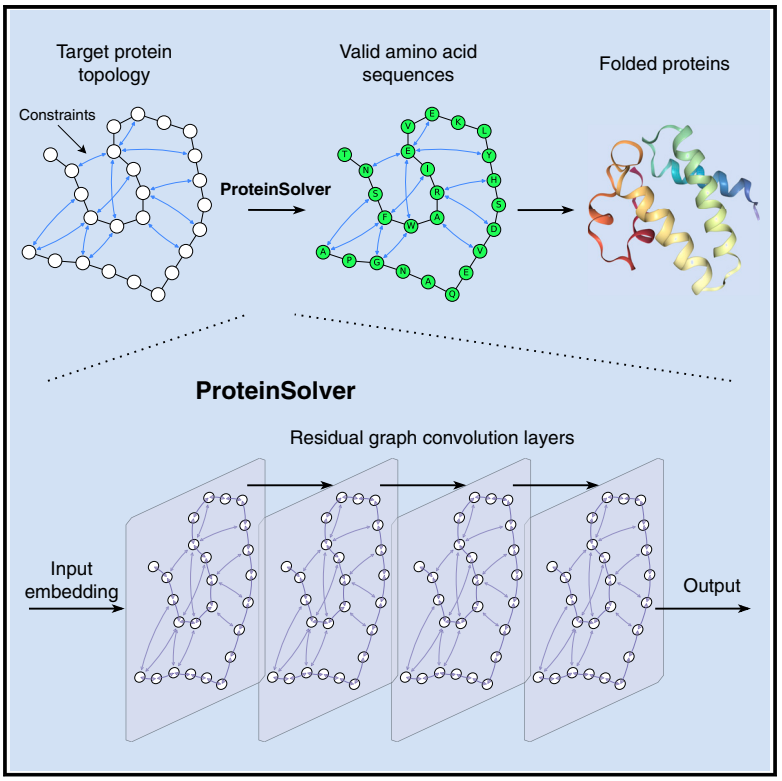
Cell Systems

Brief Report

Fast and Flexible Protein Design Using Deep Graph Neural Networks

Graphical Abstract



Authors

Alexey Strokach, David Becerra, Carles Corbi-Verge, Albert Perez-Riba, Philip M. Kim

Correspondence

pi@kimlab.org

In Brief

Strokach et al. developed ProteinSolver, a graph convolutional neural network trained on the PDB and sequences in UniParc to reconstruct amino acid sequences that adhere to constraints imposed by protein topologies. It can generate new sequences that fold into predetermined shapes and predict effects of mutations on stability.

Highlights

- Graph neural network generates new proteins with predetermined topologies
- Probabilities assigned to individual amino acids correlate with stability of mutants
- Probabilities assigned to amino acid sequences correlate with stability of designs
- Orders of magnitude faster than traditional approaches

Strokach et al., 2020, Cell Systems *11*, 402–411
October 21, 2020 © 2020 The Author(s). Published by Elsevier Inc.
https://doi.org/10.1016/j.cels.2020.08.016

CellPress

CellPress
OPEN ACCESS

Cell Systems

Brief Report

Fast and Flexible Protein Design Using Deep Graph Neural Networks

Alexey Strokach,[1] David Becerra,[2] Carles Corbi-Verge,[2] Albert Perez-Riba,[2] and Philip M. Kim[1,2,3,4,*]
[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
[4]Lead Contact
*Correspondence: pi@kimlab.org
https://doi.org/10.1016/j.cels.2020.08.016

SUMMARY

Protein structure and function is determined by the arrangement of the linear sequence of amino acids in 3D space. We show that a deep graph neural network, ProteinSolver, can precisely design sequences that fold into a predetermined shape by phrasing this challenge as a constraint satisfaction problem (CSP), akin to Sudoku puzzles. We trained ProteinSolver on over 70,000,000 real protein sequences corresponding to over 80,000 structures. We show that our method rapidly designs new protein sequences and benchmark them *in silico* using energy-based scores, molecular dynamics, and structure prediction methods. As a proof-of-principle validation, we use ProteinSolver to generate sequences that match the structure of serum albumin, then synthesize the top-scoring design and validate it *in vitro* using circular dichroism. ProteinSolver is freely available at http://design.proteinsolver.org and https://gitlab.com/ostrokach/proteinsolver. A record of this paper's transparent peer review process is included in the Supplemental Information.

INTRODUCTION

Protein structure and function emerges from the specific geometric arrangement of their linear sequence of amino acids, commonly referred to as a fold. Engineering novel protein sequences has a broad variety of uses, including academic research, industrial process engineering (Bommarius and Paye, 2013), and most notably, protein-based therapeutics, which are now a very important class of drugs (Chevalier et al., 2017; Fischman and Ofran, 2018).

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 challenging task. Conventionally, a sampling technique such as Markov-chain Monte Carlo is used to generate sequences optimized with respect to a force field or statistical potential (Chevalier et al., 2017; Shultis et al., 2019; Sun and Kim, 2017). Limitations of those methods include the relatively low accuracy of existing force fields (Khan and Vihinen, 2010; Kroncke et al., 2016) 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 techniques (Rocklin et al., 2017; Sun et al., 2016), those approaches remain reliant on labor-intensive experiments.

Here, we introduce a methodology for protein design that overcomes the limitations of traditional methods by combining a classic idea with the expressive power of graph neural networks (Battaglia et al., 2018). 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 interacting amino acids are favorable and compatible with the fold. To overcome previous difficulties with phrasing protein design as a CSP (Dal Palù et al., 2004; Ullah and Steinhöfel, 2010), we elucidate the rules governing constraints using deep learning. Such methods have been applied to a vast diversity of fields with impressive results (Kocić et al., 2019; Silver et al., 2017; Wainberg et al., 2018), 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 (Dawson et al., 2017). 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 (Brookes et al., 2019; Ingraham et al., 2019; Li et al., 2014; O'Connell et al., 2018; Wang et al., 2018a). 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.

 © 2020 The Author(s). Published by Elsevier Inc.
This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

CellPress
OPEN ACCESS

The distance matrix is commonly used to represent protein folds (Senior et al., 2020). It is an NxN matrix consisting of the distances between residues, optionally restricted to only interacting pairs of residues that are within a certain distance of one another. The distance matrix can be thought of as placing constraints on pairs of residues, such that the forces governing the interaction between those residues are not violated (e.g., interactions between residues with the same charge or divergent hydropathicities are usually not well tolerated). A given protein structure, corresponding to a single distance matrix, can be formed by many different homologous sequences, and those sequences all satisfy the constraints imposed by the distance matrix. Such solutions to this CSP are given to us by evolution and are available in sequence repositories such as Pfam (Punta et al., 2012) or Gene3D (Lewis et al., 2017). While such CSPs in the specific case of a given protein fold can be found by comparing sequences from one of such repositories and can be captured as Hidden Markov models (HMMs) or position weight matrices (PWMs), often represented as sequence logos, the general version of the CSP—connecting any given protein fold or distance matrix with a set of sequences—has not been solved. Here, we use a graph neural network, denoted ProteinSolver, for this purpose. The graph in this case is made up of nodes, corresponding to amino acids, and edges between those nodes, corresponding to the spatial interactions between amino acids, as represented in the distance matrix. The edges thus represent the constraints that are imposed on the node properties (amino acid types).

Notably, we show that a ProteinSolver network trained to elucidate the rules governing the CSP of protein folding by reconstructing masked sequences can be successful in generating novel protein sequences for a predetermined fold. Previous approaches to protein design were hampered by the enormous computational complexity, in particular when taking into account backbone flexibility. Our approach sidesteps this problem and delivers plausible designs for a wide range of folds. In the future, similar algorithms could help with designing completely novel folds. Furthermore, as a neural network approach, its evaluation is many orders of magnitude faster than classical approaches and should enable the exploration of vastly more potential backbones.

Finally, we present a web server which allows users to run a trained ProteinSolver model to generate sequences matching the geometries of their own reference proteins. We hope that this web server will lower the barrier to entry for protein design and will facilitate the generation of many novel proteins. The web server is freely available at: http://design.proteinsolver.org.

RESULTS

Network Architecture

As there had been little previous work in using neural networks to solve CSPs (Palm et al., 2017; Prates et al., 2018), we first had to devise a network architecture that would be well suited for this problem. In order to facilitate this search, we focused on designing a neural network capable of solving Sudoku puzzles, which is a well-defined CSP (25) for which predictions made by the network can easily be verified. It is analogous to protein design with numbers instead of amino acid letters and both long- and short-range constraints; the main difference is that it has a defined single solution (also see below). We treat Sudoku puzzles as graphs having 81 nodes, corresponding to squares on the Sudoku grid, and 1,701 edges, corresponding to pairs of nodes that cannot be assigned the same number (Figure 1B). 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 1,701 pairs of nodes that are constrained such that they cannot have the same number, and the edge attributes are all the same value because all edges in the graph impose identical constraints. We generated 30 million solved Sudoku puzzles using the sugen program (Beer, 2011), 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 at the requested difficulty level. Neural networks with different architectures were 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 those networks achieve on the training dataset (Figure S1A, blue line) and on the validation dataset (Figure S1A, orange line), which contains 1,000 puzzles that were excluded from the training dataset.

After a broad scan over different neural network architectures that we conceived for this problem, we converged on the "ProteinSolver" graph neural network architecture presented in Figure 1A. The inputs to the network are a set of node attributes and a set of edge attributes describing interactions between pairs of nodes. The node and edge attributes are embedded in an m-dimensional space using linear transformations or a multi-layer perceptron. The resulting node and edge attribute embeddings are passed through N residual edge convolution and aggregation (ETA) blocks. In the convolution step, we update the edge attributes using a modified version of the edge convolution layer (Wang et al., 2018b), which takes as input a concatenation of node and edge attributes and returns an update to the edge attributes. In our work, g_θ is a multi-layer perceptron, although other neural network architectures are possible. In the aggregation step, we update node attributes using an aggregation over transformed edge attributes incident on every node. In our work, h_θ is a learned linear transformation, although other neural network architectures, including attention layers (Vaswani et al., 2017), are possible.

In the case of Sudoku, the fully trained network with optimized hyperparameters predicts correctly 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 (Figure S1B). Similar accuracy is achieved on an independent test set containing puzzles from an online Sudoku puzzle provider (Figure S1C) (Sudoku, 2020).

Reconstructing and Evaluating Protein Sequences

After optimizing the general network architecture for the well-defined problem of solving Sudoku puzzles, we applied a similar network to protein design, which is a less well-defined problem than Sudoku and for which the accuracy of predictions is more

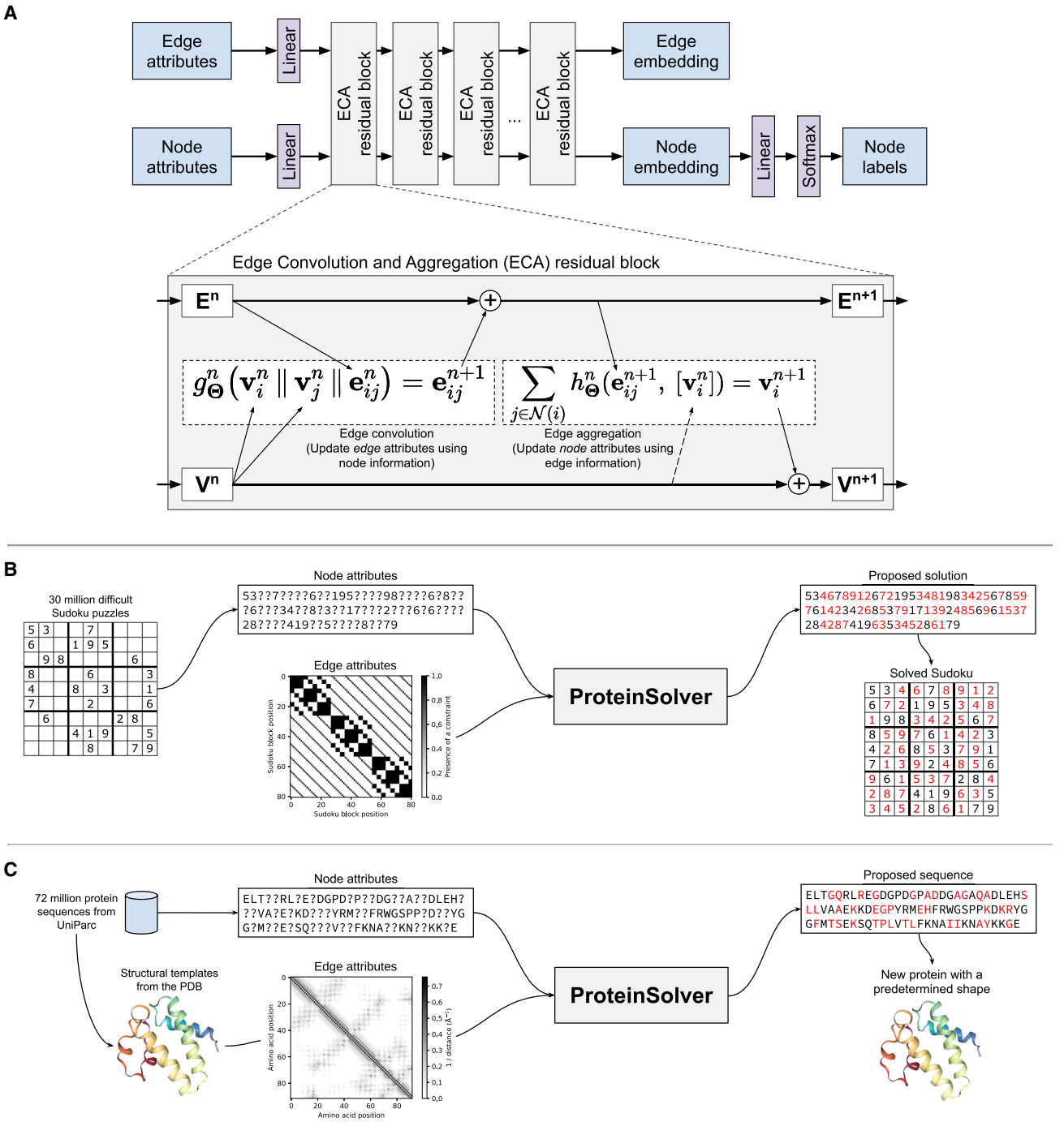


Figure 1. Graph Convolutional Neural Network Used by ProteinSolver to Assign Node Labels that Satisfy the Provided Node and Edge Constraints

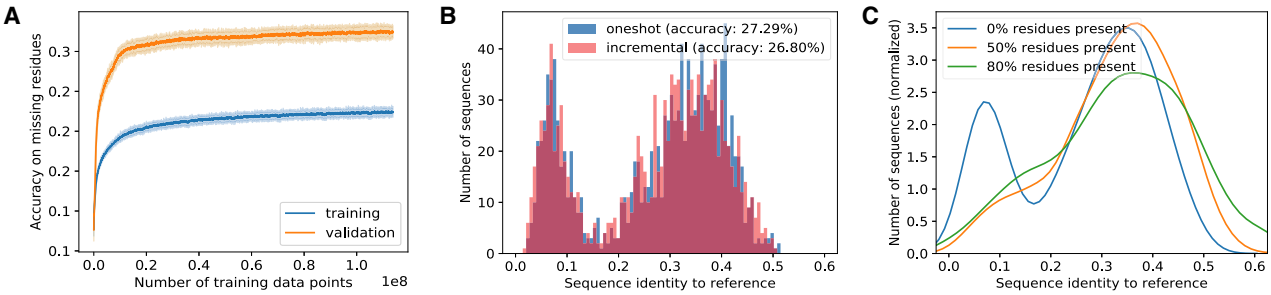
(A) ProteinSolver network architecture.

(B) Training a ProteinSolver network to solve Sudoku puzzles. Node attributes encode the numbers provided in the starting Sudoku grid. Edge attributes encode the presence of constraints (depicted with black in the edge attributes matrix) or the absence of constraints (depicted with white in the edge attributes matrix) between pairs of nodes (i.e., that a given pair of nodes cannot be assigned the same number).

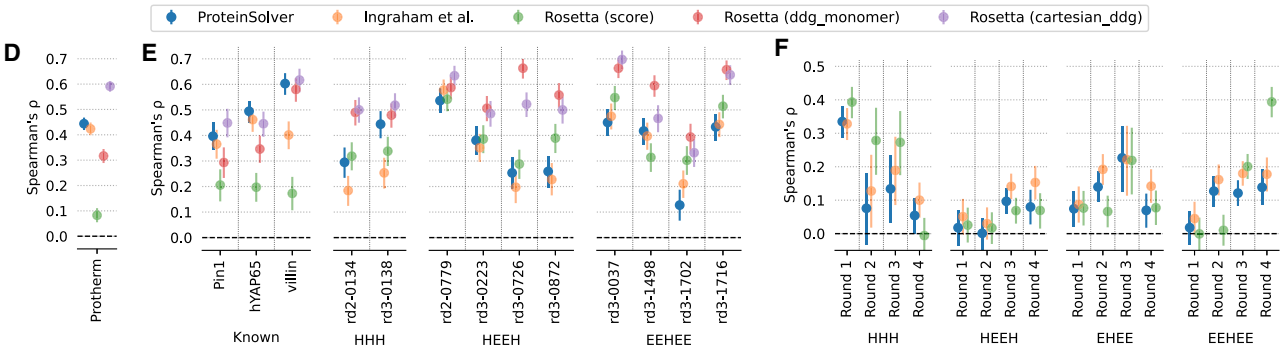
(C) Training a ProteinSolver network to reconstruct protein sequences. Node attributes encode the identities of individual amino acids. Edge attributes encode Euclidean distances between amino acids (depicted with shades of gray in the edge attributes matrix) and the relative positions of those amino acids along the amino acid chain.

CellPress
OPEN ACCESS

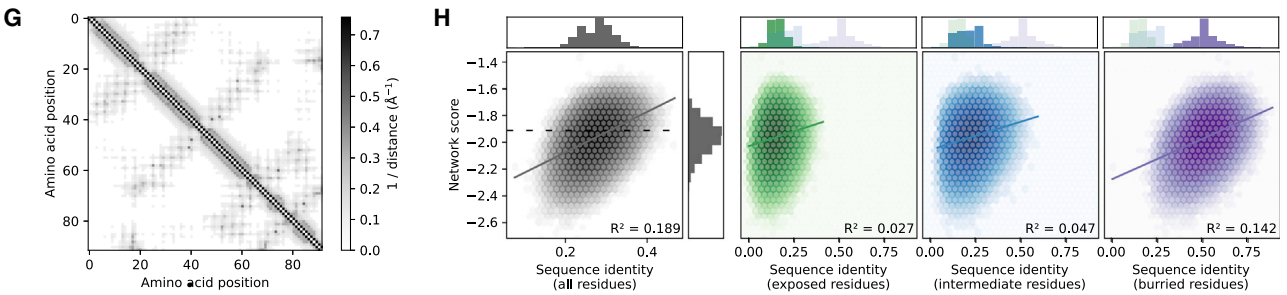
Training and validation accuracy of a ProteinSolver network optimized to reconstruct amino acid sequences

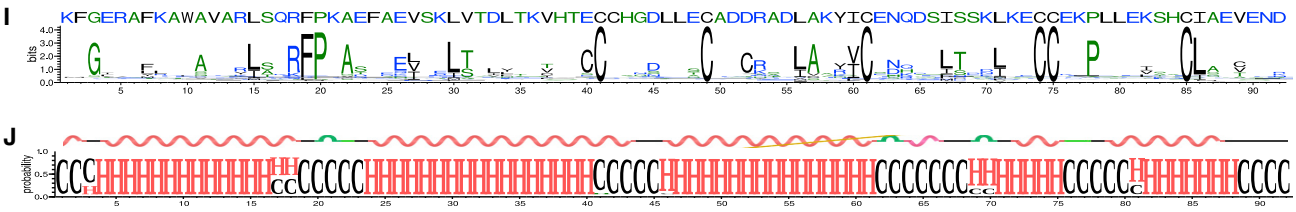


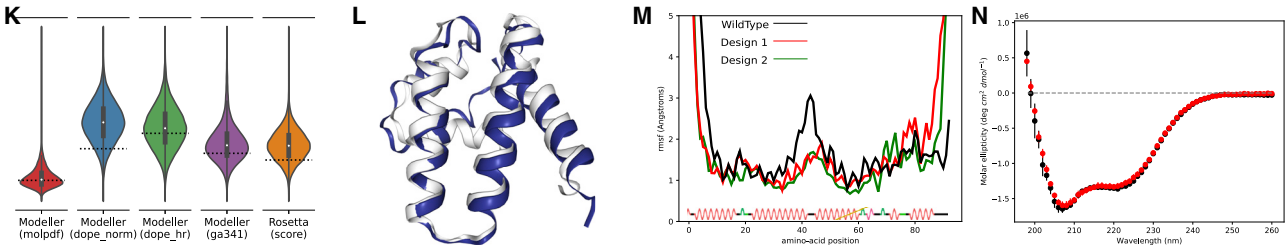
Independent validation of the trained network using stability measurements of mutated or designed proteins



Computational and experimental validation of sequences generated to match the architecture of a *serum albumin*







(legend on next page)

difficult to ascertain (Figure 1C). We treat proteins as graphs, where nodes correspond to the individual amino acids, and edges correspond to shortest distances between pairs of amino acids, calculated by considering all heavy atoms, and keeping only those pairs of amino acids that are within 12 Å of one another. In addition, nodes and edges have attributes, with node attributes specifying the amino acid, with an additional flag to indicate that the amino acid is not known; 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.

The structure of a protein contains all the information required to produce the amino acid sequence of the protein with perfect accuracy. As our goal was not to train a model capable of reconstructing the amino acid sequence of a reference structure but rather to train a model capable of producing a diverse set of sequences for a given protein fold, we strove to limit the amount of structural information available to the network and to increase the diversity of amino acid sequences in our training dataset. The former was accomplished by encoding structural information as shortest distances between pairs of residues, ignoring other information, such as backbone dihedral angles, the relative orientations between pairs of residues, or rotamers, which could facilitate overfitting on the reference structures. The latter was accomplished by including in our training dataset not only the sequences of proteins in the Protein Data Bank (PDB) but also of their homologs, which could be expected to have the same general fold.

We compiled a dataset of 72 million unique Gene3D domain sequences from UniParc (The UniProt Consortium, 2015) for which a structural template could be found in the PDB (Berman et al., 2000), and we split this dataset at the level of Gene3D superfamilies into training, validation, and test subsets. In order to improve the signal-to-noise ratio, we removed from the validation and test subsets sequences having less than 80% sequence identity with their structural templates. We trained the network by providing as input a partially masked amino acid sequence together with the 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 (Figure 1C). The training and validation accuracies achieved by the network reach a plateau after around 100 million training examples, with a training accuracy of ~22% and a validation accuracy of ~32%, when half of the residues are masked in the starting sequence (Figures 2A and 2B). The training accuracy is lower than the validation accuracy because the training dataset has no restriction on the similarity between the sequences and the structural templates and therefore likely contains more sequences for which the true structure diverges from the topology defined by the structural template. Reconstruction accuracy is considerably lower than Sudoku (Figures S1A–S1C), 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, and the achieved accuracy of 30%–40% roughly corresponds to the common level of sequence identity within a protein fold (Dawson et al., 2017). Evaluating our network by having it reconstruct sequences from our validation dataset, we observe a bimodal distribution in sequence identities between generated and reference

Figure 2. Using ProteinSolver to Evaluate the Stability of Proteins and to Generate New Protein Sequences

(A–F)Training and validation accuracy of a ProteinSolver network optimized to reconstruct amino acid sequences.

(A) Training and validation accuracy of the ProteinSolver network being trained to recover the identity of masked amino acid residues. During training, 50% of the amino acid residues in each input sequence were randomly masked as missing.

(B) Accuracy achieved by the ProteinSolver network on the test dataset, comprising 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.

(C) Sequence identity between generated and reference sequences in cases where 0%, 50%, or 80% of the reference sequences are made available to the network. Independent validation of the trained network using stability measurements of mutated or designed proteins.

(D and E) Spearman correlation coefficients between experimentally measured changes in protein stability associated with mutation and predictions made using ProteinSolver (blue), the Structured Transformer model described by Ingraham et al. (orange), Rosetta's relax and score protocols (green), Rosetta's ddg_monomer protocol (red), and Rosetta's cartesian_ddg protocol (purple). Experimentally measured changes in protein stability correspond either to changes in the Gibbs free energy of folding (Bava et al., 2004) (D) or changes in protein stability measured using an enzyme digestion assay (Rocklin et al., 2017) (E).

(F) Spearman correlation coefficients between the stability of proteins designed *de novo* to match a specific architecture, and predictions made using Proteinsolver (blue) and the Rosetta protocol and energy function that were used to generate those proteins (orange). HHH, HEEH, EHEE, and EEHEE denote proteins designed *de novo* using Rosetta in order to have a helix-helix-helix, helix-sheet-sheet-helix, sheet-helix-sheet-sheet, or sheet-sheet-helix-sheet-sheet architecture, respectively.

(D–F) Error bars show 95% confidence intervals obtained by calculating 10,000 Spearman correlation coefficients for each dataset using the bootstrap method.

(G–N) Computational and experimental validation of sequences generated to match the architecture of a serum albumin.

(G) Amino acid contact map extracted from the reference structure (PDB: *1N5U*) and provided as input to the network.

(H) Correlation between the scores assigned by the network to ~2 million generated sequences and the fraction of residues that the generated sequences have in common with the reference protein, considering either all residues (black), residues that are on the protein surface (green), residues that are intermediate (blue), or residues that are buried in the protein core (purple). The vertical dashed line denotes the score assigned by ProteinSolver to the wild-type sequence.

(I) Sequence logo showing the conservation of residues in the generated sequences.

(J) Secondary structure logo showing predicted secondary structures for a sample of 20,000 generated sequences.

(K) Distributions of scores obtained using Modeller and Rosetta to evaluate the stability of homology models constructed for a sample of 20,000 generated sequences. The vertical dashed line indicates the score assigned to the reference protein structure. In all five cases, a lower score indicates a more stable structure.

(L) Structure of the reference protein (white) overlaid with the structure of a model produced for one of the generated sequences using QUARK, a *de-novo* structural modeling tool (blue).

(M) Average residue fluctuation in 100-ns molecular dynamics simulations of the reference structure and two homology models of generated sequences.

(N) Circular dichroism spectra of the reference protein (black) and a generated protein (red).

sequences, with a smaller peak around 7% sequence identity and a larger peak around 37% sequence identity, corresponding to generated sequences about as similar to the reference sequences as other members of their family (see Figure 2C); other algorithms are reporting similar values, ranging from 30% to 39% (Ingraham et al., 2019; Li et al., 2014; O'Connell et al., 2018). Do note that reconstruction accuracy may not be the best measure for algorithm performance, indeed, values substantially higher than what we achieve would likely be an artifact, as it is already known for sequences with the achieved level of sequence identity to adopt the same fold and thus fulfill the same CSP. In fact, it may be more desirable to find new regions of sequence space that solve the same CSP, which nature has not sampled.

We next asked whether the score assigned by a trained network to single mutations can be predictive of whether those mutations are stabilizing or destabilizing. We speculated that a destabilizing mutation would also disrupt some of the constraints in the CSP and would thus be scored unfavorably by the graph neural network. We find that predictions made using ProteinSolver, which is trained solely to reconstruct protein sequences, show a significant correlation with experimentally measured changes in protein stability reported in Protherm (Bava et al., 2004) (Spearman ρ: 0.44; $p < 0.001$) and, in fact, show a stronger correlation than predictions made by scoring the wild-type and mutant structures using Rosetta's energy function or by using Rosetta's ddg_monomer protocol (Figure 2D). While predictions made using ProteinSolver show a weaker correlation than predictions made using Rosetta's cartesian_ddg protocol, both the cartesian_ddg protocol, and the beta_nov16 energy function used by that protocol, have been optimized in sight of the Protherm dataset and may have inadvertently overfit on this data (Park et al., 2016). Furthermore, Rosetta's cartesian_ddg protocol performs extensive structural relaxation and sampling around the site of the mutation and takes on the order of minutes to hours to evaluate a single mutation, while ProteinSolver can typically evaluate a mutation in under a second (see Table S2; Figure S2).

While the ProteinSolver network was not trained using any mutation data, we did not explicitly exclude the proteins in the Protherm dataset from our training dataset. In order to ascertain that the correlation between predictions made using ProteinSolver and the experimentally measured $\Delta\Delta G$ values is not biased by the presence of the wild-type sequences in our training dataset, we calculated the correlation between predictions made using ProteinSolver and the effect of mutations on the stability of a number of proteins reported to be designed *de novo* (Rocklin et al., 2017) (Figure 2E). While none of those *de-novo* designs appear in the ProteinSolver training dataset, ProteinSolver nevertheless achieves similar correlations to the scores obtained using Rosetta's beta_nov16 energy function, albeit showing somewhat weaker correlations than predictions made using Rosetta's ddg_monomer and cartesian_ddg protocols.

Finally, in order to evaluate how well ProteinSolver can score entire protein sequences and prioritize them for subsequent experimental evaluation, we calculated the correlation between the scores assigned by ProteinSolver to completely novel proteins that have been designed *de novo* using Rosetta, and the stability of those proteins, as measured using a high-throughput sequencing approach (Rocklin et al., 2017) (Figure 2F). We observe that, for most protein geometries and rounds of selection, the correlation between scores assigned by ProteinSolver and the stability of the proteins is similar to the correlation observed for scores produced by the Rosetta protocol used to generate the protein library. One exception is the round 4 library of the EEHEE designs, where Rosetta achieves a Spearman correlation of $\sim$0.4 while ProteinSolver achieves a Spearman correlation of $\sim$0.14, although it should be noted that ProteinSolver achieves a higher correlation for the round 2 library of designs with the same geometry.

The Structured Transformer model presented by Ingraham et al. (Ingraham et al., 2019) shows comparable performance to ProteinSolver on the three validation datasets (Figures 2D–2F), although correlations achieved by ProteinSolver are significantly stronger for two of the mutation subsets and are never significantly weaker (Tables S6 and S7). The two methods also have similar runtime characteristics (Figure S2) and are able to generate sequences with similar sequence identities to the reference proteins (Figure S7) and similar position-specific scoring matrix (PSSM) probabilities when using PSSMs for the CATH domains of the reference proteins (Figure S8). Conceptually, the biggest advantage of the ProteinSolver model over the Structured Transformer model is that ProteinSolver is bidirectional and thereby is able to take into account the identities of amino acids in both the preceding and the succeeding positions when making its predictions. In addition, ProteinSolver is trained using not only the proteins in the PDB but also their homologs, which has the potential to decrease overfitting and improve the generalizability of the model. On the other hand, the Structured Transformer model uses more structural information than ProteinSolver, including dihedral angles of the protein backbone and the relative orientation of amino acid residues with respect to each other, which could potentially improve the accuracy of its predictions.

Generating New Protein Sequences with Predefined Geometries

Motivated by the observation that the ProteinSolver network is able to reconstruct protein sequences with a reasonable level of accuracy and to assign probabilities to individual residues, as well as entire proteins, that correlate well with experimental measurements of protein stability, we sought to use the network to generate entirely novel protein sequences for specific protein folds. To that end, we chose four protein folds that had been left out of the training set and covered the breadth of the CATH hierarchy, and for each of those folds, we extracted a distance matrix from a protein structure representative of that fold. We designed new protein sequences matching those distance matrices by starting with entirely empty or "masked" protein sequences and, to each of the positions in those sequences, iteratively assigning residues by sampling from the residue probability distributions defined by the network (see STAR Methods). This corresponds to *de-novo* protein design—designing novel sequences for a given fold.

In total, we generated over 600,000 sequences for each of the four selected folds: serum albumin (mainly α; Figures 2G–2N), alanine racemase ($\alpha + \beta$; Figure S3), PDZ3 domain (mainly β; Figure S4), and immunoglobulin (mainly β; Figure S5). The

generated sequences tend to fall within 37%–44% sequence identity with the native sequence, in line with the sequence identity within the respective protein families (Figures 2H, S3B, S4B, and S5B). As expected, sequence identities are higher for residues that are buried in the core of the proteins than for residues that appear on the surface (Figures 2H, S3B, S4B, and S5B). Furthermore, the sequence conservation profiles of the generated sequences show that the network has a clear preference for specific residues at some positions, and in cases where several residues are equally preferred, those residues tend to have similar chemical properties (Figures 2I, S3C, S4C, and S5C). As no information about the sequences was provided to the network, it suggests that the network is able to learn the features pertinent to mapping structural information to sequences from the training dataset.

For each of the four selected folds, we took the top 20,000 sequences, as scored by our network, and performed further computational validation. First, we used PSIPRED (Jones, 1999) to predict the secondary structure of each of the generated sequences (Figures 2J, S3D, S4D, and S5D), and we found that the predicted secondary structures of our designs match the secondary structures of the reference proteins almost exactly. Next, we calculated the physico-chemical properties of our designs, including their molecular weight, hydropathy, isoelectric point, and instability (Figure S6), and we found that those values fall in the range of values obtained for the reference proteins. We proceeded to create homology models of our designs, and we evaluated those homology models using a number of metrics, including the Modeller (molpdf, dope_normalized, dope_hr, ga_341) (Marti-Renom et al., 2004) and the Rosetta (REU) scores (Alford et al., 2017) (Figures 2K, S3E, S4E, and S5E). In all cases, the scores obtained for the designs are in the same range or better than the scores obtained for the reference structures, suggesting that the sequences that our network generated *do novo* indeed fold in the shape corresponding to the provided distance matrix. We selected the top 2 sequences for each assessment metric plus our two best scored sequences (for a total of 12 models) to run QUARK (Xu and Zhang, 2012), a *de-novo* (not template-based) structure prediction algorithm, on our sequences (Figures 2L, S3F, S4F, and S5F). For all 4 folds, the obtained structures match the reference structure almost exactly. Finally, we performed 100-ns molecular dynamics (MD) simulations of the reference structures and the top 12 homology models of our *de-novo* designs (Figures 2M, S3G, S4G, and S5G). In all four cases, the designs show comparable fluctuation in MD to the reference proteins, indicating that they are of comparable stability as the reference proteins and are thus stably folded.

After obtaining encouraging results from our computational validation experiments, for two of the four selected folds, we chose sequences with the highest combined network and Modeller NormDOPE score, and we attempted to express and evaluate those sequences experimentally. Selected sequences were not exceptional relative to other designs in terms of their sequence identities to the reference proteins, and they did not have homologs in the training and validation datasets (Tables S4 and S5). We had each of the sequences synthesized as oligonucleotides, and we 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 spectrum in the far-UV region, and thus, similar folds should present similar absorbance spectra. For the 2 folds, serum albumin and alanine racemase, the selected sequences show a CD spectrum that is similar to the spectrum obtained for the native protein (Figures 2N and S3H). In the case of the serum albumin template, where the spectra are nearly indistinguishable (Figure 2N), and the predicted compositions of secondary structure elements differ by less than 2% ($p > 0.05$; Figures S11). It also makes it likely that the protein is folded, as without tertiary structure, the propensity for secondary structure would probably decrease. The sequence generated for the alanine racemase template displayed a considerable loss of solubility compared with the sequence from the target structure. Although this made its characterization challenging, we were able to obtain 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 (Figure S3H), this may be due to technical issues resulting from low solubility or a more dynamic nature of the designed protein (consistent with the MD in Figure S3G). As above, the spectrum corresponds to a largely helical secondary structure and is consistent with the predetermined conformation. Of course, for proof more experiments would be required, for instance dynamic light scattering to confirm that it exists in a monomer. Ultimately, a proper structure would have to be acquired either through crystallography or NMR.

Aligning the sequences of the reference proteins with the sequences of the designs that have been validated by CD spectroscopy reveals substantial differences not only in the surface-exposed and disordered regions of proteins but also in buried regions and in regions forming alpha helices and beta sheets (Figures S9 and S10). For example, in the case of serum albumin, a glutamic acid in position 40, which is inside an alpha helix and has a rSASA < 0.5, is replaced with a cysteine (Figure S9). In the homology model of the designed protein, the introduced cysteine is predicted to form a disulfide bond with the cysteine in position 49, and the presence of a new disulfide bond is supported by SDS-PAGE experiments of the reference and the designed protein under oxidizing and reducing conditions (Figure S13). In the case of alanine racemase, an asparagine in position 54, which is inside an alpha helix and has a rSASA < 0.2, is replaced with a leucine, and an arginine in position 71, which is inside a β sheet and has a rSASA < 0.05, is replaced with a phenylalanine (Figure S10). In the homology model of the designed protein, those two residues are predicted to interact, and the two substitutions could potentially serve to improve the hydrophobic packing in the protein core.

DISCUSSION

In this article, we present ProteinSolver, a graph neural network-based method for solving the protein design problem, formulated as a CSP, and generating protein sequences that satisfy the specified geometric and amino acid constraints. We show that a trained ProteinSolver network can reconstruct protein sequences with a high degree of accuracy, and that it assigns probabilities to individual residues, and to entire protein

sequences, that correlate well with the stability of the resulting proteins. Finally, a trained ProteinSolver network can generate previously unknown protein sequences, which are likely to fold into the same shapes as the reference proteins from which the geometric constraints are extracted.

Currently, we only use ProteinSolver to generate sequences for given geometries derived from existing proteins, but it would be possible to adapt it for use in conjunction with backbone exploration methods for designing novel architectures—the much greater speed would enable much more sampling of the large conformational space.

There has been growing interest in using neural network-based approaches for protein representation learning and design (Alley et al., 2019; Anand and Huang, 2018; Ingraham et al., 2019; Rives et al., 2019; Yang et al., 2018), with several new methods reported during the preparation of this manuscript (Anand et al., 2020; Ding et al., 2019). While most methods are accompanied by a variety of metrics which attempt to illustrate the accuracy of the predictions, it is inherently difficult to evaluate the quality of generative models, as their ultimate goal is to generate entirely novel sequences with no existing counterparts. To address this concern, we provide preliminary experimentally validation of our designs, having shown that their circular dichroism spectra are consistent with their target shapes.

One limitation of existing methods for protein design is the steep learning curve and the high degree of domain expertise that is necessary to make reasonable predictions. We circumvent those limitations by developing a web server which the users can use to generate, in near-real time, hundreds to thousands of sequences matching a given protein topology and amino acid constraints (Figure S12). It can be freely accessed at http://design.proteinsolver.org. We believe that this web server will make our graph neural network-based approach to protein design accessible to the widest possible audience and will facilitate the generation of many novel proteins.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- RESOURCE AVAILABILITY
  - Lead Contact
  - Materials Availability
  - Data and Code Availability
- METHOD DETAILS
  - Data Preparation
  - Network Implementation
  - Network Architecture
  - Scoring Existing Protein Sequences
  - Generating Novel Protein Sequences
  - Molecular Dynamics
  - Data Analysis
  - Web Server Implementation
  - Experimental Validation - Protein Purification
  - Experimental Validation - Circular Dichroism
- QUANTIFICATION AND STATISTICAL ANALYSIS
- ADDITIONAL RESOURCES

SUPPLEMENTAL INFORMATION

Supplemental Information can be found online at https://doi.org/10.1016/j.cels.2020.08.016.

ACKNOWLEDGMENTS

We thank Quaid Morris for valuable discussions and the anonymous reviewers, whose comments and suggestions have improved this manuscript. A.S. acknowledges support from an NSERC PGS-D graduate scholarship and the Google Cloud Research Credits program. D.B. acknowledges support from the Fonds de Recherche du Québec Nature et Technologies (FRQNT) - Bourses de recherche postdoctorale. P.M.K. acknowledges support from an NSERC Discovery grant (RGPIN-2017-064) and a CIHR project grant (PJT-166008). We also acknowledge HPC support from a Compute Canada Resource Allocation and the Nvidia academic GPU grant program.

AUTHOR CONTRIBUTIONS

Conceptualization, A.S. and P.M.K.; Data Curation, A.S. and D.B.; Formal Analysis and Methodology,: A.S.; Software, A.S. and D.B.; Resources, A.S., D.B., C.C.-C., A.P.-R., and P.M.K.; Validation, A.S., C.C.-C., D.B., and A.P.R.; Writing, A.S., D.B., C.C.-C., A.P.-R., and P.M.K.; Project Administration, P.M.K. Supervision, P.M.K.; Funding Acquisition, P.M.K.

DECLARATION OF INTERESTS

The authors are in the process of obtaining a patent on the described technology. P.M.K. is a cofounder of Resolute Bio Inc. and serves on the scientific advisory board of ProteinQure.

Received: April 2, 2020
Revised: June 27, 2020
Accepted: August 26, 2020
Published: September 23, 2020

REFERENCES

Alford, R.F., Leaver-Fay, A., Jeliazkov, J.R., O'Meara, M.J., DiMaio, F.P., Park, H., Shapovalov, M.V., Renfrew, P.D., Mulligan, V.K., Kappel, K., et al. (2017). The Rosetta all-atom energy function for macromolecular modeling and design. J. Chem. Theor. Comput. *13*, 3031–3048.

Alley, E.C., Khimulya, G., Biswas, S., AlQuraishi, M., and Church, G.M. (2019). Unified rational protein engineering with sequence-only deep representation learning. bioRxiv biorxiv.org/content/10.1101/589333v1.

Anand, N., Eguchi, R.R., Derry, A., Altman, R.B., and Huang, P.-S. (2020). Protein sequence design with a learned potential. bioRxiv biorxiv.org/content/10.1101/2020.01.06.895466v1.

Anand, N., and Huang, P. (2018). Generative modeling for protein structures. In Advances in Neural Information Processing Systems, *31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds. (Curran Associates, Inc.)), pp. 7494–7505.

Anthis, N.J., and Clore, G.M. (2013). Sequence-specific determination of protein and peptide concentrations by absorbance at 205 nm. Protein Sci *22*, 851–858.

Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. arXiv, arXiv:1806.01261.

Bava, K.A., Gromiha, M.M., Uedaira, H., Kitajima, K., and Sarai, A. (2004). ProTherm, version 4.0: thermodynamic database for proteins and mutants. Nucleic Acids Res *32*, D120–D121.

Beer, D.. https://dlbeer.co.nz/downloads/sugen.c.

Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., and Bourne, P.E. (2000). The protein data bank. Nucleic Acids Res *28*, 235–242.

Bommarius, A.S., and Paye, M.F. (2013). Stabilizing biocatalysts. Chem. Soc. Rev. *42*, 6534–6565.

Brookes, D.H., Park, H., and Listgarten, J. (2019). Conditioning by adaptive sampling for robust design. arXiv, arXiv:1901.10060.

Case, D.A., Cheatham, T.E., III, Darden, T., Gohlke, H., Luo, R., Merz, K.M., Jr., Onufriev, A., Simmerling, C., Wang, B., and Woods, R.J. (2005). The Amber biomolecular simulation programs. J. Comput. Chem. *26*, 1668–1688.

Chevalier, A., Silva, D.A., Rocklin, G.J., Hicks, D.R., Vergara, R., Murapa, P., Bernard, S.M., Zhang, L., Lam, K.H., Yao, G., et al. (2017). Massively parallel de novo protein design for targeted therapeutics. Nature *550*, 74–79.

Dal Palù, A., Dovier, A., and Fogolari, F. (2004). Constraint Logic Programming approach to protein structure prediction. BMC Bioinformatics *5*, 186.

Dawson, N.L., Lewis, T.E., Das, S., Lees, J.G., Lee, D., Ashford, P., Orengo, C.A., and Sillitoe, I. (2017). CATH: an expanded resource to predict protein function through structure and sequence. Nucleic Acids Res *45*, D289–D295.

Ding, X., Zou, Z., and Brooks, C.L., iii. (2019). Deciphering protein evolution and fitness landscapes with latent space models. Nat. Commun. *10*, 5644.

Fey, M., and Lenssen, J.E. (2019). Fast graph representation learning with PyTorch geometric. arXiv, arXiv:1903.02428.

Fischman, S., and Ofran, Y. (2018). Computational design of antibodies. Curr. Opin. Struct. Biol. *51*, 156–162.

Ingraham, J., Garg, V., Barzilay, R., and Jaakkola, T. (2019). Generative models for graph-based protein design. In Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, eds. (Curran Associates, Inc.)), pp. 15820–15831.

Jones, D.T. (1999). Protein secondary structure prediction based on position-specific scoring matrices. J. Mol. Biol. *292*, 195–202.

Khan, S., and Vihinen, M. (2010). Performance of protein stability predictors. Hum. Mutat. *31*, 675–684.

Kocić, J., Jovičić, N., and Drndarević, V. (2019). An end-to-end deep neural network for autonomous driving designed for embedded automotive platforms. Sensors *19*, 2064.

Kroncke, B.M., Duran, A.M., Mendenhall, J.L., Meiler, J., Blume, J.D., and Sanders, C.R. (2016). Documentation of an imperative to improve methods for predicting membrane protein stability. Biochemistry *55*, 5002–5009.

Kunzmann, P., and Hamacher, K. (2018). Biotite: a unifying open source computational biology framework in Python. BMC Bioinformatics *19*, 346.

Lewis, T.E., Sillitoe, I., Dawson, N., Lam, S.D., Clarke, T., Lee, D., Orengo, C., and Lees, J. (2017). Gene3D: extensive prediction of globular domains in proteins. Nucleic Acids Res *46*, D1282.

Li, Z., Yang, Y., Faraggi, E., Zhan, J., and Zhou, Y. (2014). Direct prediction of profiles of sequences compatible with a protein structure by neural networks with fragment-based local and energy-based nonlocal profiles. Proteins *82*, 2565–2573.

Maier, J.A., Martinez, C., Kasavajhala, K., Wickstrom, L., Hauser, K.E., and Simmerling, C. (2015). ff14SB: improving the accuracy of protein side chain and backbone parameters from ff99SB. J. Chem. Theor. Comput. *11*, 3696–3713.

Marti-Renom, M.A., Madhusudhan, M.S., and Sali, A. (2004). Alignment of protein sequences by their profiles. Protein Sci *13*, 1071–1087.

McGibbon, R.T., Beauchamp, K.A., Harrigan, M.P., Klein, C., Swails, J.M., Hernández, C.X., Schwantes, C.R., Wang, L.P., Lane, T.J., and Pande, V.S. (2015). MDTraj: a modern open library for the analysis of molecular dynamics trajectories. Biophys. J. *109*, 1528–1532.

Micsonai, A., Wien, F., Kernya, L., Lee, Y.H., Goto, Y., Réfrégiers, M., and Kardos, J. (2015). Accurate secondary structure prediction and fold recognition for circular dichroism spectroscopy. Proc. Natl. Acad. Sci. USA *112*, E3095–E3103.

O'Connell, J., Li, Z., Hanson, J., Heffernan, R., Lyons, J., Paliwal, K., Dehzangi, A., Yang, Y., and Zhou, Y. (2018). SPIN2: predicting sequence profiles from protein structures using deep neural networks. Proteins *86*, 629–633.

Palm, R.B., Paquet, U., and Winther, O. (2017). Recurrent relational networks. arXiv, arXiv:1711.08028.

Park, K. (2019). Can neural networks crack sudoku?. https://github.com/Kyubyong/sudoku.

Park, H., Bradley, P., Greisen, P., Liu, Y., Mulligan, V.K., Kim, D.E., Baker, D., and DiMaio, F. (2016). Simultaneous optimization of biomolecular energy functions on features from small molecules and macromolecules. J. Chem. Theor. Comput. *12*, 6201–6212.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch. 31st Conference on Neural Information Processing Systems (NIPS), pp. 8024–8035.

Perez-Riba, A., and Itzhaki, L.S. (2017). A method for rapid high-throughput biophysical analysis of proteins. Sci. Rep. *7*, 9071.

Prates, M.O.R., Avelar, P.H.C., Lemos, H., Lamb, L.C., and Vardi, M.Y. (2018). Learning to solve NP-complete problems - a graph neural network for decision TSP. arXiv, arXiv:1809.02721.

Punta, M., Coggill, P.C., Eberhardt, R.Y., Mistry, J., Tate, J., Boursnell, C., Pang, N., Forslund, K., Ceric, G., Clements, J., et al. (2012). The Pfam protein families database. Nucleic Acids Res *40*, D290–D301.

Rives, A., Goyal, S., Meier, J., Guo, D., Ott, M., Zitnick, C.L., Ma, J., and Fergus, R. (2019). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. bioRxiv https://www.biorxiv.org/content/10.1101/622803v1.

Rocklin, G.J., Chidyausiku, T.M., Goreshnik, I., Ford, A., Houliston, S., Lemak, A., Carter, L., Ravichandran, R., Mulligan, V.K., Chevalier, A., et al. (2017). Global analysis of protein folding using massively parallel design, synthesis, and testing. Science *357*, 168–175.

Ryckaert, J.-P., Ciccotti, G., and Berendsen, H.J.C. (1977). Numerical integration of the Cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. J. Comput. Phys. *23*, 327–341.

Senior, A.W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A.W.R., Bridgland, A., et al. (2020). Improved protein structure prediction using potentials from deep learning. Nature *577*, 706–710.

Shultis, D., Mitra, P., Huang, X., Johnson, J., Khattak, N.A., Gray, F., Piper, C., Czajka, J., Hansen, L., Wan, B., et al. (2019). Changing the apoptosis pathway through evolutionary protein design. J. Mol. Biol. *431*, 825–841.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of Go without human knowledge. Nature *550*, 354–359.

Sun, M.G.F., and Kim, P.M. (2017). Data driven flexible backbone protein design. PLoS Comput. Biol. *13*, e1005722.

Sun, M.G.F., Seo, M.H., Nim, S., Corbi-Verge, C., and Kim, P.M. (2016). Protein engineering by highly parallel screening of computationally designed variants. Sci. Adv. *2*, e1600692.

Toukmaji, A., Sagui, C., Board, J., and Darden, T. (2000). Efficient particle-mesh Ewald based approach to fixed and induced dipolar interactions. J. Chem. Phys. *113*, 10913–10927.

Ullah, A.D., and Steinhöfel, K. (2010). A hybrid approach to protein folding problem integrating constraint programming with local search. BMC Bioinformatics *11*, S39.

UniProt Consortium (2015). UniProt: a hub for protein information. Nucleic Acids Res *43*, D204–D212.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. arXiv, arXiv:1706.03762.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods *17*, 261–272.

Wainberg, M., Merico, D., Delong, A., and Frey, B.J. (2018). Deep learning in biomedicine. Nat. Biotechnol. *36*, 829–838.

CellPress
OPEN ACCESS

Wang, J., Cao, H., Zhang, J.Z.H., and Qi, Y. (2018a). Computational protein design with deep learning neural networks. Sci. Rep. *8*, 6349.

Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., and Solomon, J.M. (2018b). Optimization of a Si-SiO2 waveguide coupler for photonic integrated circuits. Circuits and Systems *9*, 101–106.

Xu, D., and Zhang, Y. (2012). Ab initio protein structure assembly using continuous structure fragments and optimized knowledge-based force field. Proteins *80*, 1715–1735.

Yang, K.K., Wu, Z., Bedbrook, C.N., and Arnold, F.H. (2018). Learned protein embeddings for machine learning. Bioinformatics *34*, 2642–2648.

Sudoku (2020) Sudoku free online to play and print. 1sudoku.com.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Bacterial and Virus Strains		
E. coli OverExpress C41(DE3) (Green cap)	Lucigen	Cat#60442-2
Critical Commercial Assays		
Pierce BCA Protein Assay Kit	ThermoFisher	Cat#23227
Deposited Data		
Protein Data Bank (PDB)	Berman et al., 2000	ftp://ftp.wwpdb.org/pub/pdb/data/structures/divided/
UniParc	The UniProt Consortium, 2015	http://www.uniprot.org/downloads
Recombinant DNA		
Plasmid: pRSET-A	ThermoFisher	Cat#V35120
Software and Algorithms		
PyTorch	Paszke et al., 2017	https://github.com/pytorch/pytorch
PyTorch Geometric	Fey and Lenssen, 2019	https://github.com/rusty1s/pytorch_geometric
MDTraj	McGibbon et al., 2015	https://github.com/mdtraj/mdtraj
SciPy	Virtanen et al., 2020	https://github.com/scipy/scipy
Biotite	Kunzmann and Hamacher, 2018	https://github.com/biotite-dev/biotite
Amber 16	Case et al., 2005	https://ambermd.org/
PSIPRED	Jones, 1999	http://bioinf.cs.ucl.ac.uk/psipred/
Modeller	Marti-Renom et al., 2004	https://salilab.org/modeller/
Rosetta	Alford et al., 2017	https://www.rosettacommons.org/software/
Quark	Xu and Zhang, 2012	https://zhanglab.ccmb.med.umich.edu/QUARK/
BeStSel	Micsonai et al., 2015	http://bestsel.elte.hu/index.php

RESOURCE AVAILABILITY

Lead Contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the Lead Contact, Dr. Philip M. Kim (pi@kimlab.org).

Materials Availability

This study did not generate new unique reagents.

Data and Code Availability

The code generated during this study is hosted on GitLab: https://gitlab.com/ostrokach/proteinsolver/.

The datasets generated during this study are available at the following URL: http://deep-protein-gen.data.proteinsolver.org/.

METHOD DETAILS

Data Preparation

We downloaded from (UniPort Consortium, 2015) 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 extracted the amino acid sequence and the distance matrix of every chain in every structure. The distance matrix consists of distances between all pairs of residues that are within 12 Angstroms of one another, considering both the backbone and the sidechain residues in this calculation. Finally, we attempted to find a structural template for every Gene3D domain sequence, and we transferred the distance matrices from the structural templates to each of those sequences. 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. Instructions on how to download the training and validation datasets are provided on the ProteinSolver documentation page (https://ostrokach.gitlab.io/proteinsolver). A list of all resources used to construct those datasets is provided Table S1.

In the case of Sudoku, the training and validation datasets were generated using the sugen program (Beer, 2011) with the target difficulty of the puzzles set to 500. The training dataset was composed of 30 million generated puzzles, while the validation dataset was composed of 1000 puzzles that do not appear in the training dataset. The test dataset was comprised of 30 Sudoku puzzles collected from http://1sudoku.com (Park, 2019, [CSL STYLE ERROR: reference with no printed form.]).

Network Implementation

The source code for ProteinSolver is freely available at https://gitlab.com/ostrokach/proteinsolver. The network was implemented in the Python programming language using PyTorch (Paszke et al., 2017) and PyTorch Geometric (Fey and Lenssen, 2019) libraries. The repository also includes Jupyter notebooks that can be used to reproduce all the figures presented in this manuscript.

Network Architecture

We used Sudoku as a toy problem while optimizing the general design of the ProteinSolver network, including selecting the objective function that is optimized (masking a fraction of node labels and minimizing the cross-entropy loss between predicted and actual labels), tuning the specific implementation of edge convolutions and aggregations (using a 2-layer feed-forward network to update edge attributes, summing over linearly-transformed edge attributes of the incident edges to update node attributes, etc.), and selecting the types of non-linearities and normalizations that are applied (ReLU and LayerNorm, respectively).

Once we had a network that showed promising results in its ability to solve Sudoku puzzles, we tuned the specific hyperparameters of that network and selected variants that achieved the highest accuracies on the validation datasets. For the task of solving Sudoku puzzles (Figures 2A–2C), the model that achieved the highest accuracy on the validation dataset had 16 residual edge convolution and aggregation (ECA) blocks and a node and edge embedding space of 162. For the task of reconstructing protein sequences (Figures 2D–2F), the model that achieved the highest accuracy on the validation dataset had 4 residual blocks and a node and edge embedding space of 128.

Scoring Existing Protein Sequences

We evaluated the plausibility of existing protein sequences by using a trained ProteinSolver network to calculate the log-probability of every residue in those sequences, given all other residues, and taking the average of those log-probabilities. In effect, for every residue in a given sequence, we replaced the node label corresponding to that residue with a "mask" token, and we used a trained network to obtain the log-probability of the correct residue at the masked position. The score for the protein (e.g. Figure 2I) was calculated as the average of the log-probabilities assigned to all residues. The score for a mutation (e.g. Figures 2G and 2H) was calculated as the difference between log-probabilities of the mutant and the wild-type protein.

For comparison, we used Rosetta's relax and score protocols, Rosetta's ddg_monomer protocol, and Rosetta's cartesian_ddg protocol to predict changes in protein stability caused by mutations (see Table S3).

Generating Novel Protein Sequences

In order to calculate the most probable protein sequence, given a specific distance matrix, we evaluated two different approaches: one-shot generation and incremental generation. In one-shot generation, we passed the inputs with the missing node labels through the network only once, for every node accepting the label assigned by the network the highest probability. In incremental generation, we passed the inputs through the network once for every missing label. At each iteration, we accepted the label for which the network was making the most confident prediction, and we treated that label as given in all subsequent iterations. The one-shot generation method is substantially faster, requiring O(1), rather than O(N), passes through the network, while the incremental generation method appears to produce more accurate results, especially in the case of Sudoku (see Figures 2C and 2E).

In order to generate *a library* of protein sequences, given a specific distance matrix, we used an approach similar to the incremental generation method described above. However, at each iteration, instead of deterministically accepting the residue to which the network assigns the highest probability, we select the residue by randomly sampling from the probability distribution assigned to that position by the network.

For the purpose of calculating amino acid sequence identity between the reference protein and the designs (Figures 2H, S3B, S4B, and S5B), buried residues were defined as residues with relative surface accessible surface area (rSASA) under 20%, intermediate residues were defined as residues with rSASA between 20% and 50%, and exposed residues were defined as residues with rSASA greater than 50%.

Molecular Dynamics

All water and ion 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 (Case et al., 2005) and AMBER ff14SB force field (Maier et al., 2015), 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 (Ryckaert et al., 1977) and a 2 fs time step was used. The particle mesh Ewald (Toukmaji et al., 2000) algorithm was used to treat long-range interactions.

Data Analysis

Protein structural analysis, including the calculation of solvent-accessible surface area and secondary structure components, was carried out using MDTraj (McGibbon et al., 2015). Figures depicting the pairwise alignment between reference and design proteins (Figures S9 and S10) were created using Biotite (Kunzmann and Hamacher, 2018). Spearman's correlation coefficients were calculated using SciPy (Virtanen et al., 2020).

Web Server Implementation

In order to make our method accessible to the widest possible audience, we developed a web server which allows the user to run a trained ProteinSolver model to generate new protein sequences matching geometric constraints extracted from a reference structure and, optionally, specific amino acid constraints imposed directly by the user (Figure S12; http://design.proteinsolver.org). For the curious reader, we have also made available a web server which can be used to run a ProteinSolver model trained to solve Sudoku puzzles (http://sudoku.proteinsolver.org). The web servers were implemented using the Voila web framework and are deployed as Docker containers running on Google Kubernetes Engine. We use Knative in order to automatically scale the number of virtual machines and running Docker containers, providing a responsive experience for an arbitrary number of users while minimizing the overall cost of operation. The protein design web server is deployed to virtual machines equipped with an NVIDIA Tesla T4 GPU, with at most eight concurrent users sharing a single GPU.

Experimental Validation - Protein Purification

All constructs were synthesized by Invitrogen GeneArt Synthesis (Thermo Fisher) as Gene Strings. The construct designs 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 pelletized 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 (Perez-Riba and Itzhaki, 2017).

Protein concentration was calculated using the Pierce BCA Protein Assay Kit (Thermo Fisher). The concentration of 4beu_Design was corroborated by absorbance at 205 nm as described by Anthis et. al. (Anthis and Clore, 2013). 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).

Experimental Validation - Circular Dichroism

All CD measurements were made on a Jasco J-810 CD spectrometer in a 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 spectra were collected in the 198 nm to 260 nm wavelength range using a 1 nm bandwidth and 1 nm intervals at 50 nm/min; each reading was repeated then times. All measurements were taken at 20 °C. The obtained CD spectra were input into the BeStSel webserver in mean residue ellipticity units to obtain predicted composition of secondary structure elements (Micsonai et al., 2015).

QUANTIFICATION AND STATISTICAL ANALYSIS

In Figures 2D–2F, error bars were calculated by generating 10,000 Spearman's correlation coefficients for each point using the bootstrapping technique and using those generated correlation coefficients to compute the 95% confidence interval.

In Table S6, comparisons to ProteinSolver were performed by comparing 10,000 bootstrapped samples of Spearman correlation coefficients obtained for ProteinSolver and for each of the other methods. A method was deemed to be superior to ProteinSolver for a given dataset if the Spearman's correlation coefficients obtained for that method were higher than the Spearman's correlation coefficients obtained for ProteinSolver at least 99.955% of the time, and a method was deemed to be inferior if the Spearman's

correlation coefficients obtained for that method were lower than the Spearman's correlation coefficients obtained for ProteinSolver at least 99.955% of the time. The value of 99.955% was calculated by applying the Bonferroni correction for multiple comparisons to the standard cutoff of 97.5%.

In Table S7, comparisons to ProteinSolver were performed by comparing 10,000 bootstrapped samples of Spearman correlation coefficients obtained for ProteinSolver and for each of the other methods. A method was deemed to be superior to ProteinSolver for a given dataset if the Spearman's correlation coefficients obtained for that method were higher than the Spearman's correlation coefficients obtained for ProteinSolver at least 99.922% of the time, and a method was deemed to be inferior if the Spearman's correlation coefficients obtained for that method were lower than the Spearman's correlation coefficients obtained for ProteinSolver at least 99.922% of the time. The value of 99.922% was calculated by applying the Bonferroni correction for multiple comparisons to the standard cutoff of 97.5%.

ADDITIONAL RESOURCES

Documentation for ProteinSolver is available at https://proteinsolver.org. A web server which allows users to run a trained ProteinSolver model to generate sequences matching the geometries of their own reference proteins is available at: http://design.proteinsolver.org/.