression model seems an attractive means for the scoring of these residue combinations. However, structure and function of proteins often depend on nonlinear and nonadditive relationships between the physical properties of residues [33]. Especially for protein engineering and design, it is highly recommended to consider these complex interactions [34], which strongly argues against the usage of simple regression models. In contrast, neural networks (NNs) have proven well-suited to solve complex classification and regression problems of computational biology [35, 36]. Thus, we explored, whether we can utilize an NN in a hybrid approach to rapidly sample candidate sequences during Rosetta protocols. More specifically, we wanted to test whether the existence of a moderate number of $3D_{opt}$ models is sufficient to teach an NN the energy landscape of a specific design problem. Thus, we implemented the multi-state framework `ROSETTA:MSF:NN` and used benchmark datasets to confirm that NNs can learn the design-specific energy landscapes of protein design. We found that `ROSETTA:MSF:NN` converges 10-times faster than our previous protocol and samples alternative areas of sequence space.

Materials and methods

Datasets used for the initial performance test

The dataset HisB_GA$_{raw}$ consisted of 48,588 tuples $des\_seq_j^{raw} = (aa_1^j, \ldots, aa_{14}^j, RS_{3DM}^j)$ that were generated by means of `Rosetta:MSF:GA:enzdes` during 500 iterations of a genetic algorithm (GA) for an ongoing design project. Based on one-hot encoding, each 20-dimensional vector aa_{pos}^j represented one amino acid residue at position *pos* of a candidate sequence *j*. This experiment was aimed at the redesign of the 14 residues constituting the ligand-binding site of the bifunctional enzyme HisB-N from *Escherichia coli*. This enzyme hydrolyses L-histidinol phosphate to L-histidinol and phosphate as well as O-phospho-L-serine to L-serine and phosphate. For this MSD approach, 11 states were used that represented slightly different conformations generated by means of a short (1 ns) molecular dynamic simulation seeded with the structure of the N-terminal domain of *E. coli* HisB (chain A of PDB-ID 2fpu). The sequences enumerated by the GA are highly similar, due to the preferential introduction of single point mutations.

In order to create a second, non-redundant dataset HisB_GA$_{nr}$ consisting of tuples $des\_seq_j^{nr} = (aa_1^j, \ldots, aa_{14}^j, RS_{3DM}^j)$, the maximal pairwise sequence identity was limited to 70%, which gave rise to 533 sequences. Selecting the sequences randomly, a training dataset consisting of 67% and a test dataset consisting of 33% of the sequences were created both for HisB_GA$_{raw}$ and HisB_GA$_{nr}$.

Benchmark dataset MD_EnzBench

The dataset MD_EnzBench has been generated previously for benchmarking ligand binding design based on `Rosetta:MSF` [3]. It has been deduced from molecular dynamics (MD) simulations of length 10 ns generated with YASARA (version 14.7.17) and the YAMBER3 force field that has been parameterized to produce crystal-structure-like protein coordinates [37]. MD_EnzBench consists of 16 proteins $prot_k$ with bound ligand taken from the scientific sequence recovery benchmark of Rosetta [38] and each $prot_k$ is indicated by the PDB-ID. To introduce conformational flexibility during the MD simulations, the ligand has been removed and for each of the 16 apoproteins, 1000 conformations have been sampled at an interval of 10 ps. As a structural basis for the subsequent MSD protocol, protein conformations have been saved every 1 ns and used as states. After sampling, the native ligands have been re-introduced in all conformations of the respective apoproteins by means of `PyMOL:superpose` [39]. The design and repack shells of all enzymes have been listed previously [2]. All design shell residues have been replaced with alanine and prior to design, all conformations have been energy-minimized by means of `Rosetta:fastrelax` with backbone constraints.

Design and implementation of the NN

The 4-layered architecture of the NN is shown in **Fig 1A**. The input layer consists of 20*n* neurons that are supplied with a one-hot encoding of amino acid residues aa_{pos}^j of the *n* design shell residues whose composition constituted the candidate sequence under study. The first hidden layer consists of 10+20*n*/2 neurons and the second hidden layer of 5+20*n*/4 neurons. The output layer consists of one neuron that computes the score RS_{NN} as a real value. Each layer is fully connected with the previous layer and no bias is used in any layer.

The NN was created using the python package *Keras* version 2.2.4 and *TensorFlow* 1.12.0 as backend. For initialization, a `RandomNormal` kernel was used in all layers. Both hidden layers utilize a `tanh` and the output layer a `linear` activation function. The network was optimized by means of a stochastic gradient descent with momentum. Prior to training, the scores RS_{3DM} computed by

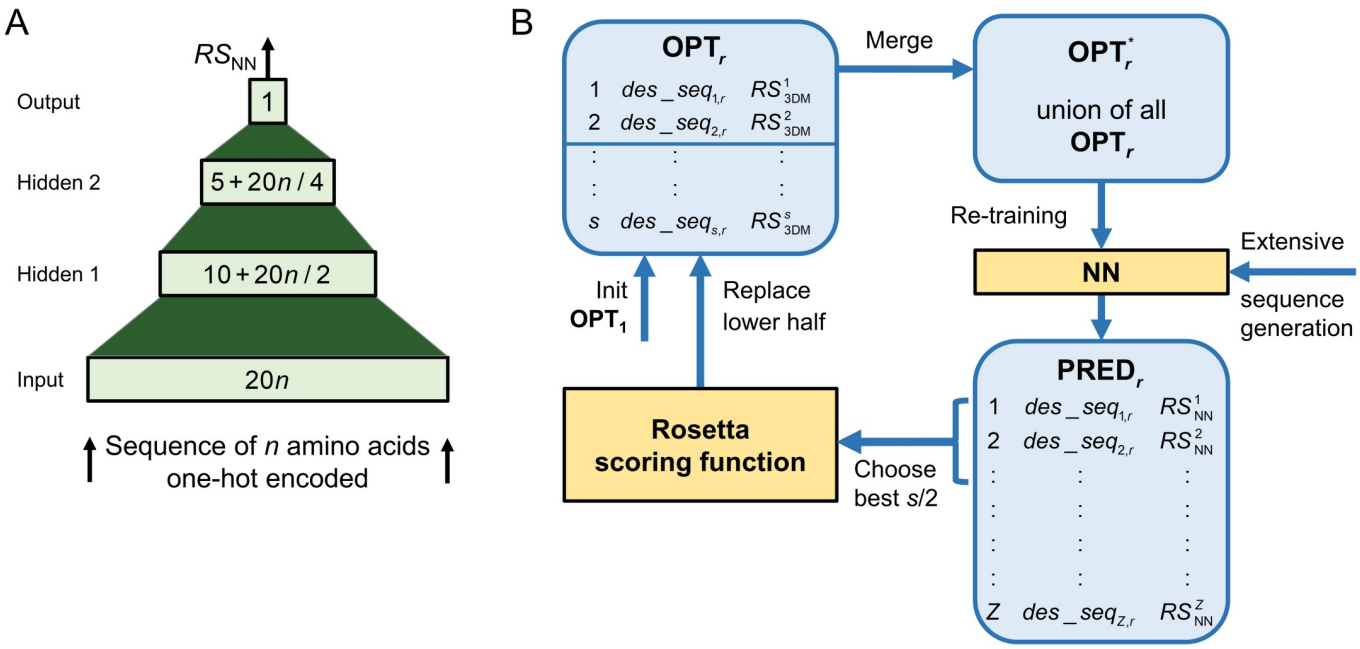

Fig 1. NN architecture and data generation within Rosetta:MSF:NN. (A) The input layer of the NN contains $20n$ neurons that process the 20-dimensional vectors representing the amino acids aa^j_{pos} of the $pos = 1,\ldots,n$ design shell residues under study. The output layer consists of one neuron that determines the RS_{NN} value. The first hidden layer consists of $10+20n/2$ neurons and the second hidden layer of $5+20n/4$ neurons. All neurons of adjacent layers are fully connected. (B) During all iterations r of sequence optimization, Rosetta:MSF:NN administers a pool OPT_r of s sequences, whose RS_{3DM} values are known. The union OPT^*_r of these sequences is growing iteration-wise and used to re-train the NN that has an architecture as in (A). The re-trained NN is utilized for an extensive sequence scan that computes per default the RS_{NN} values for $Z = 2{,}000{,}000$ randomly generated sequences based on a seed. For $s/2$ sequences possessing highest RS_{NN} values, the (costly) RS_{3DM} is computed. These sequences and their RS_{3DM} values constitute the lower half of the updated set OPT_{r+1}. For the first iteration, OPT_1 is initialized with s datasets.

https://doi.org/10.1371/journal.pone.0256691.g001

Rosetta for 3D models, were converted to z-scores. The model was trained for 100 epochs in incremental mode. Note that we used a lightweight NN that does not require the usage of GPUs.

Design and data flow of Rosetta:MSF:NN

Our previously introduced approach `Rosetta:MSF:GA` utilizes the same strategy as a recently published generic program [14]: In an outer loop, a GA is used to search sequence space and for each state, rotamers are optimized in an inner loop. However, `Rosetta:MSF:GA` does not operate on a population of 100, but on 239 sequences.

In order to allow for a fair comparison with `Rosetta:MSF:GA`, the novel NN-based approach `Rosetta:MSF:NN` administers for each iteration r a set OPT_r of $s = 239$ sequences, whose RS^j_{3DM} values are used for their ranking; see **Fig 1B**. These des_seq_j sequences represent the amino acid residues aa^j_{pos} chosen for the positions $pos = 1,\ldots,n$ of the design shell under study and for each des_seq_j, the RS^j_{3DM} value was computed by means of the chosen Rosetta scoring function. Initially, `Rosetta:MSF:NN` generates the set OPT_1 consisting of the given seed sequences and mutants, each with a randomly introduced single point mutation [15]. During each iteration r, OPT_r is added to OPT^*_r, which contains the shuffled union of all so far chosen des_seq_j data. The updated set OPT^*_r is used to re-train the NN, which is then utilized to assess an extensive set of Z novel sequences. The seeds for these novel sequences are the iteration-specific top-scoring sequences. Per default, $Z = 2{,}000{,}000$ random sequences are generated, which get mutated at a random number of 1 to n positions of design shell residues to randomly chosen amino acid residues and are fed into the NN to compute their score RS^j_{NN}. The dataset $PRED_r$ contains the Z

sequences $des\_seq_{j,r}$ ranked according to their RS^{j}_{NN} values. For the $s/2$ best scoring candidate sequences $cand_r$, the chosen Rosetta scoring function is used to compute their RS^{j}_{3DM} values. To prepare the set OPT_{r+1} utilized in the next iteration, `Rosetta:MSF:NN` replaces the bottom half of the OPT_r sequences with the $s/2$ candidates $cand_r$. These iterations continue until a user-defined stopping criterion is satisfied. As `Rosetta:MSF:NN` was utilized for an MSD protocol, a separate NN was trained for each individual state of each protein.

Assessing design performance

To determine the score of a candidate sequence des_seq_j for an MSD protocol, the mean Rosetta score was computed:

$$RS^{j}_{\mathrm{3DM}}(des\_seq_j) = \frac{1}{m}\sum_{i=1}^{m} ts_i(des\_seq_j) \tag{1}$$

Here, m is the number of states and $ts_i(des\_seq_j)$ is the Rosetta total score for a sequence given a state i. In all equations, Rosetta scores are indicated in REUs.

To assess the fitness of a sequence set OPT_r of an iteration r, the mean of all $s = 239$ $RS^{j}_{\mathrm{3DM}}(des\_seq_j)$ values was determined:

$$RS(\mathrm{OPT}_r) = \frac{1}{s}\sum_{j=1}^{s} RS^{j}_{\mathrm{3DM}}(des\_seq_{j,r}) \tag{2}$$

To distinguish the values related to the NN- and GA-based protocol, they were designated $RS_{\mathrm{NN}}(\mathrm{OPT}_r)$ and $RS_{\mathrm{GA}}(\mathrm{OPT}_r)$, respectively.

For the determination of the design-specific normalized areas above (NAA) the $RS_{\mathrm{NN}}(\mathrm{OPT}_r)$ and $RS_{\mathrm{GA}}(\mathrm{OPT}_r)$ values, the areas flanked by the specific $RS(\mathrm{OPT}_1)$ and $RS(\mathrm{OPT}_{100})$ values were calculated. For their normalization between 0.0 and 1.0, the lowest value reached after 100 generations by either of the two protocols was used.

Following trends of sequence sampling

To characterize trends of sequence sampling, for each design $prot_k$ of MD_EnzBench the composition of the sequence sets $\mathrm{OPT}^{\mathrm{NN}}_{k,r}$ ($r = 1$–100) generated by the NN-based protocol was compared with a well-defined reference sequence set $\mathrm{OPT}^{\mathrm{GA}}_{k,\delta^*}$.

To begin with, for each design $prot_k$ the reference set $\mathrm{OPT}^{\mathrm{GA}}_{k,\delta^*}$ was identified. This set represents among the first $\delta = 1$–500 GA iterations the earliest generation δ^*, whose score value was most similar to $RS_{\mathrm{NN}}(\mathrm{OPT}^{\mathrm{NN}}_{k,100})$, which was generated by `Rosetta:MSF:NN:enzdes` during the last iteration of the NN-based protocol. Utilizing the amino acid composition of the related 239 sequences $des\_seq_{j,r}$ belonging to $\mathrm{OPT}^{\mathrm{GA}}_{k,\delta^*}$, a normalized frequency table $ft^{\mathrm{GA}}_{k,pos} = f^{\mathrm{GA}}_{k,pos}(aa_i)(i = 1-20)$ was deduced for each of the n residue positions pos of the design shells. Analogously, frequency tables $ft^{\mathrm{NN}}_{k,pos,r}$ were derived from the 100 $\mathrm{OPT}^{\mathrm{NN}}_{k,r}$ sets created during all iterations of the NN-based protocol. Position-specific Euclidean distances $euk_r(ft^{\mathrm{NN}}_{k,pos,r}, ft^{\mathrm{GA}}_{k,pos}) \times (r = 1-100)$ were computed according to:

$$euk_r(ft^{\mathrm{NN}}_{k,pos,r}, ft^{\mathrm{GA}}_{k,pos}) = \sqrt{\sum_{i=1}^{20}\left(f^{\mathrm{NN}}_{k,pos,r}(aa_i) - f^{\mathrm{GA}}_{k,pos}(aa_i)\right)^2} \tag{3}$$

To assess the mean amino acid variation for each iteration r and each design k, the Euclidean distances were averaged according to:

$$dist_{k,r} = \frac{1}{n}\sum_{pos=1}^{n} euk_r(ft^{NN}_{k,pos,r}, ft^{GA}_{k,pos}) \quad (4)$$

$dist_{k,r}$ values were divided in two groups $NN^k_1 = \{dist_{k,r} | r = 1 - 50\}$ and $NN^k_2 = \{dist_{k,r} | r = 51 - 100\}$ to distinguish the composition of the sequences generated during the first and second half of the iterations.

To compute "null model" distributions R^k_1 and R^k_2 that served as references, the frequencies were shuffled table-wise for each of the $100 \times n$ $ft^{NN}_{k,pos,r}$ sets. Afterwards, Euclidean distances to $ft^{GA}_{k,pos}$ were computed according to Eq 3 and their mean was determined according to Eq 4. The four $prot_k$ specific distributions of $dist_{k,r}$ values were visualized by means of a box plot.

Results and discussion

A 4-layer NN is able to approximate the energy landscape of a design problem

Our basic assumption was that an NN can deduce the energy landscape of a design problem, if it is trained with a small number of sequences, whose design-specific Rosetta scores are known. In order to test this hypothesis, we had to choose a suitable representation of amino acid residues, the number of residues fed into the NN, and a network topology. We decided to represent each amino acid residue *aa* by means of one-hot encoding. The amino acid specific, 20-dimensional vectors are listed in **S1 Table**. We restricted the number of residue positions fed into the NN to those n ones that are subjected to the design process, i.e. belong to the design shell. To deduce a prediction of the Rosetta score from these $20n$ features, we opted for a 4-layered, fully connected, feed-forward network. We chose two hidden layers, because these feed-forward networks generalize better than those with only one hidden layer [40]. As we expected mutual dependencies in the occupancy of the structurally adjacent positions of design shells, we opted for a fully connected architecture, because it is capable of learning any function [41]. The representation of the candidate sequences proposed an input layer consisting of $20n$ neurons and an output layer consisting of one neuron that presents the predicted score as a real value. Based on a grid search for the number of hidden neurons (**S1 Fig**), we chose a first hidden layer with $10+20n/2$ neurons and a second hidden layer with $5+20n/4$ neurons; see **Fig 1A**.

For a first assessment of the predictive power of the NN, we utilized the outcome of a comprehensive `Rosetta:MSF:GA:enzdes` run of a HisB-N design, which was based on a genetic algorithm (GA). Briefly, a GA imitates the process of natural selection by maintaining a population of design sequences that are evolving for several generations. The MSD protocol `Rosetta:MSF:GA` generates candidate sequences by using the well-proven GA of Rosetta and computes their Rosetta total score averaged over all chosen states; for details see [15] and references therein. MSD is superior over SSD even for seemingly simpler design task: Compared to SSD, a positive MSD approach similar to the one used here has more clearly identified for a dataset of 15 design problems the sequences that are optimal for all states [23].

These HisB-N data are from an on-going design project, where we want to accommodate a new ligand in the binding site of the enzyme HisB-N by altering the occupancy of $n = 14$ residue positions. During 500 iterations of the GA, the Rosetta scores RS_{3DM} of 48,588 candidate sequences were computed, each based on a sequence-specific $3D_{opt}$ model. This set, which we

named HisB_GA$_{raw}$ consisted of tuples $des\_seq_j^{raw} = (aa_1^j, .., aa_{pos}^j, .., aa_{14}^j, RS_{3DM}^j)$, where each 20-dimensional vector aa_{pos}^j represents one amino acid at a position *pos* of a candidate sequence *j* and RS_{3DM}^j is the corresponding and normalized score (Eq 1) deduced by Rosetta from specific 3D$_{opt}$ models. We randomly selected 67% of the HisB_GA$_{raw}$ tuples and used them to train the NN; for details see Materials and methods. After training, we utilized this NN to determine predicted Rosetta scores RS_{NN}^j for the remaining 33% of the feature vectors $des\_seq_j^{raw} = (aa_1^j, \ldots, aa_{14}^j)$.

In Fig 2A, RS_{NN}^j values are plotted versus the corresponding RS_{3DM}^j values. The clearly visible correspondence of both scores is evidenced by the high value of the Pearson correlation coefficient (PCC), which was 0.95 ($p << 1E\text{-}100$) for the set of all test data. The average error determined for the test dataset was not larger than 0.96 Rosetta Energy Units (REU). This test set was generated by the GA and of the 16,196 sequences, not more than 110 contain more than two mutations. Thus, this dataset did not allow us to study whether the NN is able to correctly score more strongly deviating sequences.

In order to assess the predictive performance of the NN for more difficult cases, we utilized HisB_GA$_{raw}$ to deduce the non-redundant dataset HisB_GA$_{nr}$. By accepting a maximal pairwise sequence identity of 70%, 533 tuples were identified that differed at least in four residue occupancies. The set HisB_GA$_{nr}$ was used to train and test the NN as described above. As Fig 2B shows, the NN performed well also for the more difficult cases: The PCC was 0.97 ($p < 1E\text{-}100$) and the average error was 1.8 REU.

To generate the plot shown in S2B Fig, we exclusively utilized the 238 HisB_GA$_{raw}$ tuples generated during the first iteration of the genetic algorithm. For the 80 test cases, the PCC dropped to 0.62 ($p = 1.2E\text{-}9$) and the average error was 2.9 REU. These values indicate that one

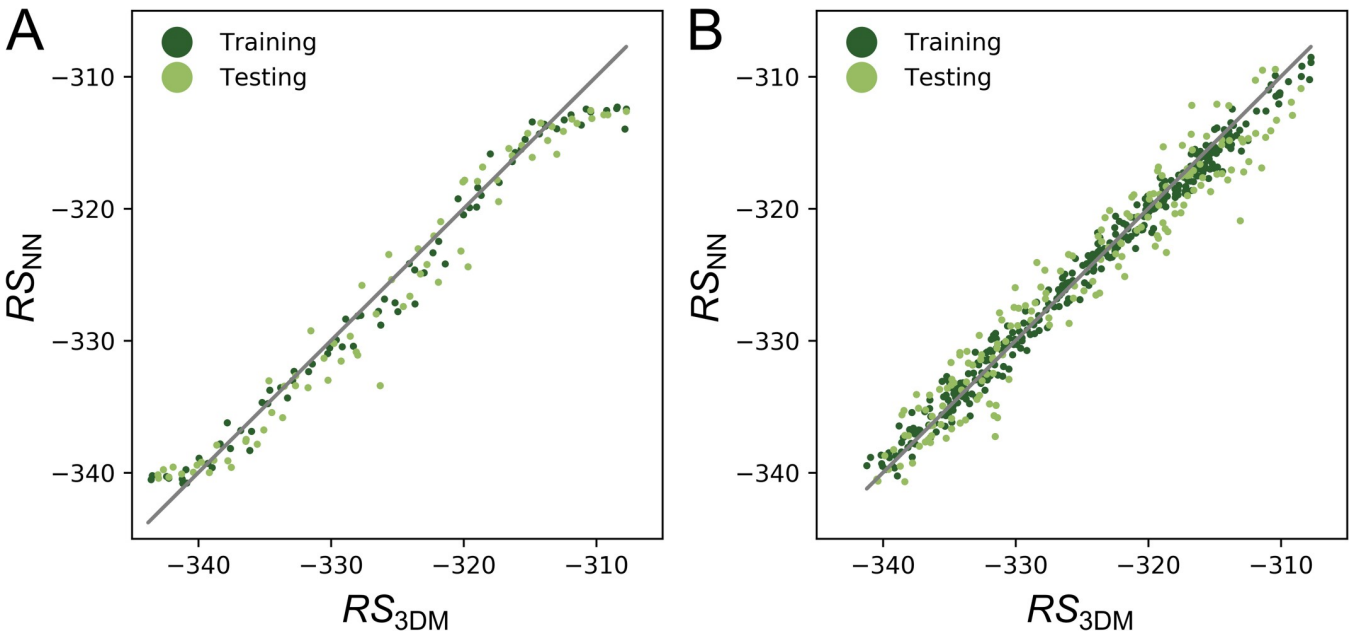

Fig 2. Performance of NNs for sequence sets with different sequence similarities. In both panels, Rosetta scores RS_{3DM}^j deduced from 3D$_{opt}$ models are plotted versus the Rosetta scores RS_{NN}^j predicted by NNs for an enzyme design. Scores are given in REUs and training and test data are represented as dark and light green dots, respectively. The gray lines indicate the diagonal, i.e. the position of perfect predictions. (A) Performance resulting for the dataset HisB_GA$_{raw}$ that consists mainly of sequences with point mutations. The PCC deduced from RS_{3DM}^j and RS_{NN}^j values of the test data was 0.95. For reasons of clarity, a large number of overlapping dots were removed; the plot shown in S2A Fig contains the full dataset. (B) Performance resulting from the redundancy-free dataset HisB_GA$_{nr}$ that consists of 533 sequences with a maximal pairwise sequence identity of 70%. The PCC was 0.97.

https://doi.org/10.1371/journal.pone.0256691.g002

iteration of the GA is not sufficient to sample the complex energy landscape of this design problem in an adequate manner.

Taken together, we concluded that the representation of amino acid side chains with one-hot encoding and a trained NN are appropriate to model that part of the energy landscape (Rosetta scores) sampled by a GA during the problem-specific `enzdes` calculation.

Rosetta:MSF:NN, a hybrid framework for MSD

The training with the HisB_GA$_{raw}$ tuples generated during the first GA iteration was too sparse for the complex energy landscape to be learnt by the NN. S2B Fig suggests that the RS^j_{3DM} values of several rounds of sequence optimization are required for an adequate representation of this energy landscape. However, one cannot predefine the number of iterations needed to find optimal sequences for a given design problem, as convergence speed is problem-specific. Guided by this constraint, we designed a novel protocol for sequence search; compare Fig 1B. We decided to replace the genetic algorithm of `Rosetta:MSF:GA` with an NN-based sequence selection, but to continue keeping a specific set $OPT_r = \{des\_seq_{j,r} | 1 \leq j \leq s\}$ of s optimal sequences for each iteration r. `Rosetta:MSF:NN` computes the RS^j_{3DM} score based on sequence-specific $3D_{opt}$ models only for newly generated elements of OPT_r. The union of all OPT_t tuples determined during the preceding iterations $t = 1, \ldots, r\text{-}1$ constitutes the continuously growing training set OPT^*_r, which is used to re-train the NN for iteration r.

The re-trained NN is then utilized to predict the scores RS^j_{NN} of an extensive number Z of randomly generated sequences; the default value is $Z = 2{,}000{,}000$. For those $s/2$ sequences with highest RS^j_{NN} values, `Rosetta:MSF:NN` computes the RS^j_{3DM} values based on sequence-specific $3D_{opt}$ models; for details see Materials and methods. To compile the next set OPT_{r+1}, half of the $des\_seq_{j,r}$ tuples are replaced with newly generated ones and their corresponding RS^j_{3DM} values. Analogously to `Rosetta:MSF:GA`, the user has to execute the novel algorithm for several iterations until convergence is reached. This hybrid algorithm of sequence selection combines three major advantages:

1. Due to the high speed of the NN in computing RS^j_{NN} values, an extensive number of candidate sequences can be assessed, and the RS^j_{NN} values approximate the RS^j_{3DM} values quite well. The determination of RS^j_{3DM} values for all possible candidate sequences is currently not feasible due to the computational costs of the $3D_{opt}$ models.
2. By merging the iteratively generated sets OPT_r that consist of all hitherto found well-scoring sequences, the prediction quality of the NN increases continuously due to the denser sampling of the problem-specific energy landscape.
3. The iteration-specific determination of RS^j_{3DM} values for the sequences with highest RS^j_{NN} values provides a corrective feedback and helps to eliminate less optimal NN predictions.

Encouraged by these promising initial results, we wanted to answer the following four questions in order to assess the potential benefit of integrating an NN into `Rosetta:MSF`:

1. Does the use of NNs reduce the number of iterations needed to identify optimal candidate sequences?
2. How robust is this approach with respect to the chosen features and scoring functions?
3. Does the NN-based approach find sequences with better Rosetta scores?
4. Does the extensive sampling of sequence space lead to candidate sequences not found by the GA?

`Rosetta:MSF:NN` converges 10-times faster than Rosetta:MSF:GA and enumerates better scoring sequences

For a comprehensive comparison of `Rosetta:MSF:NN` with the older `Rosetta:MSF:GA` protocol, we utilized the previously introduced benchmark MD_EnzBench [15]. This set has been compiled to test the ability of protocols to rebuild the ligand-binding site of 16 proteins $prot_k$. For each $prot_k$, 10 specifically prepared conformations that served as states of an MSD protocol have been deduced by means of molecular dynamics simulations. Each conformation contains the bound ligand and in order to increase the difficulty of the design task, all residues of the design shells were replaced with alanines; for details see Materials and methods.

For these 16 design problems, `Rosetta:MSF:GA:enzdes` and `Rosetta:MSF:NN:enzdes` were executed as 10-state MSD protocols for at least 100 iterations and the NNs were re-trained during each iteration as described. In order to follow the convergence of the two design processes, the mean Rosetta score $RS(\mathrm{OPT}_r)$ (Eq 2) was determined for each iteration r. $RS(\mathrm{OPT}_r)$ is the mean of the $RS^j_{3\mathrm{DM}}$ scores of all sequences related to the iteration-specific set OPT_r. In **Fig 3**, the protocol-specific convergence is shown for four representative examples *repr_prot* = {2dri,2ifb,lopb,2rct}, which are indicated by their PDB-ID, and all 16 designs are documented in **S3 Fig**. As can be seen, both protocols find sequences that score considerably better than the native ones. Most interestingly, the $RS_{\mathrm{NN}}(\mathrm{OPT}_r)$ values dropped more rapidly than the $RS_{\mathrm{GA}}(\mathrm{OPT}_r)$ values. In order to assess the momentum of convergence numerically, we determined the normalized area above (NAA) the $RS_{\mathrm{NN}}(\mathrm{OPT}_r)$ and the $RS_{\mathrm{GA}}(\mathrm{OPT}_r)$ values

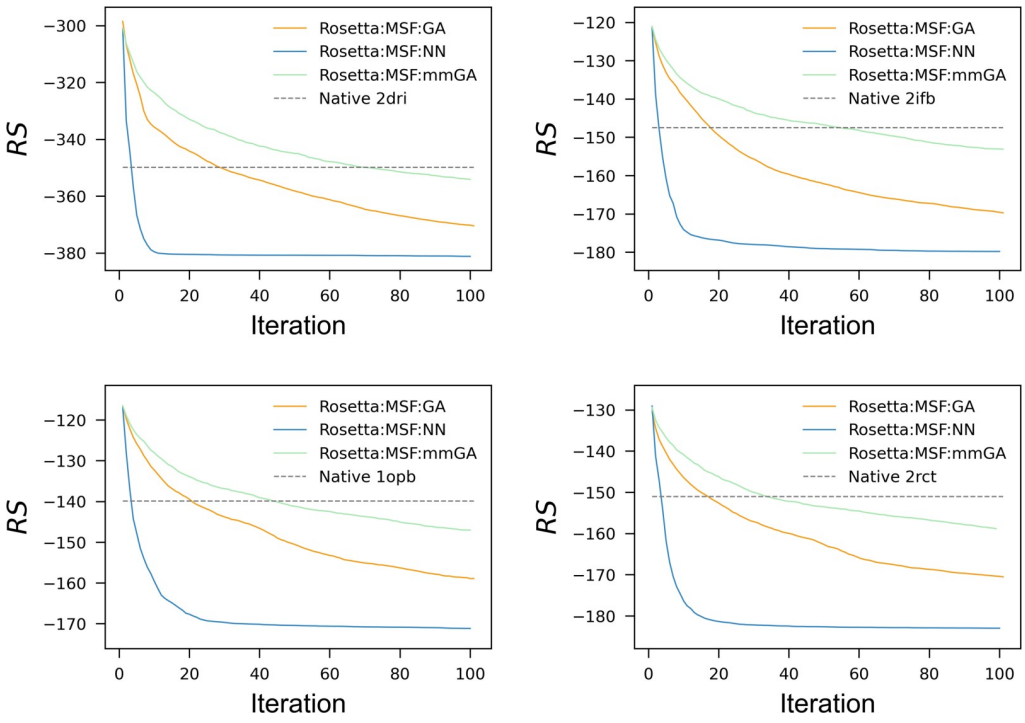

Fig 3. Convergence of `Rosetta:MSF:GA:enzdes`, `Rosetta:MSF:mmGA:enzdes`, and of `Rosetta:MSF:NN:enzdes`. The three protocols were utilized for each of the four designs from *repr_prot*. The mean Rosetta scores $RS(\mathrm{OPT}_r)$ were determined in REUs for each iteration r = 1–100 and plotted. The blue lines represent the $RS_{\mathrm{NN}}(\mathrm{OPT}_r)$ values, the orange lines the $RS_{\mathrm{GA}}(\mathrm{OPT}_r)$ and the green line the $RS_{\mathrm{mmGA}}(\mathrm{OPT}_r)$ values. The dashed horizontal line marks the score of the relaxed native protein; the PDB-ID of the corresponding native protein is indicated. The `Rosetta:MSF:GA:enzdes` and `Rosetta:MSF:NN:enzdes` plots for all 16 designs are shown in S3 Fig. `mmGA` is a genetic algorithm that introduces multiple mutations.

https://doi.org/10.1371/journal.pone.0256691.g003

Table 1. Performance comparison of the GA- and NN-based protocols.

PDB-ID	*n*	*ss*	NAA_{GA}	NAA_{NN}	r_k^*	$r_{k,GA}^{native}$	$r_{k,NN}^{native}$
1fzq	20	29	0.59	0.96	8	32	4
1hsl	19	42	0.73	0.96	9	16	3
1j6z	27	45	0.75	0.95	9	31	5
1n4h	25	51	0.62	0.95	7	19	3
1nq7	28	57	0.62	0.95	9	24	4
1opb	22	50	0.57	0.93	10	21	4
1pot	19	41	0.67	0.94	7	4	2
1urg	19	40	0.56	0.96	7	24	3
2b3b	17	46	0.68	0.97	6	25	3
2dri	19	42	0.67	0.97	6	29	4
2ifb	22	54	0.63	0.95	8	18	3
2q2y	23	34	0.67	0.96	8	47	5
2qo4	22	40	0.58	0.96	7	25	4
2rct	22	51	0.57	0.94	7	17	4
2rde	20	35	0.61	0.93	7	19	4
2uyi	23	35	0.61	0.96	7	61	5
Average	**22**	**43**	**0.63**	**0.95**	**7.6**	**25.8**	**3.8**

All values were determined for each of the $k = 16$ proteins (indicated by their PDB-ID) from the MD_EnzBench set. *n* is the number of design shell residues and *ss* the number of second shell residues, respectively. The NAA_{GA} and NAA_{NN} values specify the area above the plots of the corresponding *RS* values for the GA- and NN-based protocol. r_k^* is the number of the first NN iteration, whose $RS_{NN}(OPT_r)$ value reached the $RS_{GA}(OPT_{100})$ value that served as a reference. The $r_{k,GA}^{native}$ and $r_{k,NN}^{native}$ values indicate for the GA- and the NN-based protocols the number of the first iteration that generated a sequence set having native-like Rosetta energies.

https://doi.org/10.1371/journal.pone.0256691.t001

in analogy to a ROC curve [42]. An NAA value gets close to 1.0, if the curve drops vertically to its minimum during the first few iterations. In **Table 1**, the NAA_{NN} and NAA_{GA} values are listed for all 16 designs as well as the numbers *n* and *ss* of design shell residues and second shell residues; for their definition see [8]. The comparison of the design-specific NAA_{NN} and NAA_{GA} values confirmed that Rosetta:MSF:NN:enzdes converged much faster than the GA-based protocol: The mean $RS_{NN}(OPT_r)$ value was 0.95, whereas the mean $RS_{GA}(OPT_r)$ value was 0.63. Although trained with relatively few examples during the first iterations, Rosetta:MSF:NN:enzdes found more rapidly sequences with low Rosetta scores than the GA-based protocol did.

To estimate the performance gain, we determined for each design *k* the first iteration r_k^*, whose $RS_{NN}(OPT_r)$ value reached the $RS_{GA}(OPT_{100})$ value of iteration 100 of `Rosetta:MSF:GA:enzdes`. As **Table 1** shows, the r_k^* values varied between 6 and 10; this spread reflects most likely the differing complexity of the design-specific energy landscapes. On average, 7.6 iterations of the NN-based protocol were sufficient to find sequences that scored as good as those created by the GA-based protocol in iteration 100. We concluded that `Rosetta:MSF:NN:enzdes` converges approximately ten times faster than the GA-based protocol. A sevenfold gain results, if one compares for both approaches the first iterations $r_{k,GA}^{native}$ and $r_{k,NN}^{native}$ that generated a sequence set OPT_r having native-like Rosetta energies. On average, the GA needed 25.8, and the NN approach not more than 3.8 iterations; compare **Table 1**. Interestingly, convergence speed seems uncorrelated with the number of the design shell residues; of the three Spearman rank order correlation tests of *n* with NAA_{NN}, NAA_{GA}, and r_k^*, none gave a statistically significant result.

The comparison of the NAA_{NN} and NAA_{GA} values and the plots shown in S3 Fig indicate that `Rosetta:MSF:NN:enzdes` finds for a given number of iterations better-scoring sequences than the GA-based protocol. If both protocols were executed for 100 iterations, the $RS_{NN}(OPT_{100})$ value was superior to the $RS_{GA}(OPT_{100})$ value in all designs; compare S3 Fig. These findings confirm that the NN can beneficially exploit the extensive sampling to identify sequences with superior Rosetta scores.

The notably reduced number of iterations that are sufficient for `Rosetta:MSF:NN:enzdes` to reach a minimum, leads to time savings, if one condition is met: The protocol is faster, if the extra time required to train and use the NN has no drastic effect on the computing time needed to create a new generation of sequences. In order to estimate runtimes, we compared the performance of the GA- and the NN-based approach for the HisB design. For a fair comparison, execution times were determined for designs running on one CPU.

Most expensive was the determination of the RS^j_{3DM} scores and `Rosetta:MSF:GA:enzdes` required constantly 32 h for each iteration. In contrast, the execution time of `Rosetta:MSF:NN:enzdes` increases linearly with the size of the training set, which grows due to the union of the OPT_r sets. During the first iteration, two minutes were required for training and approximately 15 minutes for the determination of the new candidate sequences. For generation r, execution time $t(r)$ accumulated to $t(r) = r \cdot (32\text{h} + 10\text{min}) + \sum_{s=1}^{r}(s \cdot 2\text{min})$.

This minor increase of the iteration-specific execution time is more than compensated by the reduced number of iterations: As Table 1 shows, `Rosetta:MSF:NN:enzdes` needs on average only eight iterations to find sequences having scores comparable to those from generation 100 of `Rosetta:MSF:GA:enzdes`. Thus, for a design task whose complexity is comparable to the HisB case, the GA consumes approximately 3200 h to finish 100 iterations. In contrast, the NN-based approach needs not more than 8 iterations, which are finished after 256 h + 80 min + 72 min, i.e. after less than 260 h. This approximation makes clear that the faster convergence of `Rosetta:MSF:NN:enzdes` has a drastic effect on runtime due to the relatively minor expense added by the NN.

In principle, the NN-based protocol is applicable to all design algorithms that fulfill two prerequisites: i) The algorithm can provide tuples $des\_seq_j = (aa^j_1, .., aa^j_{pos}, .., aa^j_n, 3D - score^j)$ each consisting of a sequence and the corresponding $3D\text{–}score^j$. ii) The algorithm can process the NN output, which is a list of newly generated des_seq_j candidates. The NN-based protocol is most useful, if the calculation of approximative $3D - score^j_{approx}$ values reduces execution time, which has to be determined for each application. For example, protein design by means of EvoEF2 [43] or FoldX [44] is relatively fast: EvoEF2 is capable of designing de novo proteins consisting of 200 residues in 15 min and FoldX is at most five times slower [43]. Thus, the additional expense for training and using an NN does not seem justified for these applications.

Rosetta:MSF:NN performs well with different representations of amino acid residues, scoring functions, and design protocols

In order to assess the robustness and the applicability of `Rosetta:MSF:NN`, we performed additional redesign experiments of the four proteins from the set *repr_prot*. First, we wanted to make plausible that the performance of `Rosetta:MSF:NN` does not critically depend on the chosen one-hot encoding of residues shown in S1 Table. For the experiment named `NN_FT`, residues were not represented by one-hot encoding, but by vectors indicating the five features "volume", "polarity", "isoelectric point", "hydrophobicity", and "mean solvent accessibility" [45]; compare S2 Table. The analysis of the plots shown in S4 Fig reveals that this

alternative residue representation does not reach the performance of one-hot encoding, but performs better than the GA: In all cases, the NN-based algorithm converged faster than the GA-based one.

For all designs presented so far, we utilized the REF15 scoring function [46]. In order to make plausible that the convergence of `Rosetta:MSF:NN` is not dependent on a specific scoring system, another four redesign experiments were performed for the proteins from *repr_prot* by using the alternative *talaris* [47] scoring function: A clear convergence gain of the NN- over the GA-based protocol is confirmed by **S5 Fig** for the *talaris* scoring function as well.

The `FastDesign` (FD) protocol iterates between sequence optimization and structure refinement and has been used to design novel protein folds and assemblies [48], but also for enzyme design [49]. For an initial assessment of the NN's ability to find proper sequences in the FD-specific energy landscape, we applied the settings "`FastDesign` (with ligand)" introduced recently [49] to the four proteins from *repr_prot*. For a comprehensive sampling of the energy landscape and an initial training of the NN, we used `FastDesign` to enumerate for each protein 1000 sequences $\mathrm{FD}_{1000} = des\_seq^{\mathrm{FD}}_{j=1-1000} = (aa^j_1, .., aa^j_{pos}, .., aa^j_n, RS^j_{\mathrm{FD}})$, which is more than twice of the designs used elsewhere [49]. As **Table 2** indicates, the extensive sampling has no drastic effect on the scores: The mean of all 1000 RS^j_{FD}-values and of the 250 best ones ($\mathrm{FD}_{\mathrm{best250}}$) differed at most by 6.5 REU. As already observed with the `enzdes` protocol, the native sequences were scored worse than the designs. The $RS^{\mathrm{native}}_{\mathrm{FD}}$ values of 2dri, 2ifb, 1opb, and 2rct were -894.4, -391.1, -419.5, and -434.8, respectively.

For each protein, the 1000 sequences and their RS^j_{FD}-values were used to train an NN. The trained NNs were utilized to score $Z = 2{,}000{,}000$ randomly generated sequences and in analogy to our other comparisons, those 250 ones possessing the highest RS^j_{NN}-values were selected. To assess the quality of the NN predictions, these sequences were scored by means of `FastDesign` and the mean values were determined. As **Table 2** indicates, the $\mathrm{NN}(\mathrm{FD}_{1000})$ approach, i.e. a singular training of an NN with 1000 optimal cases performed poorly: In all four cases, the mean values were more than 25 REU worse than the mean $\mathrm{FD}_{\mathrm{best250}}$ values.

Most plausibly, the poor performance is (also in this case) due to the lack of unsuitable sequences in the training set. As a compensation, we applied the following protocol that iterates between `FastDesign` steps and NN-based sequence predictions: Initially, `FastDesign`

Table 2. Performance of FastDesign and a protocol based on four iterations of NN training.

	2dri		2ifb		1opb		2rct	
	RS_{FD}	$dist^{\mathrm{FD}}_r$	RS_{FD}	$dist^{\mathrm{FD}}_r$	RS_{FD}	$dist^{\mathrm{FD}}_r$	RS_{FD}	$dist^{\mathrm{FD}}_r$
FD_{1000}	-929.3	-	-407.6	-	-438.0	-	-459.0	-
$\mathrm{FD}_{\mathrm{best250}}$	-934.0	-	-412.7	-	-442.7	-	-465.3	-
$\mathrm{NN}(\mathrm{FD}_{1000})$	-880.0	0.50	-386.0	0.49	-391.1	0.46	-435.4	0.51
$\mathrm{OPT}^{\mathrm{FD}}_1$	-929.8	0.03	-407.7	0.04	-438.0	0.04	-459.1	0.05
$\mathrm{FD}_{\mathrm{NN\_1}}$	-889.4	0.51	-370.4	0.43	-404.1	0.48	-413.7	0.52
$\mathrm{FD}_{\mathrm{NN\_2}}$	-897.8	0.38	-382.4	0.37	-385.8	0.55	-432.6	0.34
$\mathrm{FD}_{\mathrm{NN\_3}}$	-924.8	0.38	-400.0	0.43	-437.5	0.38	-448.5	0.39
$\mathrm{FD}_{\mathrm{NN\_4}}$	-933.2	0.50	-410.8	0.56	-442.2	0.50	-457.7	0.41

The RS_{FD} columns list mean `FastDesign` scores and the $dist^{\mathrm{FD}}_r$ columns frequency distances for designs with the four proteins from *repr_prot*. Rows FD_{1000} and $\mathrm{FD}_{\mathrm{best250}}$ represent the values related to 1000 and the best 250 `FastDesign` sequences. The $\mathrm{NN}(\mathrm{FD}_{1000})$ values resulted from an NN training with the FD_{1000} sequences. The $\mathrm{OPT}^{\mathrm{FD}}_1$ values represent the initial training set of an iterative training that generated the output $\mathrm{FD}_{\mathrm{NN\_1}}$ to $\mathrm{FD}_{\mathrm{NN\_4}}$.

https://doi.org/10.1371/journal.pone.0256691.t002

was used to generate 250 sequences, which served in combination with their `FastDesign` scores RS^j_{FD} as training cases $\mathrm{OPT}^{\mathrm{FD}}_1 = des\_seq^{\mathrm{FD}}_{j=1-250} = (aa^j_1, .., aa^j_{pos}, .., aa^j_n, RS^j_{\mathrm{FD}})$. After training, the NN was used to generate Z = 2,000,000 sequences and those 250 ones with highest RS^j_{NN}-values were scored by means of `FastDesign` and stored as the result $\mathrm{FD}_{\mathrm{NN\_1}}$. The union of this sequence set and of $\mathrm{OPT}^{\mathrm{FD}}_1$ gave the training set $\mathrm{OPT}^{\mathrm{FD}}_2$. During the second iteration, the NN was trained with these 500 sequences and used to create 250 sequences constituting $\mathrm{FD}_{\mathrm{NN\_2}}$. For these sequences, their RS^j_{FD} score was determined and they were merged with $\mathrm{OPT}^{\mathrm{FD}}_2$ to the training set $\mathrm{OPT}^{\mathrm{FD}}_3$. During a third and a fourth iteration, the NN was trained with 750 $\mathrm{OPT}^{\mathrm{FD}}_3$ and 1000 $\mathrm{OPT}^{\mathrm{FD}}_4$ sequences, and the two sets $\mathrm{FD}_{\mathrm{NN\_3}}$ and $\mathrm{FD}_{\mathrm{NN\_4}}$ were generated, each consisting of 250 sequences and their RS^j_{FD} scores.

The iteration-specific mean RS_{FD} values shown in **Table 2** confirm that this approach performed much better than the single-iteration training with 1000 sequences: For three proteins, the mean $\mathrm{FD}_{\mathrm{NN\_4}}$ value differed not more than 2 REU from the $\mathrm{FD}_{\mathrm{best250}}$ value and the maximal difference was 7.6 REU. These findings suggest for most cases that the corrective feedback generated during a minimum of four training iterations is sufficient for an NN to learn the `FastDesign` specific energy landscape. Iterative training is more effective, as the newly computed RS_{FD} values provide valuable feedback and allow the NN to avoid false positive predictions.

In summary, these results emphasize the capability of the NNs to learn a problem-specific energy landscape modelled by means of different residue representations, scoring functions, and protocols.

The extensive sampling of the sequence space finds alternative minima

With default values, our NN-based approach allows per iteration the assessment of 2,000,000 alternative sequences; in contrast, the GA-based approach generates less than 120 novel candidates per generation. We were interested to find out, whether this rigorous widening of search space sampling had a pronounced effect on the composition of the `enzdes` outcome. It is difficult to compare precisely the composition of two sequence sets, thus we opted for an approximate approach, namely the comparison of amino acid frequency tables.

To characterize trends, we wanted to compare for each design $prot_k$ of MD_EnzBench the composition of the sequence sets $\mathrm{OPT}^{\mathrm{NN}}_{k,r}$ (r = 1–100) generated by the NN-based protocol with a fixed reference sequence set $\mathrm{OPT}^{\mathrm{GA}}_{k,\delta^*}$ consisting of sequences generated by a late iteration of the GA-based protocol. If both protocols sample the same area of sequence space, the amino acid composition of the NN-based outcome should iteratively approach the amino acid frequencies of the GA-based reference.

To begin with, we determined for each design $prot_k$ of MD_EnzBench the set $\mathrm{OPT}^{\mathrm{GA}}_{k,\delta^*}$ among δ = 1,. . .,500 GA iterations. It belonged to the first generation δ^* whose sequence set $\mathrm{OPT}^{\mathrm{GA}}_{k,\delta}$ had a score comparable to that of the $\mathrm{OPT}^{\mathrm{NN}}_{k,100}$ set. In order to estimate compositional differences, we deduced for each of the n design shell residue positions pos a reference frequency table $ft^{\mathrm{GA}}_{k,pos} = f^{\mathrm{GA}}_{k,pos}(aa_i)(i = 1 - 20)$ from $\mathrm{OPT}^{\mathrm{GA}}_{k,\delta^*}$ and further frequency tables $ft^{\mathrm{NN}}_{k,pos,r}$ from all of the 100 $\mathrm{OPT}^{\mathrm{NN}}_{k,r}$ sets. Euclidean distances $euk_r(ft^{\mathrm{NN}}_{k,pos,r}, ft^{\mathrm{GA}}_{k,pos})(r = 1 - 100)$ (Eq 3) were computed and their mean $dist_{k,r}$ determined according to Eq 4. In order to assess the progression of compositional differences, these were divided into two groups $\mathrm{NN}^k_1 = \{\mathrm{dist}_{k,r} | r = 1 - 50\}$ and $\mathrm{NN}^k_2 = \{\mathrm{dist}_{k,r} | r = 51 - 100\}$ consisting of the outcome of the first and second half of the iterations. In order to generate "null model" distributions R^k_1 and R^k_2, the $ft^{\mathrm{NN}}_{k,r}$ frequencies were shuffled prior to the computation of their Euclidean distances to ft^{GA}_k.

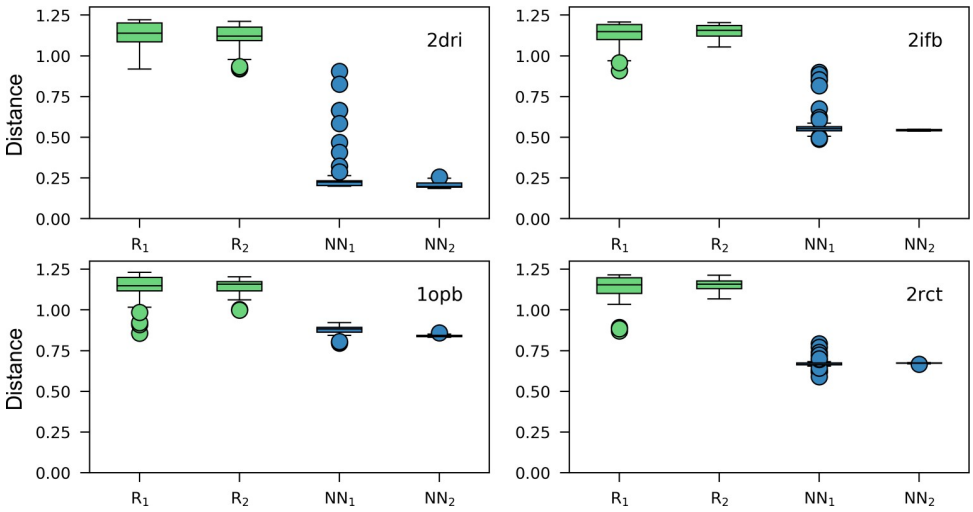

Fig 4. Amino acid frequency distributions for the outcome of the first and second half of design protocols. The boxplots represent the distributions of mean distances between amino acid frequencies tables $ft_{k,pos,r}$ related to the iterations of the NN-based protocol and a reference table $ft^{GA}_{k,pos}$ from the GA-based protocol; compare Eq 3 and Eq 4. The group NN^k_1 contains the distributions related to the first 50 and NN^k_2 the distributions of the second 50 iterations. For the computation of R^k_1 and R^k_2, the values of each table $ft^{NN}_{k,pos,r}$ were shuffled; for details see Materials and methods. The boxplots show the results for the four designs with the set *repr_prot*.

https://doi.org/10.1371/journal.pone.0256691.g004

For each design $prot_k$ these four distributions of distances were visualized by means of a box plot. Fig 4 shows that in all four cases from *repr_prot*, the mean of the distances sampled with the two null models R^k_1 and R^k_2 was close to the maximally possible distance, which is $\sqrt{2} \approx 1.4$. All NN^k_1 and NN^k_2 distributions had lower, but substantial distances to ft^{GA}_k. The sequences found for 2dri by the GA- and NN-based protocols resemble each other to a certain extent (mean 0.2), but those found for the other three designs are more dissimilar (mean > 0.56). This finding suggests that the GA- and the NN-based protocols concentrate on different regions of sequence space. In all four cases, the NN^k_1 spread (first half of iterations) of the distances was larger than the NN^k_2 spread, which indicates the convergence to a minimum. The remaining 12 designs have similar variations of their box plots patterns; compare S6 Fig: The smaller NN^k_2 values indicate the focusing on a certain region of sequence space, which differs in all cases from the one chosen by the GA: For 8 of the 16 designs, the mean of the NN^k_2 distances was > 0.4. In summary, we concluded that the GA and the NN protocols find different minima of sequence space in 8 of the 16 designs.

We expected that the NN finds also alternative minima of the `FastDesign` energy landscape. Thus, we determined for each of the four proteins from *repr_prot* mean $dist^{FD}_r$ values by comparing FD_{1000}, i.e., the 1000 sequences generated by `FastDesign`, with the four sequence sets FD_{NN_r} each comprising 250 sequences generated by the NN after training with 250, 500, 750, or 1000 sequences scored by means of `FastDesign`. The values were added to Table 2, are comparable to the differences between the GA and the NN protocols and propose the usage of an NN to broaden the sampling of the `FastDesign` energy landscape.

Epistatic effects may cause alternative minima

The region of sequence space optimal for a given protein backbone seems relatively limited to the neighborhood of the native sequence [50]. Although scoring better than native ones, the

Table 3. Comparison of sequence heterogeneity in candidate sequences added to OPT_r during $r = 100$ iterations.

Mutations	2dri		2ifb		1opb		2rct	
	GA	NN	GA	NN	GA	NN	GA	NN
1	11551	10606	11551	9562	11551	9019	11551	9226
2		708		1527		1738		1456
3		123		259		297		327
4		71		145		98		134
5		38		65		84		99
6		44		37		73		110
7		32		29		99		109
8		31		20		126		62
9		28		19		57		40
10		11		19		28		24
11		14		6		17		31
12		3		7		22		16
13		2		5		10		23
14		4		3		17		26
15		1		1		21		14
16				3		11		12
17				2		11		8
18						3		2

The table lists numbers of candidate sequences $cand^{GA}_{r=100}$ and $cand^{NN}_{r=100}$ grouped according to their differences (number of mutations) to the most similar sequence from $OPT_{r\,=\,100}$. For this analysis, $r = 100$ iterations of the designs for the four proteins from *repr_prot* were analyzed.

https://doi.org/10.1371/journal.pone.0256691.t003

sequences generated by the GA and the NN for the design shells were markedly different, which prompted us to find an explanation.

One reason could be that the width of sequence sampling differs markedly between the GA and the NN protocol. In order to compare the sampled sequences space, we determined sequence variability of the $s/2$ sequences ($cand^{GA}_r$, $cand^{NN}_r$; see Materials and methods) that were generated by the GA or the NN and used to replace the bottom half of OPT_r during each iteration r. Each sequence from a $cand_r$ set was compared to the most similar one from OPT_r to determine the number of newly introduced mutations. Sequence variability deduced from 100 iterations is shown in **Table 3** and that of the first 10 iterations in **Table 4**, because the initial iterations are crucial for NN training. Both distributions demonstrate that the GA generated the $cand^{GA}_r$ sequences mainly by introducing single point mutations, whereas the NN introduced up to 18 mutations to generate a $cand^{NN}_r$ sequence. These findings indicate that the NN approach samples sequence space much broader than the GA and generalizes well, because these highly mutated sequences reach high RS^j_{3DM} values. Moreover, these results make clear that the NN is trained with a broad sequence set, because the RS^j_{3DM} values of these novel and diverse $cand^{NN}_r$ sequences are part of the training set utilized by the next iteration r+1.

It is known that mutual dependencies in the occupancy of residue positions drastically affect the fitness landscape as has been confirmed *in silico* [51] and experimentally [52, 53]. As a consequence, high-order epistasis constrains the adaptive pathways that can be followed by evolution [54], because each given occupancy of residue positions severely restrains the subsequently tolerated mutations.

The design shells used here consist of a small number of highly constrained residue positions and the above findings strongly suggest that their occupancy is mutually dependent.

Table 4. Comparison of sequence heterogeneity in candidate sequences added to OPT_r during the first 10 iterations.

Mutations	2dri		2ifb		1opb		2rct	
	GA	NN	GA	NN	GA	NN	GA	NN
1	1067	237	1066	160	1058	77	1059	116
2		442		472		408		373
3		113		149		162		140
4		71		83		72		65
5		38		56		55		44
6		44		37		47		42
7		32		29		42		55
8		31		20		36		41
9		28		19		32		39
10		11		19		28		24
11		14		6		17		31
12		3		7		22		16
13		2		5		10		23
14		4		3		17		26
15		1		1		21		14
16				3		11		12
17				2		11		8
18						3		2

The table lists numbers of candidate sequences $cand^{GA}_{r=10}$ and $cand^{NN}_{r=10}$ grouped according to their differences (number of mutations) to the most similar sequence from $OPT_{r\,=\,10}$. For this analysis, the first 10 iterations of the designs for the four proteins from *repr_prot* were analyzed.

https://doi.org/10.1371/journal.pone.0256691.t004

Analogously to the evolution of native proteins, each design protocol induces a specific chronological order of residue substitutions and a trajectory that is–under these circumstances–most likely inaccessible, if the order of mutations is different. Thus, if GA and NN choose dissimilar residues for some key positions during the first iterations, the sequence space available to subsequent candidates will be different, if epistasis is dominant. In order to illustrate the existence of epistatic effects, we analyzed mutual dependencies of residue pairs in four design shells.

We manually compared sequence logos resulting from GA and NN generation 100 and searched for positions that were occupied by strikingly different residues. By inspecting the 3D structure of corresponding design candidates, consequences, i.e. pairwise mutual dependencies, of these choices were made plausible. Fig 5 illustrates differences in the orchestration of the design shells of *repr_prot* proteins generated by GA and NN protocols; all arrangements are valid with respect to orientation and Rosetta scores. In the design shell of 2dri, we found mutual dependencies in the occupancies of positions 15 and 235. The GA chose at position 15 preferentially Ala or Asp and at position 235 Glu. In contrast, the NN preferred at position 15 Phe and at position 235 His. In the design shell of 2ifb, the occupancies of residue positions 70 and 72 are mutually dependent: The GA prefers for position 70 His and for position 72 Asn. The NN chose at position 70 Tyr or Phe and at position 72 Leu. In the design shell of 1opb, the GA selected for position 40 Ala and for position 53 Phe or Trp. In contrast, the NN introduced at position 40 His or Ile and at position 53 Val or Ser. In the design shell of 2rct, the GA chose for position 62 preferentially Tyr and for position 81 Glu. In contrast, the NN preferred at position 62 Ser and at position 81 Trp.

In summary, these analyses illustrate strong mutual dependencies in the occupancies of design shell residues. As design shells are generated by mutating residues in a randomly chosen chronology, epistasis directs the protocols to different regions of sequence space.

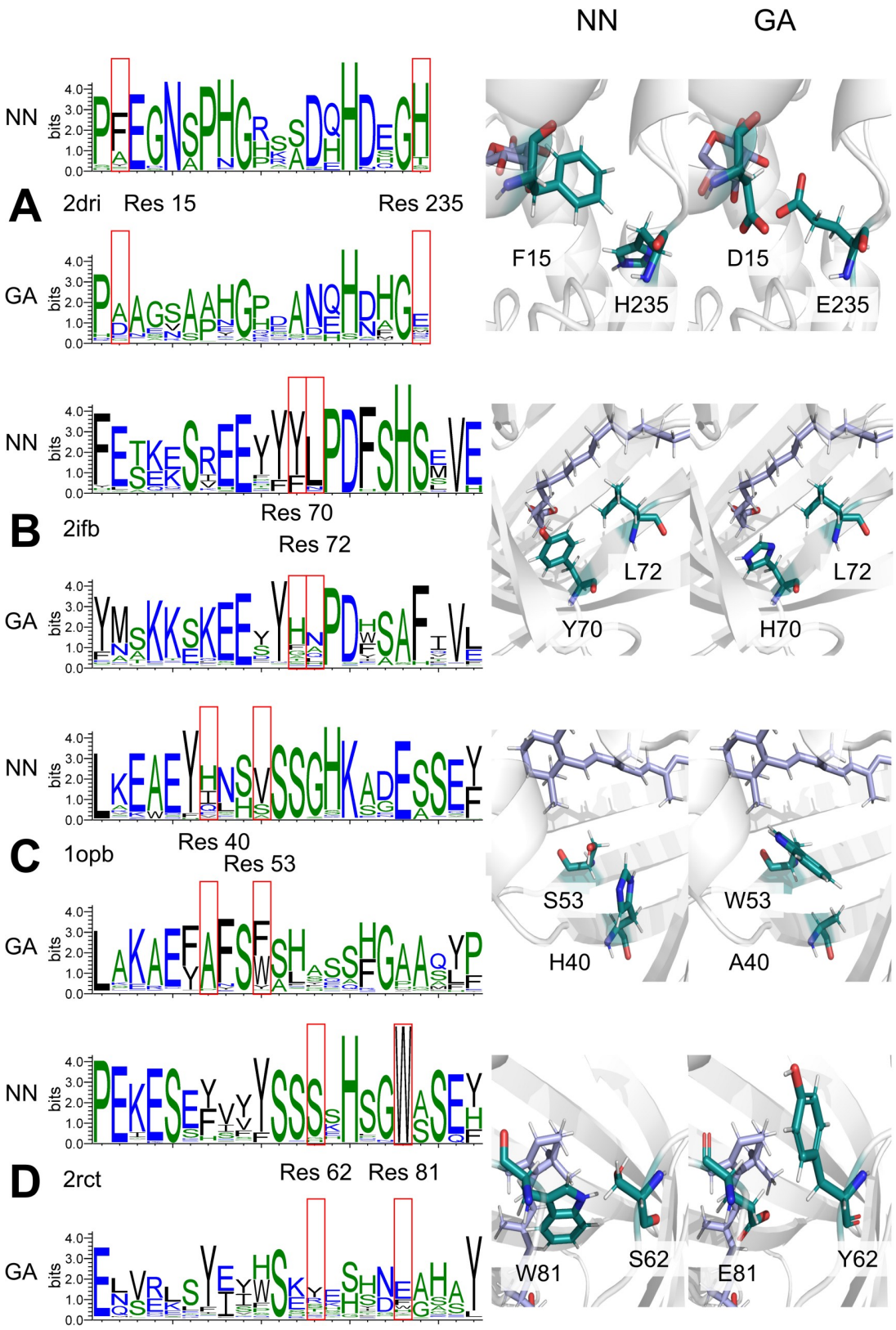

Fig 5. Residue occupancies generated by `Rosetta:MSF:GA` and Rosetta:MSF:NN for four design shells. For each design shell, the GA- and NN-specific logos resulting from the 239 sequences of generation 100 are given. In each case, one residue pair

that is most likely affected by epistatic effects is indicated with red boxes and the 3D orientation of the residues is shown as sticks for the NN (left) and the GA (right) design solution. The designs for the proteins from *repr_prot* were analyzed: (A) 2dri, (B) 2ifb, (C) 1opd, (D) 2rct. The chosen residues are shown in turquoise and the ligand in pale blue; the rest of the protein is represented as a white cartoon.

https://doi.org/10.1371/journal.pone.0256691.g005

The NN outperforms the GA on multiple mutations and adopts iteratively the energy landscape

A major difference of the NN- and GA-based protocol is the number of mutations introduced per iteration, which are 1 to *n* for the NN and one for the GA approach; compare **Table 3**. Thus, one might argue that the superior performance of the NN-based approach is simply a consequence of the more exhaustive sampling of sequence space resulting from the simultaneously introduced multiple mutations. If the number of mutations has a much stronger effect on performance than the protocol, a modified GA that can introduce up to *n* mutations per iteration, must reach a performance that is comparable to the NN-based protocol. Thus, we created a multi-mutation GA protocol, which we named `mmGA`, and equipped it with a function that chooses 1 to *n* mutations as does the NN. **Fig 3** makes clear that multi mutations degraded the performance of the GA: Compared to the standard GA that introduces single mutations, the `mmGA` required twice the number of iterations to find sequences having native-like RS-values.

The training with the tuples generated during the first iteration of the HisB_GA$_{raw}$ design resulted in a relatively low PCC of 0.62 (p = 1.2E-9) and the average error was 2.9 REU. This relatively disappointing result argues in favor of utilizing a simpler and much faster regression model that might replace the NN. On the other hand, the results shown in **Fig 5** are indicative of non-linear dependencies in the orchestration of residue positions, which is in agreement with literature [51–53]. In order to test whether the NNs improved their approximation of the energy landscape during the iterative training, we determined for each iteration the PCC of the RS^{j}_{NN} and the corresponding RS^{j}_{3DM} values. In **Table 5** the result for the four designs from *repr_prot* are tabulated for the first ten iterations. In all four cases, the initially modest correlation reaches a PCC value of 0.95 not later than after 10 iterations. This finding strongly suggests that the iterative training allowed the NNs to improve their approximation of the energy landscape and argues against a simpler modelling approach.

Table 5. Generation-specific PCC values determined for the first ten iterations of four designs.

Generation	2dri	2ifb	1opb	2rct
1	0.53	0.54	0.48	0.42
2	0.84	0.87	0.70	0.69
3	0.90	0.92	0.78	0.74
4	0.95	0.94	0.82	0.82
5	0.97	0.96	0.88	0.89
6	0.98	0.97	0.91	0.92
7	0.98	0.97	0.92	0.94
8	0.98	0.98	0.93	0.94
9	0.98	0.98	0.95	0.96
10	0.98	0.98	0.95	0.96

The table lists Pearson correlation coefficient (PCC) values for the first 10 iterations of the designs with the proteins from *repr_prot* and `Rosetta:MSF:NN:enzdes`.

https://doi.org/10.1371/journal.pone.0256691.t005

For the NNs, a limited number of samples maps the complexity of a problem-specific energy landscape in sufficient detail

Generally, the reconstruction of an all-atom model from an incomplete representation of a protein is a challenging problem [55]. Thus, at first glance it seems surprising that a one-hot encoded representation of candidate sequences suffices to predict the correct *RS* values with an average error of 1.8 REUs; compare **Fig 2**. To determine an RS_{3DM} value, Rosetta has to build a 3D model and to find a combination of rotamers that is suitable for all residues of the design shell. Why is an NN that lacks the assessment of a three-dimensional representation, so successful in an `enzdes` protocol? Although the NN does not explicitly process three-dimensional information, it can learn the energy landscape, because the RS_{3DM} values of the training data implicitly transfer three-dimensional information into the scoring function learned by the NN.

Another example for the beneficial mapping of structure space by means of an NN is `refineD`, which is aimed at protein structure refinement. This program utilizes additional restraints integrated into a Rosetta all-atom energy function that have been deduced by means of a deep convolutional neural field from the starting structure. `refineD` outperformed unrestrained relaxation strategies, most likely because these restraints guide conformational sampling [56].

Furthermore, the performance of our NN and our analysis of residue pairs suggests that the orchestration of the design shells with amino acid residues and their orientation is severely biased. This assumption is supported by recent statistical findings related to residue preferences: The algorithm `NEPRE` assesses successfully the quality of three-dimensional protein models. It is based on a scoring system for residue neighborhood preferences deduced from 14,647 PDB structures. It turned out that certain residues exhibit strong preferences for their neighboring residues and their relative positions [57]. `trRosetta` generates with high quality the three-dimensional structure of proteins based on predictions for inter-residue contacts, distances, and residue orientations. These predictions originate from a deep residual network that analyzes protein-specific MSAs. For the 31 FM targets of the CASP13 contest, the mean TM-score, which signals the correspondence with the native structure, has been 0.625. The score had dropped only marginally to 0.592, if residue orientation has been ignored during structure prediction [58]. Along this line, a restricted number of residue orientations has been observed at certain positions in more than 2600 antibody structures [59]. Taken together, these findings support the idea that `Rosetta:MSF:NN:enzdes` performs well, because the orchestration of the design shell with a restricted number of amino acid residues has a stronger effect on scores than the choice of rotamers.

Supporting information

S1 Fig. Grid search varying the number of neurons of the hidden layers.
(PDF)

S2 Fig. Performance of an NN for two different HisB_GA datasets.
(PDF)

S3 Fig. Convergence of Rosetta:MSF:GA:enzdes and of Rosetta:MSF:NN:enzdes.
(PDF)

S4 Fig. Design performance of an alternative residue representation.
(PDF)

S5 Fig. Design performance with the *talaris* scoring function.
(PDF)

S6 Fig. Amino acid frequency distributions for the outcome of the first and second half of design protocols.
(PDF)

S1 Table. One-hot encoding of the 20 amino acid residues.
(PDF)

S2 Table. Feature vectors of the 20 amino acid residues.
(PDF)

S1 Text. Structure of the benchmark dataset.
(PDF)

S2 Text. Additional redesigns to assess the robustness of Rosetta:MSF:NN.
(PDF)

Author Contributions

Conceptualization: Julian Nazet, Elmar Lang, Rainer Merkl.

Data curation: Julian Nazet.

Formal analysis: Julian Nazet, Elmar Lang, Rainer Merkl.

Funding acquisition: Rainer Merkl.

Investigation: Julian Nazet.

Methodology: Julian Nazet.

Project administration: Rainer Merkl.

Resources: Rainer Merkl.

Software: Julian Nazet.

Supervision: Rainer Merkl.

Validation: Julian Nazet, Elmar Lang, Rainer Merkl.

Visualization: Julian Nazet, Rainer Merkl.

Writing – original draft: Julian Nazet, Rainer Merkl.

Writing – review & editing: Julian Nazet, Elmar Lang, Rainer Merkl.

References

1. Gainza P, Nisonoff HM, Donald BR. Algorithms for protein design. Curr Opin Struct Biol. 2016; 39:16–26. https://doi.org/10.1016/j.sbi.2016.03.006 PMID: 27086078
2. Shah PS, Hom GK, Ross SA, Lassila JK, Crowhurst KA, Mayo SL. Full-sequence computational design and solution structure of a thermostable protein variant. J Mol Biol. 2007; 372(1):1–6. https://doi.org/10.1016/j.jmb.2007.06.032 PMID: 17628593
3. Goldenzweig A, Goldsmith M, Hill SE, Gertman O, Laurino P, Ashani Y, et al. Automated structure- and sequence-based design of proteins for high bacterial expression and stability. Mol Cell. 2016; 63 (2):337–346. https://doi.org/10.1016/j.molcel.2016.06.012 PMID: 27425410
4. Looger LL, Dwyer MA, Smith JJ, Hellinga HW. Computational design of receptor and sensor proteins with novel functions. Nature. 2003; 423(6936):185–190. https://doi.org/10.1038/nature01556 PMID: 12736688

5. Shifman JM, Mayo SL. Exploring the origins of binding specificity through the computational redesign of calmodulin. Proc Natl Acad Sci U S A. 2003; 100(23):13274–13279. https://doi.org/10.1073/pnas.2234277100 PMID: 14597710

6. Fleishman SJ, Whitehead TA, Ekiert DC, Dreyfus C, Corn JE, Strauch EM, et al. Computational design of proteins targeting the conserved stem region of influenza hemagglutinin. Science. 2011; 332 (6031):816–821. https://doi.org/10.1126/science.1202617 PMID: 21566186

7. Procko E, Berguig GY, Shen BW, Song Y, Frayo S, Convertine AJ, et al. A computationally designed inhibitor of an Epstein-Barr viral Bcl-2 protein induces apoptosis in infected cells. Cell. 2014; 157 (7):1644–1656. https://doi.org/10.1016/j.cell.2014.04.034 PMID: 24949974

8. Richter F, Leaver-Fay A, Khare SD, Bjelic S, Baker D. De novo enzyme design using Rosetta3. PLoS One. 2011; 6(5):e19230. https://doi.org/10.1371/journal.pone.0019230 PMID: 21603656

9. Kaplan J, DeGrado WF. De novo design of catalytic proteins. Proc Natl Acad Sci U S A. 2004; 101 (32):11566–11570. https://doi.org/10.1073/pnas.0404387101 PMID: 15292507

10. Röthlisberger D, Khersonsky O, Wollacott AM, Jiang L, Dechancie J, Betker J, et al. Kemp elimination catalysts by computational enzyme design. Nature. 2008; 453(7192):164–166. https://doi.org/10.1038/453164a PMID: 18464727

11. Adolf-Bryfogle J, Kalyuzhniy O, Kubitz M, Weitzner BD, Hu X, Adachi Y, et al. RosettaAntibodyDesign (RAbD): A general framework for computational antibody design. PLoS Comp Biol. 2018; 14(4): e1006112. https://doi.org/10.1371/journal.pcbi.1006112 PMID: 29702641

12. Lippow SM, Wittrup KD, Tidor B. Computational design of antibody-affinity improvement beyond in vivo maturation. Nat Biotechnol. 2007; 25(10):1171–1176. https://doi.org/10.1038/nbt1336 PMID: 17891135

13. Wrabl JO, Gu J, Liu T, Schrank TP, Whitten ST, Hilser VJ. The role of protein conformational fluctuations in allostery, function, and evolution. Biophys Chem. 2011; 159(1):129–141. https://doi.org/10.1016/j.bpc.2011.05.020 PMID: 21684672

14. Leaver-Fay A, Jacak R, Stranges PB, Kuhlman B. A generic program for multistate protein design. PLoS One. 2011; 6(7):e20937. https://doi.org/10.1371/journal.pone.0020937 PMID: 21754981

15. Löffler P, Schmitz S, Hupfeld E, Sterner R, Merkl R. Rosetta:MSF: a modular framework for multi-state computational protein design. PLoS Comp Biol. 2017; 13(6):e1005600. https://doi.org/10.1371/journal.pcbi.1005600 PMID: 28604768

16. Yanover C, Fromer M, Shifman JM. Dead-end elimination for multistate protein design. J Comput Chem. 2007; 28(13):2122–2129. https://doi.org/10.1002/jcc.20661 PMID: 17471460

17. Davey JA, Chica RA. Multistate approaches in computational protein design. Protein Sci. 2012; 21 (9):1241–1252. https://doi.org/10.1002/pro.2128 PMID: 22811394

18. Negron C, Keating AE. Multistate protein design using CLEVER and CLASSY. Methods Enzymol. 2013; 523:171–190. https://doi.org/10.1016/B978-0-12-394292-0.00008-4 PMID: 23422430

19. Allen BD, Mayo SL. An efficient algorithm for multistate protein design based on FASTER. J Comput Chem. 2010; 31(5):904–916. https://doi.org/10.1002/jcc.21375 PMID: 19637210

20. Harbury PB, Plecs JJ, Tidor B, Alber T, Kim PS. High-resolution protein design with backbone freedom. Science. 1998; 282(5393):1462–1467. https://doi.org/10.1126/science.282.5393.1462 PMID: 9822371

21. Fromer M, Yanover C, Harel A, Shachar O, Weiss Y, Linial M. SPRINT: side-chain prediction inference toolbox for multistate protein design. Bioinformatics. 2010; 26(19):2466–2467. https://doi.org/10.1093/bioinformatics/btq445 PMID: 20685957

22. Karimi M, Shen Y. iCFN: an efficient exact algorithm for multistate protein design. Bioinformatics. 2018; 34(17):i811–i820. https://doi.org/10.1093/bioinformatics/bty564 PMID: 30423073

23. Vucinic J, Simoncini D, Ruffini M, Barbe S, Schiex T. Positive multistate protein design. Bioinformatics. 2020; 36(1):122–130. https://doi.org/10.1093/bioinformatics/btz497 PMID: 31199465

24. Havranek JJ, Harbury PB. Automated design of specificity in molecular recognition. Nat Struct Biol. 2003; 10(1):45–52. https://doi.org/10.1038/nsb877 PMID: 12459719

25. Pokala N, Handel TM. Energy functions for protein design: adjustment with protein-protein complex affinities, models for the unfolded state, and negative design of solubility and specificity. J Mol Biol. 2005; 347(1):203–227. S0022-2836(04)01589-X [pii] https://doi.org/10.1016/j.jmb.2004.12.019 PMID: 15733929

26. Humphris-Narayanan E, Akiva E, Varela R, S OC, Kortemme T. Prediction of mutational tolerance in HIV-1 protease and reverse transcriptase using flexible backbone protein design. PLoS Comp Biol. 2012; 8(8):e1002639. https://doi.org/10.1371/journal.pcbi.1002639 PMID: 22927804

27. Howell SC, Inampudi KK, Bean DP, Wilson CJ. Understanding thermal adaptation of enzymes through the multistate rational design and stability prediction of 100 adenylate kinases. Structure. 2014; 22 (2):218–229. https://doi.org/10.1016/j.str.2013.10.019 PMID: 24361272

28. Sevy AM, Wu NC, Gilchuk IM, Parrish EH, Burger S, Yousif D, et al. Multistate design of influenza antibodies improves affinity and breadth against seasonal viruses. Proc Natl Acad Sci U S A. 2019; 116 (5):1597–1602. https://doi.org/10.1073/pnas.1806004116 PMID: 30642961

29. Humphris EL, Kortemme T. Design of multi-specificity in protein interfaces. PLoS Comp Biol. 2007; 3 (8):e164. https://doi.org/10.1371/journal.pcbi.0030164 PMID: 17722975

30. St-Jacques AD, Eyahpaise MEC, Chica RA. Computational Design of Multisubstrate Enzyme Specificity. Acs Catalysis. 2019; 9(6):5480–5485. https://doi.org/10.1021/acscatal.9b01464

31. Leaver-Fay A, Tyka M, Lewis SM, Lange OF, Thompson J, Jacak R, et al. ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules. Methods in Enzymology. 2011; 487:545–574. https://doi.org/10.1016/B978-0-12-381270-4.00019-6 PMID: 21187238

32. Kaufmann KW, Lemmon GH, Deluca SL, Sheehan JH, Meiler J. Practically useful: what the Rosetta protein modeling suite can do for you. Biochemistry. 2010; 49(14):2987–2998. https://doi.org/10.1021/bi902153g PMID: 20235548

33. Starr TN, Thornton JW. Epistasis in protein evolution. Protein Sci. 2016; 25(7):1204–1218. https://doi.org/10.1002/pro.2897 PMID: 26833806

34. Miton CM, Tokuriki N. How mutational epistasis impairs predictability in protein evolution and design. Protein Sci. 2016; 25(7):1260–1272. https://doi.org/10.1002/pro.2876 PMID: 26757214

35. Rost B. PHD: predicting one-dimensional protein structure by profile-based neural networks. Methods in Enzymology. 1996; 266:525–539. https://doi.org/10.1016/s0076-6879(96)66033-9 PMID: 8743704

36. Kuhlman B, Bradley P. Advances in protein structure prediction and design. Nat Rev Mol Cell Biol. 2019; 20(11):681–697. https://doi.org/10.1038/s41580-019-0163-x PMID: 31417196

37. Krieger E, Darden T, Nabuurs SB, Finkelstein A, Vriend G. Making optimal use of empirical energy functions: force-field parameterization in crystal space. Proteins. 2004; 57(4):678–683. https://doi.org/10.1002/prot.20251 PMID: 15390263

38. Nivón LG, Bjelic S, King C, Baker D. Automating human intuition for protein design. Proteins. 2014; 82 (5):858–866. https://doi.org/10.1002/prot.24463 PMID: 24265170

39. Schrödinger. The PyMOL molecular graphics system, version 1.7. 2015.

40. Thomas AJ, Petridis M, Walters SD, Gheytassi SM, Morgan RE, editors. Two hidden layers are usually better than one. International Conference on Engineering Applications of Neural Networks; 2017: Springer.

41. Ramsundar B, Zadeh RB. TensorFlow for deep learning: from linear regression to reinforcement learning: O'Reilly Media, Inc.; 2018.

42. Davis J, Goadrich M, editors. The relationship between precision-recall and ROC curves. ICML'06: Proceedings of the 23rd International Conference on Machine Learning; 2006; Pittsburgh: ACM, NewYork.

43. Huang X, Pearce R, Zhang Y. EvoEF2: accurate and fast energy function for computational protein design. Bioinformatics. 2020; 36(4):1135–1142. https://doi.org/10.1093/bioinformatics/btz740 PMID: 31588495

44. Guerois R, Nielsen JE, Serrano L. Predicting changes in the stability of proteins and protein complexes: A study of more than 1000 mutations. J Mol Biol. 2002; 320(2):369–387. https://doi.org/10.1016/S0022-2836(02)00442-4 PMID: 12079393

45. Bogardt RA Jr., Jones BN, Dwulet FE, Garner WH, Lehman LD, Gurd FR. Evolution of the amino acid substitution in the mammalian myoglobin gene. J Mol Evol. 1980; 15(3):197–218. https://doi.org/10.1007/BF01732948 PMID: 7401178

46. Alford RF, Leaver-Fay A, Jeliazkov JR, O'Meara MJ, DiMaio FP, Park H, et al. The Rosetta all-atom energy function for macromolecular modeling and design. J Chem Theory Comput. 2017; 13(6):3031–3048. https://doi.org/10.1021/acs.jctc.7b00125 PMID: 28430426

47. O'Meara MJ, Leaver-Fay A, Tyka MD, Stein A, Houlihan K, DiMaio F, et al. Combined covalent-electrostatic model of hydrogen bonding improves structure prediction with Rosetta. J Chem Theory Comput. 2015; 11(2):609–622. https://doi.org/10.1021/ct500864r PMID: 25866491

48. Huang PS, Boyken SE, Baker D. The coming of age of de novo protein design. Nature. 2016; 537 (7620):320–327. https://doi.org/10.1038/nature19946 PMID: 27629638

49. Loshbaugh AL, Kortemme T. Comparison of Rosetta flexible-backbone computational protein design methods on binding interactions. Proteins. 2020; 88(1):206–226. https://doi.org/10.1002/prot.25790 PMID: 31344278

50. Kuhlman B, Baker D. Native protein sequences are close to optimal for their structures. Proc Natl Acad Sci U S A. 2000; 97(19):10383–10388. https://doi.org/10.1073/pnas.97.19.10383 PMID: 10984534

51. Weinreich DM, Lan Y, Wylie CS, Heckendorn RB. Should evolutionary geneticists worry about higher-order epistasis? Curr Opin Genet Dev. 2013; 23(6):700–707. https://doi.org/10.1016/j.gde.2013.10.007 PMID: 24290990

52. Lunzer M, Golding GB, Dean AM. Pervasive cryptic epistasis in molecular evolution. PLoS Genet. 2010; 6(10):e1001162. https://doi.org/10.1371/journal.pgen.1001162 PMID: 20975933

53. Tamer YT, Gaszek IK, Abdizadeh H, Batur TA, Reynolds KA, Atilgan AR, et al. High-order epistasis in catalytic power of dihydrofolate reductase gives rise to a rugged fitness landscape in the presence of trimethoprim selection. Mol Biol Evol. 2019; 36(7):1533–1550. https://doi.org/10.1093/molbev/msz086 PMID: 30982891

54. Yang G, Anderson DW, Baier F, Dohmen E, Hong N, Carr PD, et al. Higher-order epistasis shapes the fitness landscape of a xenobiotic-degrading enzyme. Nat Chem Biol. 2019; 15(11):1120–1128. https://doi.org/10.1038/s41589-019-0386-3 PMID: 31636435

55. Badaczewska-Dawid AE, Kolinski A, Kmiecik S. Computational reconstruction of atomistic protein structures from coarse-grained models. Comput Struct Biotechnol J. 2020; 18:162–176. https://doi.org/10.1016/j.csbj.2019.12.007 PMID: 31969975

56. Bhattacharya D. refineD: improved protein structure refinement using machine learning based restrained relaxation. Bioinformatics. 2019; 35(18):3320–3328. https://doi.org/10.1093/bioinformatics/btz101 PMID: 30759180

57. Liu S, Xiang X, Gao X, Liu H. Neighborhood preference of amino acids in protein structures and its applications in protein structure assessment. Sci Rep. 2020; 10(1):4371. https://doi.org/10.1038/s41598-020-61205-w PMID: 32152349

58. Yang J, Anishchenko I, Park H, Peng Z, Ovchinnikov S, Baker D. Improved protein structure prediction using predicted interresidue orientations. Proc Natl Acad Sci U S A. 2020:201914677. https://doi.org/10.1073/pnas.1914677117 PMID: 31896580

59. Leem J, Georges G, Shi J, Deane CM. Antibody side chain conformations are position-dependent. Proteins. 2018; 86(4):383–392. https://doi.org/10.1002/prot.25453 PMID: 29318667