Designing real novel proteins using deep graph neural networks

Alexey Strokach[1], David Becerra[2], Carles Corbi[2], Albert Perez-Riba[2] and Philip M. Kim[1,2,3,*]

1. *Department of Computer Science. University of Toronto, Toronto, ON M5S 3E1, Canada.*
2. *Donnelly Centre for Cellular and Biomolecular Research. University of Toronto, Toronto, ON M5S 3E1, Canada.*
3. *Department of Molecular Genetics. University of Toronto, Toronto, ON M5S 3E1, Canada.*

* To whom correspondence should be addressed: pi@kimlab.org

Abstract

Protein structure and function is determined by the arrangement of the linear sequence of amino acids in 3D space. Despite substantial advances, precisely designing sequences that fold into a predetermined shape (the “protein design” problem) remains difficult. We show that a deep graph neural network, ProteinSolver, can solve protein design phrased as a constraint satisfaction problem (CSP). To sidestep the considerable issue of optimizing the network architecture, we first develop a network that is accurately able to solve the related and straightforward problem of Sudoku puzzles. Recognizing that each protein design CSP has many solutions, we train this network on millions of real protein sequences corresponding to thousands of protein structures. We show that our method rapidly designs novel protein sequences and perform a variety of *in silico* and *in vitro* validations suggesting that our designed proteins adopt the predetermined structures.

One Sentence Summary: A neural network optimized using Sudoku puzzles designs protein sequences that adopt predetermined structures.

Protein structure and function emerges from the specific geometric arrangement of their linear sequence of amino acids, commonly referred to as a fold. Engineering sequences to fold in a specific structure and function has a broad variety of uses, including academic research (*1*), industrial process engineering (*2*), and most notably, protein-based therapeutics, which are now a very important class of drugs (*3*, *4*).

However, despite extraordinary advances, designing a sequence from scratch to adopt a desired structure (referred to as the “inverse folding” or “protein design” problem) remains a very challenging task. Conventionally, a sampling technique such as Markov-chain Monte Carlo is used to generate a sequence optimized with respect to a force-field or statistical potential (*4*–*6*). Limitations of those methods include the relatively low accuracy of current force fields (*7*, *8*) and the inability to sample more than a miniscule portion of the vast search space (sequence space size is 20^N, N being the number of residues). While there have been successful approaches that screen many thousands of individual designs using *in vitro* selection (*9*, *10*), these remain reliant on relatively labor-intensive experiments.

Here we overcome these limitations by combining a classic idea with a novel methodology. Filling a specific target structure with a new sequence can be formulated as a constraint satisfaction problem (CSP) where the goal is to assign amino acid labels to residues in a polymer chain such that the forces between interaction amino acids are favorable and compatible with the fold (see Fig 1). To overcome previous difficulties with phrasing protein design as a CSP (*11*, *12*), we elucidate the rules governing constraints using deep learning. Such methods have been applied to a vast diversity of fields with impressive results (*13*–*15*), partly because they can infer hitherto hidden patterns from sufficiently large training sets. For proteins, the set of different protein folds is only modestly large with a few thousand superfamilies in CATH (*16*). Indeed, previous attempts at using deep learning approaches for protein design used structural features and thus only trained on relatively small datasets and achieved

moderate success as of yet without any experimental validation (*17–20*). However, the number of sequences that share these structural templates is many orders of magnitude larger (about 70,000,000 sequences map to the CATH superfamilies), reflecting the fact that the protein design problem is inherently underdetermined with a relatively large solution space. Thus, a suitable deep neural network trained on these sequence-structure relationships could potentially outperform previous models to solve the protein design problem.

The adjacency matrix is commonly used to represent protein folds; it is an NxN matrix consisting of the distances (in Angstrom) between residues which are spatially close and thus interacting (we used a distance threshold of 12 Angstrom); such pairs of residues are subject to constraints (e.g., in such pairs, two positive charges are usually not well tolerated). A given protein structure corresponding to one distance matrix can be formed by many different homologous sequences; these sequences all satisfy the constraints imposed. Such solutions to the constraint satisfaction problem are given to us by evolution and are available in sequence repositories such as Pfam or Gene3D (*21*). While the rules for this CSP for any specific fold are easily found by simply comparing sequences from one of such repositories and can be given as position weight matrices (PWMs), often represented as sequence logos, it has thus far not been possible to deduce general rules - those that would connect any given fold or adjacency matrix with a set of sequences. We here use a graph neural network to elucidate these rules. The graph in this case is made up of nodes (corresponding to amino acids) with edges being the spatial interactions between amino acids that are represented in the distance matrix. The edges thus represent the constraints that are imposed on the node properties (amino acid types).

As there had been relatively little work in using graph neural networks to solve CSPs (*22*, *23*), it was quite difficult to find an optimal network architecture for this relatively complex problem. Hence, we first engineered and optimized a graph neural network model to solve the related and well-defined problem

of Sudoku (see Fig 1). Sudoku is a straightforward form of a CSP (*24*). Our architecture corresponds to a graph neural network; node attributes (in Sudoku, integers from 1-9) are embedded in a 128 dimensional space and later subjected to edge convolution taking into account edge indices (see Methods and Fig 1). We generated 30 million Sudoku puzzles of above-average difficulty, and we trained the network to solve those puzzles by minimizing the cross-entropy loss between network predictions and the actual missing numbers (Fig 2A). Our final graph neural network correctly predicts 72% of the missing numbers in a single pass through the network and close to 90% of the missing number if we pass the input through the network multiple times, each time adding as a known value a single prediction from the previous iteration in which the network is the most confident (Fig 2B). Similar accuracy is achieved on an independent test containing puzzles from an online Sudoku puzzle provider (Fig. 2C).

After optimizing the network architecture and parameters for the well-defined problem of solving Sudoku puzzles, we applied the same network to protein design. We compiled a dataset of 72 million unique Gene3D domain sequences from UniParc (*25*) for which a structural template could be found in the PDB (*26*), and we trained the network by providing as input a partially-masked amino acid sequence and an adjacency matrix adapted from the structural template and minimizing the cross-entropy loss between network predictions and the identities of the masked amino acid residues (Fig. 1). The training and validation accuracies achieved by the network reach a plateau after around 100 million training examples (about 6 epochs), with a training accuracy of ~22% and a validation accuracy of ~32% when half of the residues are masked in the starting sequence (Fig. 2D). The training accuracy is lower than the validation accuracy because, while the training dataset has no restriction on the similarity between sequences and structural templates, for the validation dataset we included only those sequences for which a structural template with at least 80% sequence identity to the query could be found. Reconstruction accuracy is considerably lower than Sudoku (see Fig 2A-C), as was

expected: the Sudoku CSP has a single well-defined solution, thus an accuracy approaching 1 is possible. By contrast, each protein adjacency matrix can be adopted by many different sequences. The achieved accuracy of 30%-40% roughly corresponds to the common level of sequence identity within a protein fold (*16*). Evaluating our network by having it reconstruct sequences from our validation dataset, we note that even when removing the majority of the residues, a large proportion of the sequences are reconstructed with a sequence identity around 35% (see Fig 2F). It should be noted that reconstruction accuracy much higher than this is not to be expected (as sequences can vary substantially while retaining the same fold) and is indeed not a meaningful measure of algorithm performance.

We next asked whether network scores for single mutations can be predictive of whether that mutation is stabilizing or destabilizing; presumably, a destabilizing mutation would also disrupt some of the constraints in the CSP and would thus be scored unfavourably by our graph neural network. We indeed find that we achieve a relatively strong correlation (Pearson R: 0.42) with experimentally measured mutations from Protherm (*27*) (see Fig S1). Classical and statistical methods such as Rosetta, FoldX or Elaspic have long been in use for this problem and perform quite well, and indeed somewhat better than our network (*28*–*30*) (Rosetta obtaining Pearson R: 0.56 in our hands, see Fig S1). However, it should be noted that our network was not optimized for this particular task (and was never trained on any ΔΔG data) and indeed would not be expected to perform exceptionally well at this task, since many or even most mutations in the data set would still allow the sequence to satisfy the CSP (and the protein to fold). We also note that evaluation using our network is vastly faster than classical methods (see below).

As our graph network is able to reconstruct sequences with missing residues and score mutations with relatively high accuracy, we sought to use the network to generate entire sequences for a given structure (adjacency matrix). This corresponds to *de novo* protein design - designing a novel sequence

for a given fold. To this end, we chose four protein folds that had been left out of the training set and cover the breadth of the CATH hierarchy. We used an A*-like algorithm to obtain sequences that score highly with our network (see methods). In conventional protein design, both A* and dead-end elimination, as well as Markov-chain Monte Carlo sampling approaches to obtain sequences scored by a classical forcefield have been applied (*31*, *32*). However, this has remained a difficult problem, in part due to the great computational cost involved. Due to many factors, including the fact that ProteinSolver does not require rotamer optimization to properly evaluate a single sequence, evaluation of our network is many orders of magnitude faster than current protein design methods (see Table S1).

For the four folds (mainly α, serum albumin; mainly β, PDZ3 domain; mainly β, immunoglobulin; α+β, alanine racemase), we generated 200,000 sequences each using A* search. We find that the sequences tend to fall within 37-44% sequence identity with the native sequence, in line with the sequence identity within the respective protein families (see Fig S2-S5). As no information about the sequences was provided to the algorithm, this suggests that it was able to learn significant features of protein structure. We also find that when predicting secondary structure of our sequences (*33*), it matches the target secondary structures almost exactly (see Fig S2-S5).

We next chose the top 20,000 (10%) sequences, as scored by our network, for each of the 4 folds for further evaluation. First, we used QUARK (*34*), a *de novo* (not template-based) structure prediction algorithm to obtain structures for our sequences; for all four folds the obtained structures match the initial PDB structure (that corresponds to the distance matrix used to generate the sequences) almost exactly (see Fig 3A-D). We evaluate the quality of the reconstruction using a number of computational metrics, including Modeller molpdf and Rosetta REU. In all cases, the scores are in the same range or better than the target structure, suggesting that the sequences that our network generated *do novo* indeed fold in the shape corresponding to the adjacency matrix (see Fig 3E, 3F). For a final computational evaluation, we ran 100ns molecular dynamics on each structure, including the target

structures. We find that in most cases, the target structure and the structures predicted for the designed sequences show similar root mean square fluctuation in the MD simulation, again suggesting that our sequences fold into the predetermined structure (see Fig 4A-D).

Finally, we chose two designed sequences and the corresponding sequences of the target structures (for the albumin and racemase structural templates) to evaluate experimentally. We ordered each sequence and expressed them as his-tagged constructs for Ni-NTA affinity purification. After purification, we evaluated the secondary structure of each protein using circular dichroism spectroscopy (CD). Each secondary structure element has a distinctive absorbance spectra in the far-UV region, thus similar folds should present similar absorbance spectra. We find that both sequences show a CD spectrum very similar to the native sequences. Taken together with the rest of the evidence from molecular dynamics, Modeller and Rosetta assessment scores, this strongly suggests that these sequences adopt the predetermined fold (see Fig 4E and 4F). This is particularly striking in the case of the albumin template where the spectra are indistinguishable (Fig 4E). The designed sequence from the racemase template displayed a considerable loss of solubility compared to the sequence from the target structure. Although this made its characterization challenging, we were able to obtain a clear spectra by combining a low ionic strength buffer (10 mM Na-phosphate, pH 8) with a 10 mm cuvette. While the resulting CD spectrum is somewhat different from the target (Fig 4F), this may be due to technical issues resulting from low solubility or a more dynamic nature of the designed protein (consistent with the molecular dynamics in Fig 4C); the spectrum definitely corresponds to a folded helical structure consistent with the predetermined fold.

We present ProteinSolver, a graph neural network approach to protein design that shows remarkable accuracy in generating novel protein sequences that fold into a predetermined fold. Previous

approaches were hampered by the enormous computational complexity, in particular when taking into account backbone flexibility. Our approach sidesteps this problem and delivers accurate designs for a wide range of folds. It is expected that it would be able to generate sequence for completely novel imagined protein folds. As a neural network approach, its evaluation is many orders of magnitude faster than classical approaches that will enable the exploration of vastly more potential backbones.

References

1. M. C. Deller, L. Kong, B. Rupp, Protein stability: a crystallographer's perspective. *Acta Crystallogr. Sect. F Struct. Biol. Commun.* **72**, 72–95 (2016).
2. A. S. Bommarius, M. F. Paye, Stabilizing biocatalysts. *Chem. Soc. Rev.* **42**, 6534–6565 (2013).
3. S. Fischman, Y. Ofran, Computational design of antibodies. *Curr. Opin. Struct. Biol.* **51**, 156–162 (2018).
4. A. Chevalier, D.-A. Silva, G. J. Rocklin, D. R. Hicks, R. Vergara, P. Murapa, S. M. Bernard, L. Zhang, K.-H. Lam, G. Yao, C. D. Bahl, S.-I. Miyashita, I. Goreshnik, J. T. Fuller, M. T. Koday, C. M. Jenkins, T. Colvin, L. Carter, A. Bohn, C. M. Bryan, D. A. Fernández-Velasco, L. Stewart, M. Dong, X. Huang, R. Jin, I. A. Wilson, D. H. Fuller, D. Baker, Massively parallel *de novo* protein design for targeted therapeutics. *Nature*. **550**, 74–79 (2017).
5. D. Shultis, P. Mitra, X. Huang, J. Johnson, N. A. Khattak, F. Gray, C. Piper, J. Czajka, L. Hansen, B. Wan, K. Chinnaswamy, L. Liu, M. Wang, J. Pan, J. Stuckey, T. Cierpicki, C. H. Borchers, S. Wang, M. Lei, Y. Zhang, Changing the Apoptosis Pathway through Evolutionary Protein Design. *J. Mol. Biol.* **431**, 825–841 (2019).
6. M. G. F. Sun, P. M. Kim, Data driven flexible backbone protein design. *PLOS Comput. Biol.* **13**, e1005722 (2017).
7. S. Khan, M. Vihinen, Performance of protein stability predictors. *Hum. Mutat.* **31**, 675–684 (2010).
8. B. M. Kroncke, A. M. Duran, J. L. Mendenhall, J. Meiler, J. D. Blume, C. R. Sanders, Documentation of an Imperative To Improve Methods for Predicting Membrane Protein Stability. *Biochemistry*. **55**, 5002–5009 (2016).
9. M. G. F. Sun, M.-H. Seo, S. Nim, C. Corbi-Verge, P. M. Kim, Protein engineering by highly parallel screening of computationally designed variants. *Sci. Adv.* **2**, e1600692 (2016).
10. G. J. Rocklin, T. M. Chidyausiku, I. Goreshnik, A. Ford, S. Houliston, A. Lemak, L. Carter, R. Ravichandran, V. K. Mulligan, A. Chevalier, C. H. Arrowsmith, D. Baker, Global analysis of protein folding using massively parallel design, synthesis, and testing. *Science*. **357**, 168–175 (2017).
11. A. D. Ullah, K. Steinhöfel, A hybrid approach to protein folding problem integrating constraint programming with local search. *BMC Bioinformatics*. **11 Suppl 1**, S39–S39 (2010).
12. A. Dal Palù, A. Dovier, F. Fogolari, Constraint Logic Programming approach to protein structure prediction. *BMC Bioinformatics*. **5**, 186 (2004).
13. J. Kocić, N. Jovičić, V. Drndarević, An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms. *Sensors*. **19** (2019), doi:10.3390/s19092064.
14. D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, Mastering the game of Go without human knowledge. *Nature*. **550**, 354 (2017).
15. M. Wainberg, D. Merico, A. Delong, B. J. Frey, Deep learning in biomedicine. *Nat. Biotechnol.* **36**, 829 (2018).
16. N. L. Dawson, T. E. Lewis, S. Das, J. G. Lees, D. Lee, P. Ashford, C. A. Orengo, I. Sillitoe, CATH: an expanded resource to predict protein function through structure and sequence. *Nucleic Acids Res.* **45**, D289–D295 (2016).
17. J. Wang, H. Cao, J. Z. H. Zhang, Y. Qi, Computational Protein Design with Deep Learning Neural Networks. *Sci. Rep.* **8**, 6349 (2018).
18. Z. Li, Y. Yang, E. Faraggi, J. Zhan, Y. Zhou, Direct prediction of profiles of sequences compatible with a protein structure by neural networks with fragment-based local and energy-based nonlocal profiles. *Proteins Struct. Funct. Bioinforma.* **82**, 2565–2573 (2014).
19. J. Ingraham, V. K. Garg, R. Barzilay, T. Jaakkola, Generative Models for Graph-Based Protein

Design (2019) (available at https://openreview.net/forum?id=SJgxrLLKOE).
20. J. O'Connell, Z. Li, J. Hanson, R. Heffernan, J. Lyons, K. Paliwal, A. Dehzangi, Y. Yang, Y. Zhou, SPIN2: Predicting sequence profiles from protein structures using deep neural networks. *Proteins Struct. Funct. Bioinforma.* **86**, 629–633 (2018).
21. T. E. Lewis, I. Sillitoe, N. Dawson, S. D. Lam, T. Clarke, D. Lee, C. Orengo, J. Lees, Gene3D: Extensive prediction of globular domains in proteins. *Nucleic Acids Res.* **46**, D1282–D1282 (2017).
22. R. B. Palm, U. Paquet, O. Winther, Recurrent Relational Networks. *ArXiv171108028 Cs* (2017) (available at http://arxiv.org/abs/1711.08028).
23. M. O. R. Prates, P. H. C. Avelar, H. Lemos, L. Lamb, M. Vardi, Learning to Solve NP-Complete Problems - A Graph Neural Network for Decision TSP. *ArXiv180902721 Cs Stat* (2018) (available at http://arxiv.org/abs/1809.02721).
24. H. Simonis, in *CP Workshop on modeling and reformulating Constraint Satisfaction Problems* (2005), vol. 12, pp. 13–27.
25. C. H. Wu, R. Apweiler, A. Bairoch, D. A. Natale, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, R. Mazumder, C. O'Donovan, N. Redaschi, B. Suzek, The Universal Protein Resource (UniProt): an expanding universe of protein information. *Nucleic Acids Res.* **34**, D187–D191 (2006).
26. H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, P. E. Bourne, The Protein Data Bank. *Nucleic Acids Res.* **28**, 235–242 (2000).
27. K. A. Bava, M. M. Gromiha, H. Uedaira, K. Kitajima, A. Sarai, ProTherm, version 4.0: thermodynamic database for proteins and mutants. *Nucleic Acids Res.* **32**, D120–D121 (2004).
28. H. Park, P. Bradley, P. Greisen, Y. Liu, V. K. Mulligan, D. E. Kim, D. Baker, F. DiMaio, Simultaneous Optimization of Biomolecular Energy Functions on Features from Small Molecules and Macromolecules. *J. Chem. Theory Comput.* **12**, 6201–6212 (2016).
29. J. Schymkowitz, J. Borg, F. Stricher, R. Nys, F. Rousseau, L. Serrano, The FoldX web server: an online force field. *Nucleic Acids Res.* **33**, W382–W388 (2005).
30. D. K. Witvliet, A. Strokach, A. F. Giraldo-Forero, J. Teyra, R. Colak, P. M. Kim, ELASPIC web-server: proteome-wide structure-based prediction of mutation effects on protein stability and binding affinity. *Bioinforma. Oxf. Engl.* **32**, 1589–1591 (2016).
31. J. Desmet, M. D. Maeyer, B. Hazes, I. Lasters, The dead-end elimination theorem and its use in protein side-chain positioning. *Nature*. **356**, 539–542 (1992).
32. A. R. Leach, A. P. Lemon, Exploring the conformational space of protein side chains using dead-end elimination and the A* algorithm. *Proteins Struct. Funct. Bioinforma.* **33**, 227–239 (1998).
33. L. J. McGuffin, K. Bryson, D. T. Jones, The PSIPRED protein structure prediction server. *Bioinformatics*. **16**, 404–405 (2000).
34. D. Xu, Y. Zhang, Ab initio protein structure assembly using continuous structure fragments and optimized knowledge-based force field. *Proteins Struct. Funct. Bioinforma.* **80**, 1715–1735 (2012).

Acknowledgements: We thank Quaid Morris for valuable discussions. **Funding:** AS acknowledges support from an NSERC PGS-D graduate scholarship. DB acknowledges support from the Fonds de recherche du Québec - Nature et technologies (FRQNT) - Bourses de recherche postdoctorale. PMK acknowledges support from an NSERC Discovery grant and a CIHR Project grant. **Author contributions:** Conceptualization: AS and PMK. Data curation: AS and DB. Formal analysis and Methodology: AS. Software: AS and DB. Resources: AS, DB, CC, APR and PMK. Validation: CC, DB and APR. Writing: AS, DB, CC, APR and PMK. Project administration: PMK. Supervision: PMK. Funding acquisition: PMK. **Competing interests:** The authors are in the process of obtaining a patent on the described technology. **Data and materials availability:** All code and data will be publicly available on github.

Supplementary Materials (see separate file)

Materials and Methods

Table S1

Figures S1-S6

Figure Legends

Figure 1. Deep graph neural network to solve the protein design problem. The problem is phrased as a constraint satisfaction problem with different amino acids putting constraints on other amino acids that are spatially close. These constraints can be represented as a distance matrix and are analogous to the constraints given to different numbers in the puzzle Sudoku (where a constraint matrix is also given). This allowed us to devise and optimize our solver graph neural network architecture. This architecture embeds both edge and node attributes in an X-dimensional space, and given the edge indices (i.e., the graph), they are subjected to a two layer edge convolution. This architecture then efficiently solves Sudoku puzzles and generates accurate protein sequences.

Figure 2. (A) Training and validation accuracy of a graph neural network that is being trained to solve Sudoku puzzles. **(B-C)** Accuracy achieved by a graph neural network trained to solve Sudoku puzzles on the validation dataset **(B)**, comprised of 1000 Sudoku puzzles generated in the same way as the training dataset, and on the test dataset **(C)**, comprised of 30 Sudoku puzzles extracted from an online Sudoku puzzle provider. Predictions were made using either a single pass through the network (blue bars) or by running the network repeatedly, each time taking a single prediction in which the network is the most confident (red bars). **(D)** Training and validation accuracy of a graph neural network that is being trained to recover the identity of masked amino acid residues (the ProteinSolver network). During training, 50% of the amino acid residues in each input sequence are randomly masked as missing. **(E)** Accuracy achieved by the ProteinSolver network on the test dataset, comprised of 10,000 sequences and adjacency matrices of proteins possessing a different shape (Gene3d domain) than proteins in the training and validation datasets. In the case of blue bars, predictions were made using a single pass through the network, while in the case of red bars, predictions were made by running the network repeatedly, each time taking a single prediction in which the network is the most confident. **(F)**

Sequence identity between generated and reference sequences in cases where 0%, 50%, or 80% of the reference sequences are made available to the network.

Figure 3. Generating sequences that fold into the four main folding classes (α, β+α, β/α and β). (**A**) *serum albumin (pdb_id: 1n5u)*, which is mostly comprised of alpha helices. (**B**) *immunoglobulin (pdb_id: 4unu)*, a protein containing mainly beta sheets. (**C**) *alanine racemase (pdb_id: 4beu)*, a protein containing both alpha helices and beta sheets. (**D**) *PDZ3* domain (*pdb_id:4z8j*), which is mostly comprised of beta sheets. For each protein, the target structure (from which the adjacency matrix was extracted) is shown on the left, and the structure of the ProteinSolver-designed sequence (based on this adjacency matrix) is shown on the right. The structure of the ProteinSolver-designed sequence was generated by QUARK using, as input, the sequence with the highest network score. The rms between the structures (i.e., target vs QUARK) are 3.35A, 1.5A, 1.05A and 1.64A for (A), (B), (C) and (D), respectively. Finally, (**E**) and (**F**) show the scores computed by Modeller (molPDF, lower is better) and Rosetta (Rosetta Energy Units, lower is better), respectively, for 20,000 structures constructed for ProteinSolver-designed sequences. The measures calculated for the target structures are shown as stars in the plots.

Figure 4. Molecular Dynamics and CD experiments. To validate that our designed sequences indeed adopt the pre-determined structures, we made use of computational and experimental methods. Computationally, we performed 100ns molecular dynamics simulations with both the protein in the target structure and our *de novo* designs (**A-D)**. In all four cases, the designs show comparable fluctuation in molecular dynamics to the original target proteins, indicating that they are of comparable stability as the target and thus stably folded. **(E,F)** We also experimentally expressed two of our designs and tested their structure using circular dichroism spectroscopy. We find that they are definitely folded proteins in solutions and that they adopt the same (**E)** or very similar (**F**) structure as the protein that supplied the target structure.

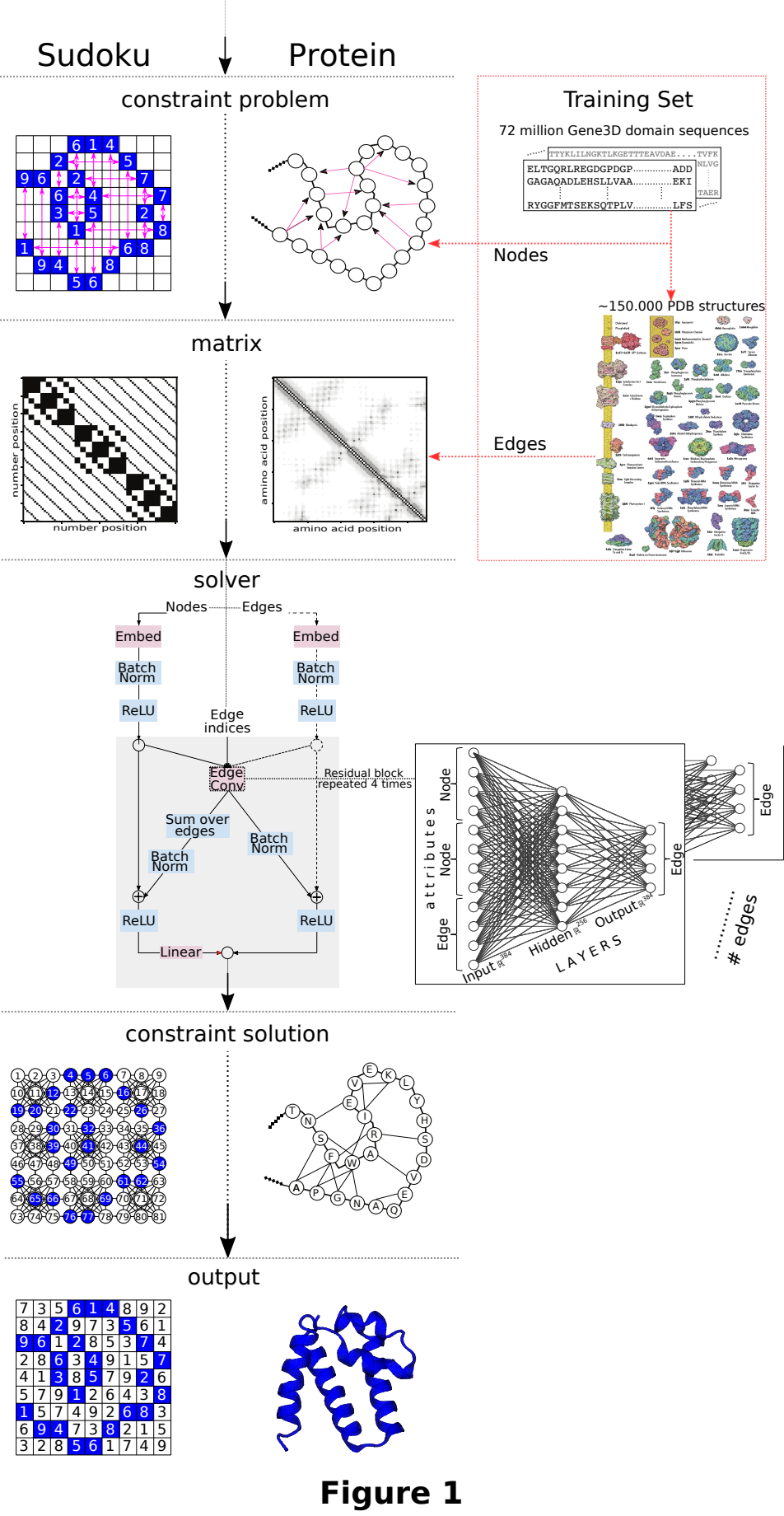

Sudoku
Protein
constraint problem
Training Set
72 million Gene3D domain sequences
Nodes
~150.000 PDB structures
matrix
Edges
number position
amino acid position
solver
Nodes
Edges
Embed
Batch Norm
ReLU
Edge indices
Edge Conv
Residual block repeated 4 times
Sum over edges
Linear
attributes
Node
Edge
Input
Hidden
Output
LAYERS
edges
constraint solution
output

Figure 1

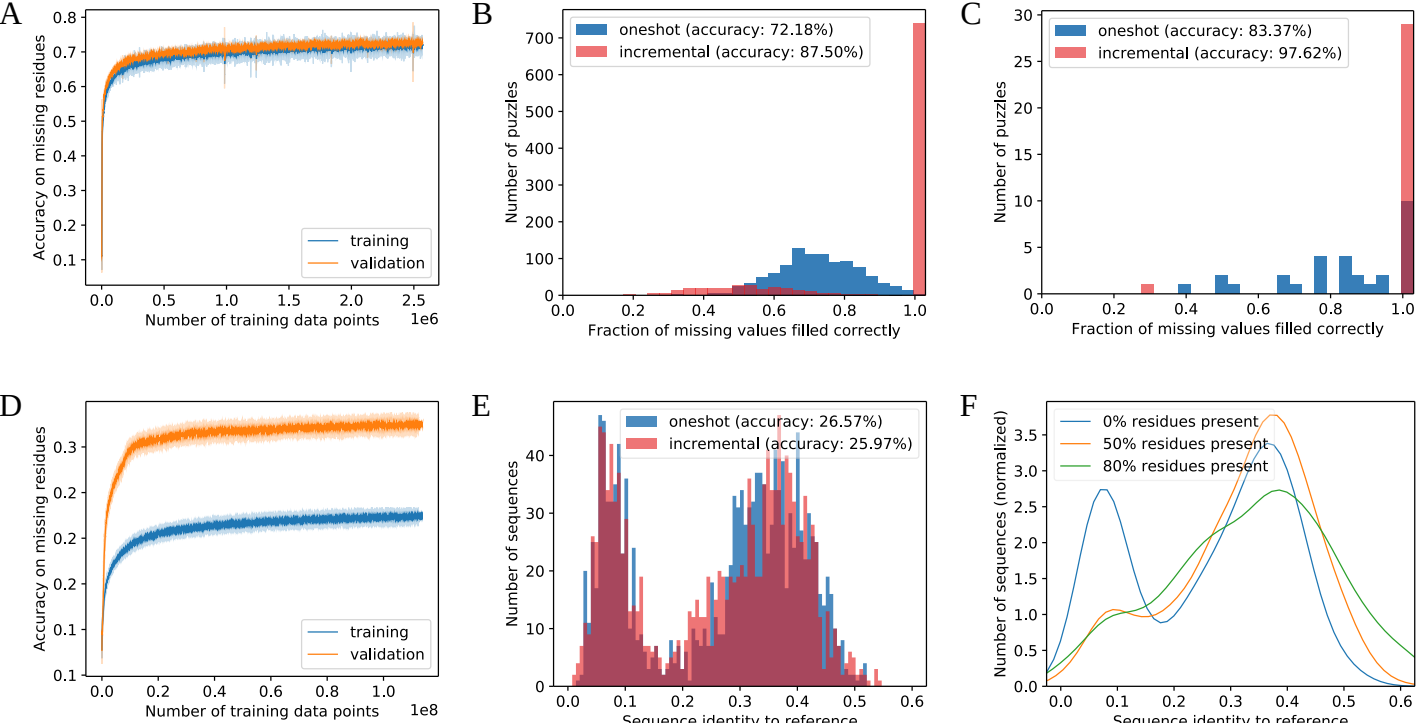
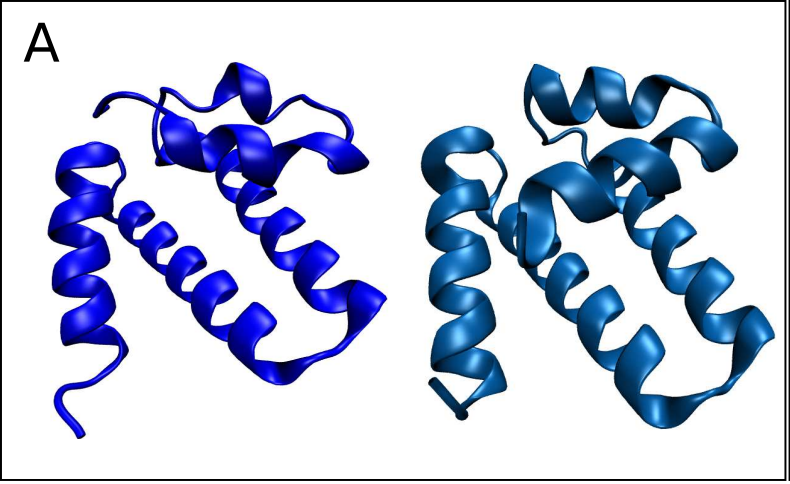
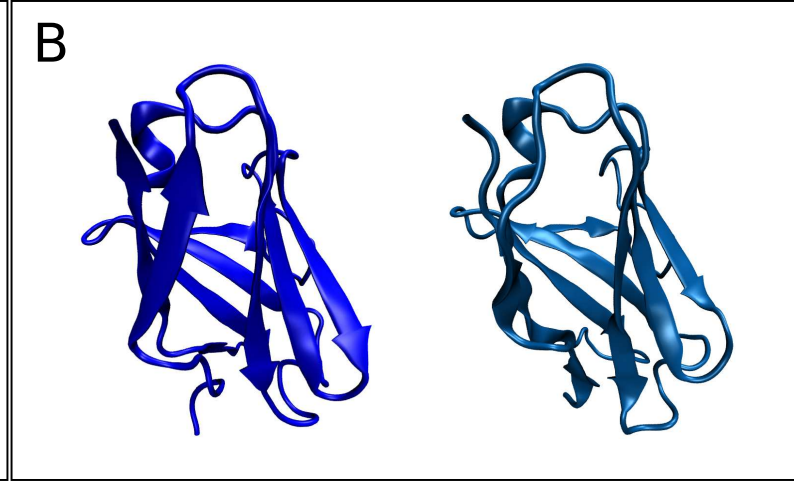
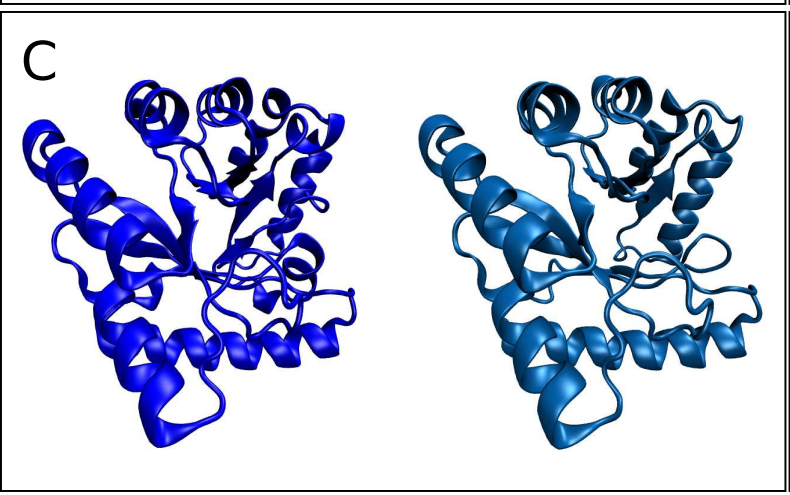
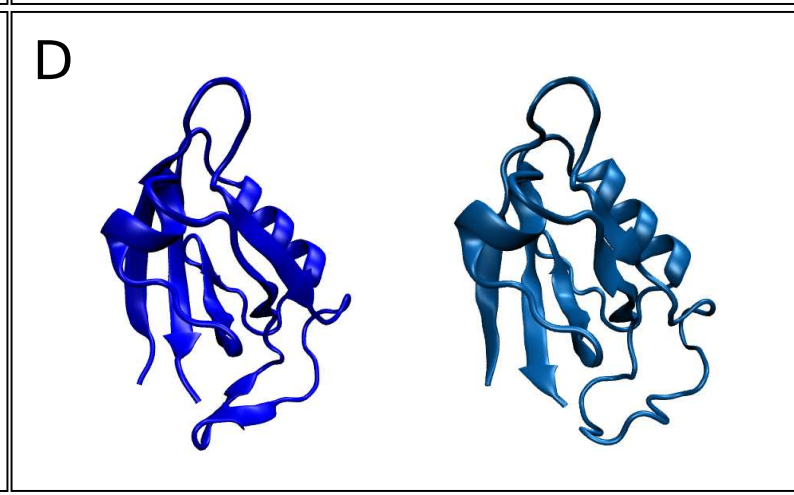

A
Accuracy on missing residues
Number of training data points
1e6
training
validation
B
Number of puzzles
Fraction of missing values filled correctly
oneshot (accuracy: 72.18%)
incremental (accuracy: 87.50%)
C
Number of puzzles
Fraction of missing values filled correctly
oneshot (accuracy: 83.37%)
incremental (accuracy: 97.62%)
D
Accuracy on missing residues
Number of training data points
1e8
training
validation
E
Number of sequences
Sequence identity to reference
oneshot (accuracy: 26.57%)
incremental (accuracy: 25.97%)
F
Number of sequences (normalized)
Sequence identity to reference
0% residues present
50% residues present
80% residues present

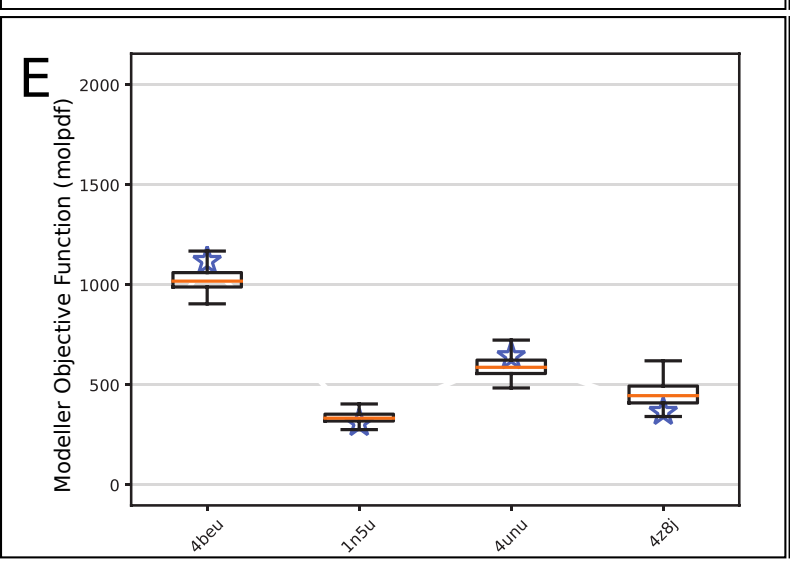
Figure 2


A


B


C


D


E
Modeller Objective Function (molpdf)
2000
1500
1000
500
0
4beu
1n5u
4unu
4z8j

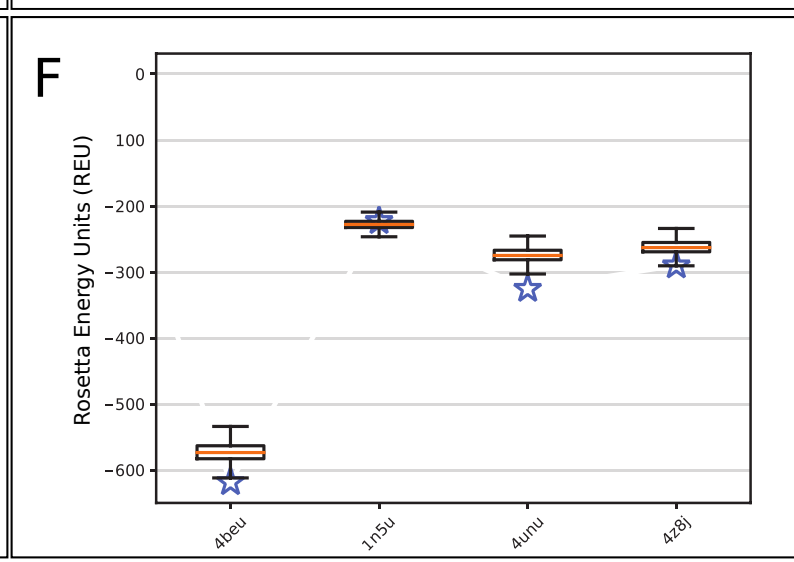

F
Rosetta Energy Units (REU)
0
100
−200
−300
−400
−500
−600
4beu
1n5u
4unu
4z8j

Figure 3

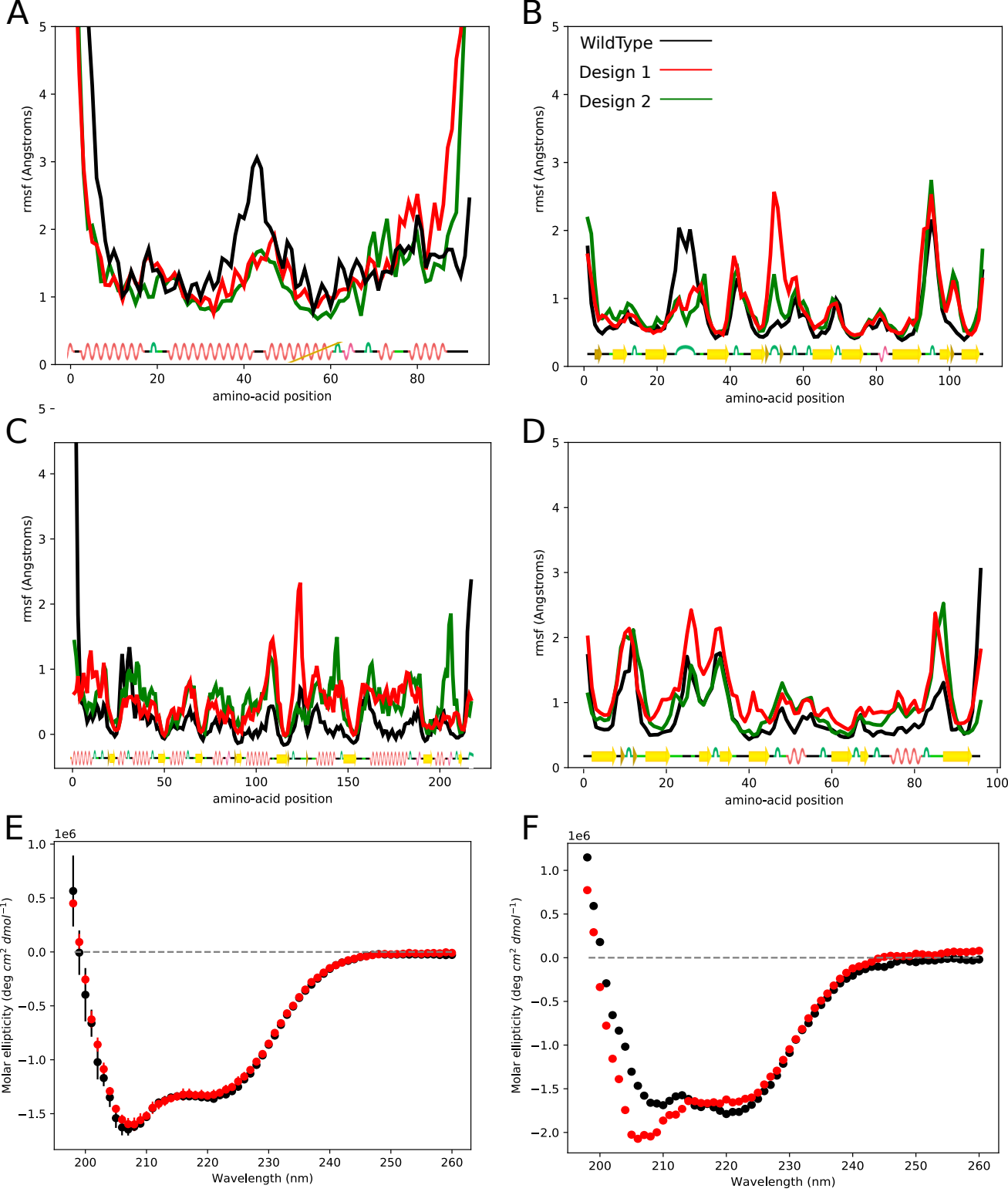

A
B
C
D
E
F
WildType
Design 1
Design 2
rmsf (Angstroms)
amino-acid position
1e6
Molar ellipticity (deg cm^2 $dmol^{-1}$)
Wavelength (nm)

Figure 4

Supplemental Materials

Online Methods

Data preparation

The websites which host the code and data that were used to generate those datasets are listed in Supp. Table S1. In brief, we downloaded from UniParc (*1*) a dataset of all protein sequences and corresponding domain definitions, and we extracted from this dataset a list of all unique Gene3D domains. We also processed the PDB database and extract the amino acid sequence and the distance matrix of every chain in every structure. For each amino acid, we recorded the distances of all other residues below a cutoff of 12 Angstrom in the distance matrix, taking into account the closest heavy atom (including the side chain). Finally, we attempted to find a structural template for every Gene3D domain sequence, and we transferred the adjacency matrix from the structural template to that sequence.

The end result of this process is a dataset of 72,464,122 sequences and adjacency matrices, clustered into 1,373 different Gene3D superfamilies. We split this dataset into a training subset, containing sequences of 1,029 Gene3D superfamilies, a validation subset, containing sequences of 172 Gene3D superfamilies, and a test subset, containing sequences of another 172 Gene3D superfamilies.

Network architecture

We used Sudoku to optimize the network architecture, including the dimensionality of the embedding space (162), the number of residual blocks (16), the edge convolution network (2-layer feed-forward network with 324 elements in the hidden layer), the use of batch normalization, as well as the choice of non-linearity (reLU). The architecture of the resulting graph neural network is presented in Figure 1. The network takes as input a set of node attributes, a set of edge indices, and optionally, a set of edge

attributes. The node and edge attributes are embedded into a 162-dimensional space, followed by the application of batch normalization (BatchNorm) (*2*) and a rectified linear unit (ReLU) (*3*) nonlinearity. The node and edge attributes, and the edge indices, are then passed into a graph convolution residual block, which produces a new set of node and edge attributes with the same dimensionality as the inputs but endowed with information about the immediate neighborhood of each node. The produced node and edge attributes are added to the inputs, and the resulting tensors are passed through a ReLU layer and into the subsequent graph convolution residual block, with four residual blocks being applied in total. The node attributes produced by the last residual block are passed through a rectified linear unit nonlinearity and into a final Linear layer, which maps the 162 features describing each node into an N-dimensional space, where N is the number of possible output values (i.e. 9 in the case of Sudoku; 20 in the case of protein design). The outputs from the Linear layer can optionally be passed through a softmax activation function in order to convert the raw output values into probabilities of a given node possessing a given label.

The primary component of the graph convolution residual block is the Edge Convolution (EdgeConv) layer (*4*), which for every edge in the graph, concatenates the edge attributes and the attributes of the interacting nodes and passes the resulting vector through a multi-layer perceptron (see Figure 1). The outputs of the multilayer perceptron form the new edge attributes, and summing over the edge attributes of each of the nodes produces the new node attributes. The node and edge attributes are normalized using a BatchNorm layer and are added to the input tensors, which completes the residual block.

Network training

The architecture and the hyperparameters of the graph neural network were optimized by training the network to solve Sudoku puzzles, which is a well-defined problem and for which predictions made by

the network can easily be verified. We treat Sudoku puzzles as graphs having 81 nodes, corresponding to squares on the Sudoku grid, and 1701 edges, corresponding to pairs of nodes that cannot be assigned the same number (see Figure 1). The node attributes correspond to the numbers entered into the squares, with an additional attribute to indicate that no number has been entered, the edge indices correspond to the 1701 pairs of nodes that are constrained such that they cannot have the same number, and the edge attributes are empty because all edges in the graph impose identical constraints. We generated 30 million solved Sudoku puzzles using the *sugen* program (*5*), which first generates a solved Sudoku grid using a backtracking grid filler algorithm, and then randomly removes numbers from that grid until it generates a Sudoku puzzle with a unique solution and the requested difficulty level. The graph neural network was trained to reconstruct the missing numbers in the Sudoku grid by minimizing the cross-entropy loss between predicted and actual values. Throughout training, we tracked the accuracy that the network achieves on the training dataset (Figure 2A, blue line) and on the validation dataset (Figure 2A, orange line), which contains 1000 puzzles which were excluded from the training dataset. After trying multiple different network architectures and different hyperparameters, we found that the highest validation accuracy is achieved using a graph neural network with 16 graph convolution residual blocks, 162-dimensional node and edge attribute embeddings, EdgeConv layers containing a two-layer feedforward network with 256 neurons in the hidden layer, ReLU nonlinearities, and batch normalization that is applied before, rather than after, the outputs of the residual blocks are added to the input channels. The accuracy that the network achieves on the training dataset is close to what it achieves on the validation dataset, indicating that the network is not overfitting (Figure 2A-B). The test accuracy of the trained network was evaluated using a dataset comprised of 30 curated Sudoku puzzles collected from http://1sudoku.com (*6*, *7*) (Figure 2C).

After optimizing the network architecture for solving Sudoku puzzles, we applied the same network to protein design, which is a less well-defined problem than Sudoku and for which the accuracy of predictions is more difficult to ascertain. We treat proteins as graphs, where nodes correspond to the

individual amino acids and edges correspond to shortest distances between pairs of amino acids, considering only those pairs of amino acids that are within 12 Å of one another. The node attributes specify the amino acid type, with an additional flag to indicate that the amino acid type is not known, while the edge attributes include the shortest distance between each pair of amino acids in Cartesian space and the number of residues separating the pair of amino acids along the protein chain. For each domain sequence in our training dataset, we marked approximately half of the amino acids in the sequence as missing, and we trained the network to reconstruct the masked amino acids by minimizing the cross-entropy loss between predicted and actual values (Figure 2D). The edge attributes corresponding to each training example were passed into the network without modification. Throughout training, we tracked the accuracy with which the network can reconstruct amino acid sequences from our training and validation datasets where, in both cases, 50% of the amino acids were marked as missing (Figure 2D), and training was terminated once the validation accuracy appeared to have reached a plateau. We evaluated the trained network by examining how well it can reconstruct sequences from our test dataset using edge attributes alone and without any sequence information (Figure 2E), by measuring the correlation between differences in network outputs for wild type and mutant residues and the change in the Gibbs free energy of folding (ΔΔG) associated with mutations (*8*) (Supp. Figure S1A). In the case of sequence reconstruction, we observe a bimodal distribution in sequence identities between generated and reference sequences, with a smaller peak around 7% sequence identity, corresponding to generated sequences that share little or no homology with the reference sequences, and a larger peak around 37% sequence identity, corresponding to generated sequences that share substantial homology with the reference sequences (Figure 2E). While the fraction of designs falling into the second category increases when we provide the network with some of the original residues (Figure 2F), there are still some cases where the generated and the reference sequences share little homology despite 80% of the original residues being given as input (Figure 2F). In the case of ΔΔG prediction, we found that the network can predict the ΔΔG associated with

mutations with a Spearman's correlation coefficient of 0.426, despite never having been trained specifically for this task (Supp. Figure 1A).

Solving puzzles and generation of novel sequences

We used two different strategies to generate or complete CSPs: *one-shot generation* and *incremental generation*. In one-shot generation, we pass the input with the missing labels through the network only once, for every node accepting the label assigned the highest probability by the network. In incremental generation, we pass the input through the network once for every missing label. At each iteration, we accept the label for which the network has the most confident prediction, and we treat that label as known in the subsequent iteration.

In order to generate a library of highly-probable solutions, we used a strategy similar to A* search (*9*), which allows us to achieve a compromise between the accuracy provided by exhaustive breadth-first search and the speed provided by greedy search. As with the *incremental generation* strategy described above, each time we pass the input through the network, we select a single node for which the network is making the most confident predictions. However, instead of accepting for that node the label with the highest probability, we accept all labels that have a probability above a threshold of 0.05, and we add the inputs partially filled with the accepted labels to a priority queue. Next, we extract from the priority queue a partially-filled input with the highest priority, extend it by one iteration, and place the results back onto the priority queue. This process is repeated until we generate the desired number of solutions.

The priority P_j , which is assigned to every partially-filled input placed on the priority queue, is defined by Equation 1. Here, p_i is the probability assigned by the network to the label accepted at iteration i, and p^* is the expected maximum probability that the network will assign to labels that are accepted in the subsequent iterations. p^* is a tunable parameter which controls how much the algorithm resembles greedy search vs. exhaustive breadth-first search. If p^* is set to 0, the algorithm is equivalent to the

incremental generation algorithm described above, where we immediately accept the most probable output from the previous iteration. On the other hand, if p^* is set to 1, the algorithm is equivalent to exhaustive breadth-first search, where we use the network to extend every possible label at a given iteration before moving on to the next iteration.

$$P_j = \sum_{i=1}^{j} log(p_i) + \sum_{j}^{N} log(p^*) \quad (1)$$

On a single Titan Xp GPU, generation of 100,000 sequences for an average protein domain takes about 30 minutes.

Molecular dynamics

All water and ions atoms were removed from the structure with PDB codes 1N5U, 4Z8J, 4UNU, and 1OC7, corresponding to an all-α protein, a mainly-β protein, an all-β protein and a mix-αβ protein, respectively. The structural models for the generated sequences were produced using Modeller, with the PDB files described above serving as templates. Using TLEAP in AMBER16 (*10*) and AMBER ff14SB force field (*11*), the structures were solvated by adding a 12 nm^3 box of explicit water molecules, TIP3P. Next, Na+ and Cl- counter-ions were added to neutralize the overall system net charge, and periodic boundary conditions were applied. Following this, the structures were minimized, equilibrated, heated over 800 ps to 300 K, and the positional restraints were gradually removed. Bonds to hydrogen were constrained using SHAKE (*12*) and a 2 fs time step was used. The particle mesh Ewald (*13*) algorithm was used to treat long-range interactions.

Experimental validation

Protein purification

All constructs were synthetized by Invitrogen GeneArt Synthesis (ThermoFisher) as Gene Strings. The constructs design included flanking BamHI and HindIII restriction sites for subcloning into a N-terminal 6xHis-tagged pRSet A vector. All genes were codon optimized for expression in *E. coli* using the GeneArt suite.

The pRSET A constructs were transformed into chemically competent E. coli OverExpress C41(DE3) cells (Lucigen) by heat shock and plated on LB-Carbenicillin plates. Colonies were grown in 15 ml of 2xYT media containing Carbenicillin (50 µg/mL) at 37 °C, 220 rpm until the optical density (O.D.) at 600 nm reached 0.6. Cultures were then induced with IPTG (0.5 mM) for 3h at 37°C. Cells were pelleted by centrifugation at 3000 g (4 °C, 10 min) and resuspended in 1 ml of BugBuster Master Mix (Millipore). The lysate mixtures were incubated for 20 min in a rotating shaker and the insoluble fraction was removed by centrifugation at 16000 g for 1 min. For a 15 ml preparation, 200 µl of Ni-NTA Agarose (QIAGEN) were pre-washed in Phosphate-buffered saline (PBS) and added to the supernatant from the cell lysate for 20 min at 4 °C in batch. The bound beads were washed thrice with PBS (1 mL) containing 30 mM of imidazole to prevent nonspecific interaction of lysate proteins. Proteins were eluted using PBS buffer with 500 mM imidazole and dialyzed against PBS (*14*).

Protein concentration was calculated using the Pierce BCA Protein Assay Kit (ThermoFisher). The concentration of 4beu_Design was corroborated by absorbance at 205 nm as described by Anthis et. al. (*15*). 1n5u and 1n5u_Design and 4beu eluted solubly at concentrations around 200-400 ug/ml. 4beu_Design presented a limit of solubility at around 15 ug/ml. Sample purity was assessed through Coomassie staining on SDS-PAGE with 4–20% Mini-PROTEAN TGX Precast Protein Gels, 10-well (Bio-Rad) and Mass Spectrometry (ESI).

Circular dichroism

All CD measurements were made on a Jasco J-810 CD spectrometer in 1 mm Quartz Glass cuvette for high performance (QS) (Hellma Analytics) with the exception of 4beu_Design where a 10 mm Quartz

Glass cuvette (QS) (Hellma Analytics) was preferred. 1n5u and 1n5u_Design were analysed in PBS whereas 4beu and 4beu_Design were analysed in 10 mM Na-Phosphate, pH 8. The CD spectrum was measured between 198 nm and 260 nm wavelengths using a 1 nm of bandwidth and 1 nm intervals collected at 50 nm/min; each reading was repeated then times. All measurements were taken at 20 °C.

Code availability

The source code for ProteinSolver will be freely available at https://gitlab.com/ostrokach/proteinsolver. The network was implemented in the Python programming language using PyTorch (*16*) and PyTorch Geometric (*17*) libraries. The repository also includes Jupyter notebooks that can be used to reproduce all the figures presented in this manuscript.

Supplemental tables

Table S1. Websites providing code and data for every step of the data processing pipeline outlined in Figure S8.

Name	Website
uniparc_xml_parser	https://gitlab.com/kimlab/uniparc_xml_parser
uniparc_all.xml.gz	http://www.uniprot.org/downloads
uniparc	https://gitlab.com/datapkg/uniparc
uniparc-domain	https://gitlab.com/datapkg/uniparc-domain
kmbio	https://gitlab.com/kimlab/kmbio
PDB	ftp://ftp.wwpdb.org/pub/pdb/data/structures/divided/
pdb-analysis	https://gitlab.com/datapkg/pdb-analysis
uniparc-domain-wstructure	https://gitlab.com/datapkg/uniparc-domain-wstructure
proteinsolver	https://gitlab.com/kimlab/proteinsolver

Table S2. Time performance comparison generating N designs of ProteinSolver and Rosetta protocols (detailed on Table S3) over different size structures. Average time in seconds of 7 independent runs over a single core in a intel's i7-5820K CPU @ 3.30GHz. Enclosed in square brackets the lowest REU produced. Lower boundary time estimation for Rosetta Optimized for N=1,000 was calculated by linear projection of N=1.

PDB	N	# Residues	ProteinSolver	Rosetta optimized
5VID	1	39	2.5 [-90.87]	577 [-57.94]
1N5U	1	92	20.9 [-234.53]	1,385 [-377.26]
4Z8J	1	96	26.5 [-275.17]	1,742 [-161.325]
4UNU	1	109	29.4 [-269.09]	1,963 [-174.46]
4BEU	1	217	150.1 [-580.37]	4,547 [-375.83]

PDB	N	# Residues	ProteinSolver	Rosetta default	Rosetta optimized
5VID	1,000	39	27 [-119.15]	10,651 [-54.75]	>280,000
1N5U	1,000	92	459 [-250.18]	28,843 [-133.22]	>690,000
4Z8J	1,000	96	87 [-291.86]	34,068 [-152.78]	>870,000
4UNU	1,000	109	95 [-298.21]	39,424 [-151.95]	>980,000
4BEU	1,000	217	254 [-611.46]	100,795 [-346.98]	>2,200,000

Table S3. System commands used to run Rosetta's standard fix backbone design (upper left) and optimized protocol (upper right), Rosetta's cartesian protocol (bottom left), and Rosetta's relax protocol (bottom right)

Rosetta's default fix backbone protocol	Rosetta's optimized fix backbone protocol
$ fixbb.static.linuxgccrelease \ -in:file:s '{structure_file}' \ -database '{rosetta_db}' \ -dun10 \ -scorefile 'default_{structure_file}' \ -out:suffix _def	$ fixbb.static.linuxgccreleas \ -in:file:s '{structure_file}' \ -database '{rosetta_db}' \ -dun10 \ -ex1 \ -ex2 \ -extrachi_cutoff 0 \ -minimize_sidechains True \ -scorefile 'optm_{structure_file}' \ -out:suffix _optm

Rosetta's cartesian_ddg protocol	Rosetta's relax protocol
$ cartesian_ddg.static.linuxgccrelease \ -in:file:s '{structure_file}' \ -in::file::fullatom \ -database '{rosetta_db}' \ -ignore_unrecognized_res true \ -ignore_zero_occupancy false \ -fa_max_dis 9.0 \ -ddg::cartesian \ -ddg::mut_file '{mutation_file}' \ -ddg::iterations 3 \ -ddg::dump_pdbs true \ -ddg::suppress_checkpointing true \ -ddg::mean true \ -ddg::min true \ -ddg::output_silent true \ -bbnbr 1 \ -beta_nov16_cart	$ relax.linuxgccrelease -in:file:s '{structure_file}' \ -in:file:fullatom \ -scorefile '{structure_file}'.sc \ -relax:quick

Supplemental figures

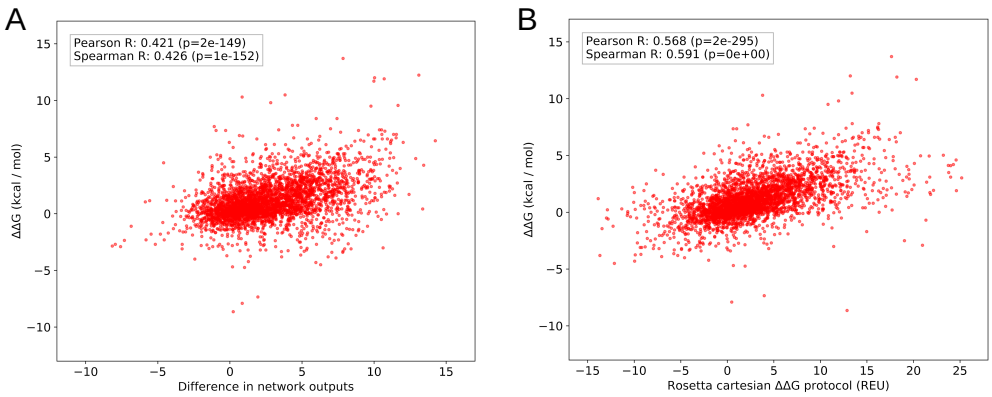


Figure S1. Correlations between changes in the Gibbs free energy of folding associated with mutations (ΔΔG) for Protherm Dataset [7]. **(A)** Predictions made by the ProteinSolver network. (B) Predictions made by Rosetta's Cartesian protocols and the *betanov15* energy functions, details are provided for reference in Table S3.

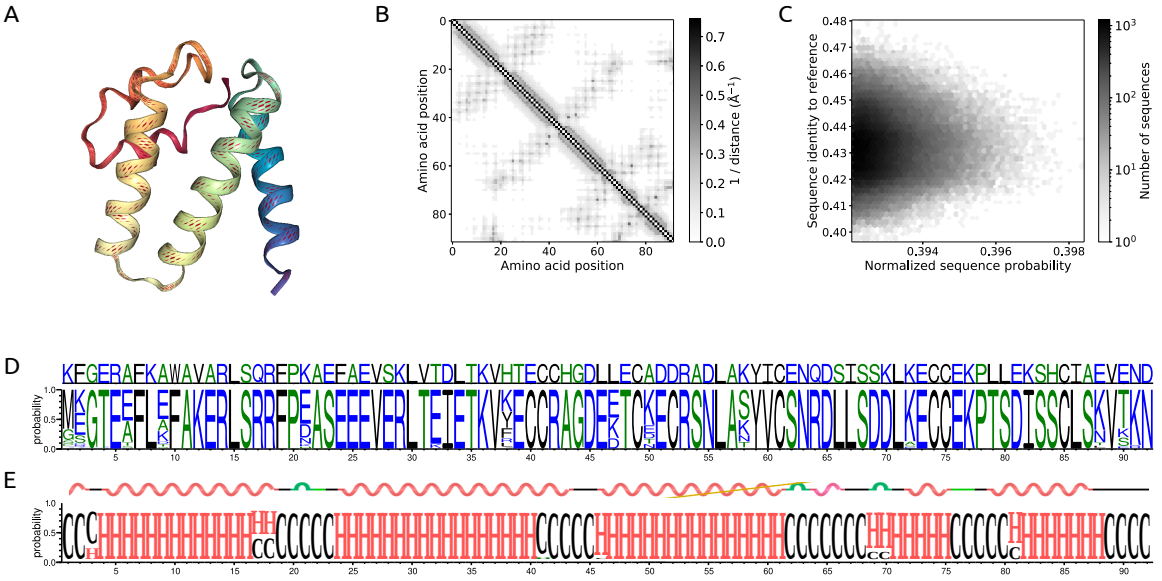


Figure S2. Generating sequences that fold into a protein resembling the structure of serum albumin, which is mostly comprised of alpha helices. **(A)** A representative structure of the serum albumin domain, extracted from chain A of the PDB structure 1n5u. **(B)** Amino acid contact map corresponding to the representative structure. **(C)** A diagram showing the probability assigned by the network to each of the 200,000 generated sequences, normalized by the length of the sequence, versus the sequence identity between the generated sequences and the reference sequence. **(D)** Sequence LOGO of the 200,000 generated sequences. The amino acid sequence of the reference structure is displayed above the LOGO. **(E)** Secondary Structure Prediction performed by Psipred on 20.000 generated sequences. The secondary structure of the reference structure is displayed above the LOGO.

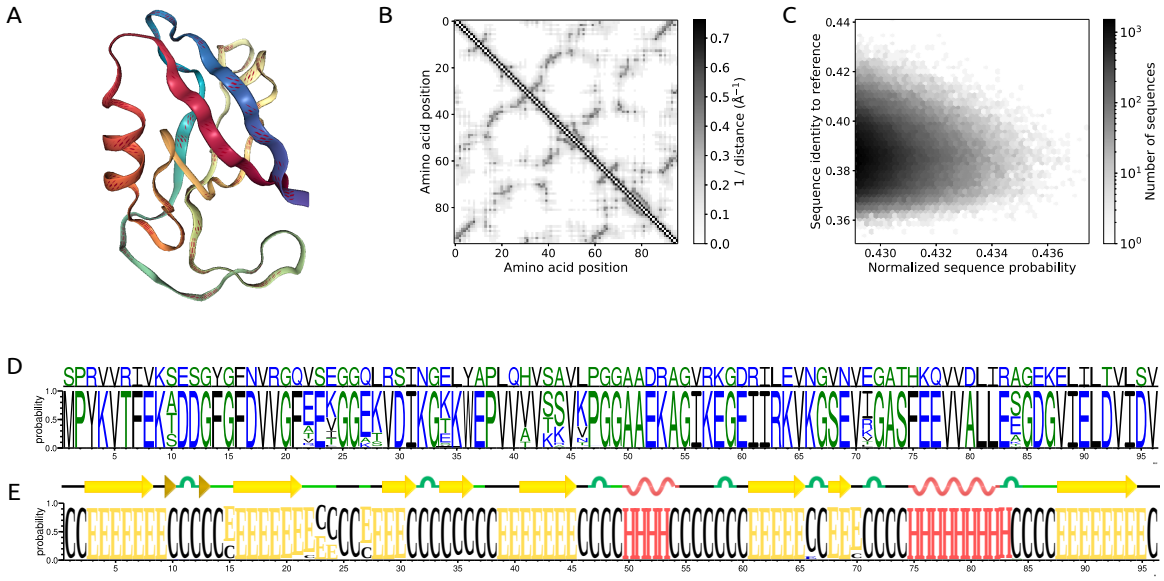


Figure S3. Generating sequences that fold into a protein resembling the structure of PDZ3 domain, which is mostly comprised of beta sheets. **(A)** A representative structure of the PDZ3 domain, extracted from chain A of the PDB structure 4z8j. **(B)** Amino acid contact map corresponding to the representative structure. **(C)** A diagram showing the probability assigned by the network to each of the 200,000 generated sequences, normalized by the length of the sequence, versus the sequence identity between the generated sequences and the reference sequence. **(D)** Sequence LOGO of the 200,000 generated sequences. The amino acid sequence of the reference structure is displayed above the LOGO. **(E)** Secondary Structure Prediction performed by Psipred on 20.000 generated sequences. The secondary structure of the reference structure is displayed above the LOGO.

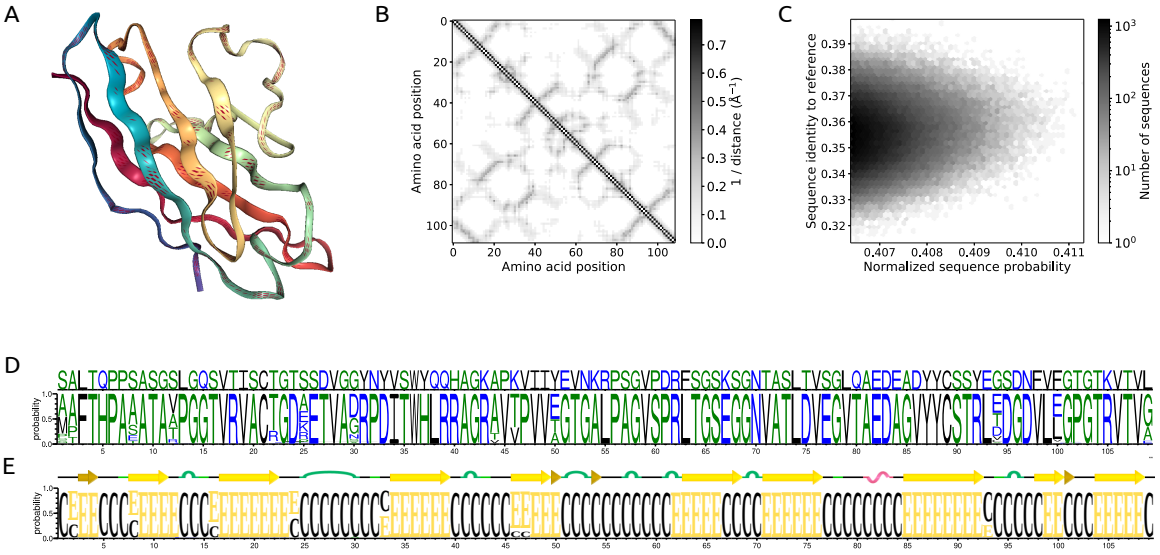


Figure S4. Generating sequences that fold into a protein resembling the structure of immunoglobulin, a protein containing mainly beta sheets. **(A)** A representative structure of the immunoglobulin domain, extracted from chain A of the PDB structure 4unu. **(B)** Amino acid contact map corresponding to the representative structure. **(C)** A diagram showing the probability assigned by the network to each of the 200,000 generated sequences, normalized by the length of the sequence, versus the sequence identity between the generated sequences and the reference sequence. **(D)** Sequence LOGO of the 200,000 generated sequences. The amino acid sequence of the reference structure is displayed above the LOGO. **(E)** Secondary Structure Prediction performed by Psipred on 20.000 generated sequences. The secondary structure of the reference structure is displayed above the LOGO.

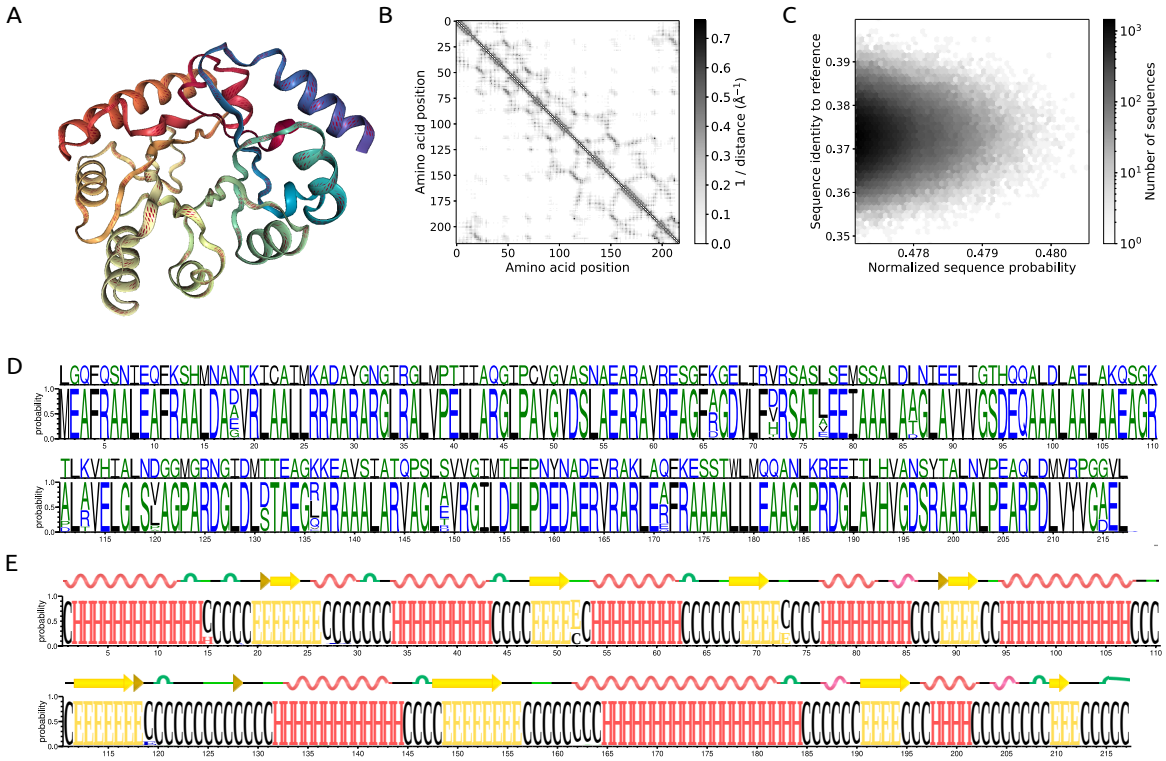


Figure S5. Generating sequences that fold into a protein resembling the structure of alanine racemase, a protein containing both alpha helices and beta sheets. **(A)** A representative structure of the alanine racemase domain, extracted from chain A of the PDB structure 4beu. **(B)** Amino acid contact map corresponding to the representative structure. **(C)** A diagram showing the probability assigned by the network to each of the 200,000 generated sequences, normalized by the length of the sequence, versus the sequence identity between the generated sequences and the reference sequence. **(D)** Sequence LOGO of the 200,000 generated sequences. The amino acid sequence of the reference structure is displayed above the LOGO. **(E)** Secondary Structure Prediction performed by Psipred on 20.000 generated sequences. The secondary structure of the reference structure is displayed above the LOGO.

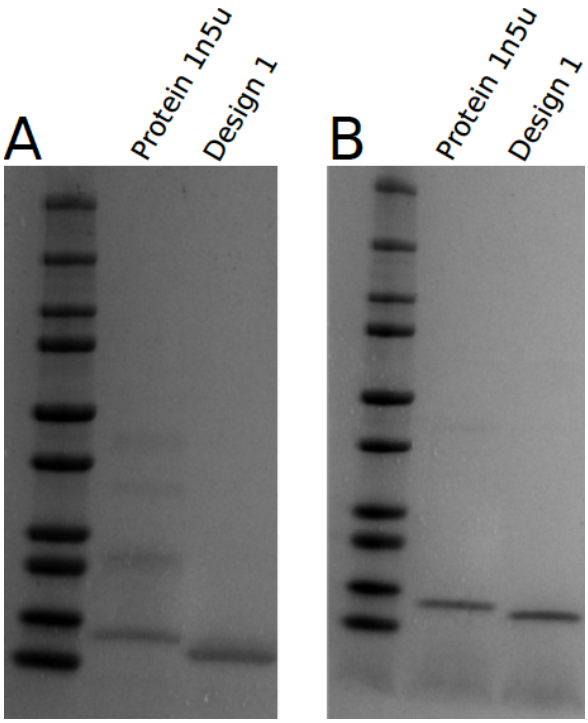


Figure S6. The designed albumin sequence and its natural counterpart (pdb-id: 1n5u) were analyzed by SDS-PAGE in **(A)** oxidizing and **(B)** reducing conditions (1 mM DTT). The natural protein, with an unpaired cysteine, forms high MW species not present in reducing conditions. The *Design 1* is only represented as a single band in both environments.

Interestingly, the chosen sequence from the albumin template contains 4 pairs of potential disulfide bonds, whereas it is known that the albumin template used has only 3 of these bonds (PDB 1n5u). The designed 1n5u run in an SDS-PAGE in oxidising conditions as a single band. Furthermore, the mass spectrometry analysis by electrospray (ESI) of the molecular weight (MW) of designed 1n5u showed a loss of 8 Da against the theoretical MW, consistent with the loss of 8 protons. All together this indicates that a new disulfide bond not present in the albumin structural template was efficiently inserted into the designed sequence.

Supplemental References

1. E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, G. M. Church, Unified rational protein engineering with sequence-only deep representation learning. *bioRxiv*, 589333 (2019).
2. S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ArXiv150203167 Cs* (2015) (available at http://arxiv.org/abs/1502.03167).
3. V. Nair, G. E. Hinton, in *Proceedings of the 27th International Conference on International Conference on Machine Learning* (Omnipress, USA, 2010; http://dl.acm.org/citation.cfm?id=3104322.3104425), *ICML'10*, pp. 807–814.
4. Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, J. M. Solomon, Dynamic Graph CNN for Learning on Point Clouds. *ArXiv180107829 Cs* (2018) (available at http://arxiv.org/abs/1801.07829).
5. D. Beer, *sugen* (2011).
6. Sudoku free online to play and print - 1sudoku.com. *Site Sudoku Free Online Print*, (available at https://1sudoku.com/).
7. K. Park, *Can Neural Networks Crack Sudoku?* (2019; https://github.com/Kyubyong/sudoku).
8. K. A. Bava, M. M. Gromiha, H. Uedaira, K. Kitajima, A. Sarai, ProTherm, version 4.0: thermodynamic database for proteins and mutants. *Nucleic Acids Res.* **32**, D120–D121 (2004).
9. A. R. Leach, A. P. Lemon, Exploring the conformational space of protein side chains using dead-end elimination and the A* algorithm. *Proteins Struct. Funct. Bioinforma.* **33**, 227–239 (1998).
10. D. A. Case, T. E. Cheatham III, T. Darden, H. Gohlke, R. Luo, K. M. Merz Jr., A. Onufriev, C. Simmerling, B. Wang, R. J. Woods, The Amber biomolecular simulation programs. *J. Comput. Chem.* **26**, 1668–1688 (2005).
11. J. A. Maier, C. Martinez, K. Kasavajhala, L. Wickstrom, K. E. Hauser, C. Simmerling, ff14SB: Improving the Accuracy of Protein Side Chain and Backbone Parameters from ff99SB. *J. Chem. Theory Comput.* **11**, 3696–3713 (2015).
12. J.-P. Ryckaert, G. Ciccotti, H. J. C. Berendsen, Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *J. Comput. Phys.* **23**, 327–341 (1977).
13. A. Toukmaji, C. Sagui, J. Board, T. Darden, Efficient particle-mesh Ewald based approach to fixed and induced dipolar interactions. *J. Chem. Phys.* **113**, 10913–10927 (2000).
14. A. Perez-Riba, L. S. Itzhaki, A method for rapid high-throughput biophysical analysis of proteins. *Sci. Rep.* **7**, 9071–9071 (2017).
15. N. J. Anthis, G. M. Clore, Sequence-specific determination of protein and peptide concentrations by absorbance at 205 nm. *Protein Sci.* **22**, 851–858 (2013).
16. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in PyTorch (2017) (available at https://openreview.net/forum?id=BJJsrmfCZ).
17. M. Fey, J. E. Lenssen, Fast Graph Representation Learning with PyTorch Geometric. *ArXiv190302428 Cs Stat* (2019) (available at http://arxiv.org/abs/1903.02428).