Neural networks to learn protein sequence–function relationships from deep mutational scanning data

Sam Gelman[a,b] ⬥, Sarah A. Fahlberg[c], Pete Heinzelman[c] ⬥, Philip A. Romero[c,1,2] ⬥, and Anthony Gitter[a,b,d,1,2] ⬥

[a]Department of Computer Sciences, University of Wisconsin–Madison, Madison, WI 53706; [b]Morgridge Institute for Research, Madison, WI 53715; [c]Department of Biochemistry, University of Wisconsin–Madison, Madison, WI 53706; and [d]Department of Biostatistics and Medical Informatics, University of Wisconsin–Madison, Madison, WI 53792

The mapping from protein sequence to function is highly complex, making it challenging to predict how sequence changes will affect a protein's behavior and properties. We present a supervised deep learning framework to learn the sequence–function mapping from deep mutational scanning data and make predictions for new, uncharacterized sequence variants. We test multiple neural network architectures, including a graph convolutional network that incorporates protein structure, to explore how a network's internal representation affects its ability to learn the sequence–function mapping. Our supervised learning approach displays superior performance over physics-based and unsupervised prediction methods. We find that networks that capture nonlinear interactions and share parameters across sequence positions are important for learning the relationship between sequence and function. Further analysis of the trained models reveals the networks' ability to learn biologically meaningful information about protein structure and mechanism. Finally, we demonstrate the models' ability to navigate sequence space and design new proteins beyond the training set. We applied the protein G B1 domain (GB1) models to design a sequence that binds to immunoglobulin G with substantially higher affinity than wild-type GB1.

protein engineering | deep learning | convolutional neural network

Understanding the mapping from protein sequence to function is important for describing natural evolutionary processes, diagnosing genetic disease, and designing new proteins with useful properties. This mapping is shaped by thousands of intricate molecular interactions, dynamic conformational ensembles, and nonlinear relationships between biophysical properties. These highly complex features make it challenging to model and predict how changes in amino acid sequence affect function.

The volume of protein data has exploded over the last decade with advances in DNA sequencing, three-dimensional structure determination, and high-throughput screening. With these increasing data, statistics and machine learning approaches have emerged as powerful methods to understand the complex mapping from protein sequence to function. Unsupervised learning methods such as EVmutation (1) and DeepSequence (2) are trained on large alignments of evolutionarily related protein sequences. These methods can model a protein family's native function, but they are not capable of predicting specific protein properties that were not subject to long-term evolutionary selection. In contrast, supervised methods learn the mapping to a specific protein property directly from sequence–function examples. Many prior supervised learning approaches have limitations, such as the inability to capture nonlinear interactions (3, 4), poor scalability to large datasets (5), making predictions only for single-mutation variants (6), or a lack of available code (7). Other learning methods leverage multiple sequence alignments and databases of annotated genetic variants to make qualitative predictions about a mutation's effect on organismal fitness or disease, rather than making quantitative predictions of molecular phenotype (8–10). There is a current need for general, easy to use supervised learning methods that can leverage large

sequence–function datasets to predict specific molecular phenotypes with the high accuracy required for protein design. We address this need with a usable software framework that can be readily adopted by others for new proteins (11).

We present a deep learning framework to learn protein sequence–function relationships from large-scale data generated by deep mutational scanning experiments. We train supervised neural networks to learn the mapping from sequence to function. These trained networks can then generalize to predict the functions of previously unseen sequences. We examine network architectures with different representational capabilities including linear regression, nonlinear fully connected networks, and convolutional networks that share parameters. Our supervised modeling approach displays strong predictive accuracy on five diverse deep mutational scanning datasets and compares favorably with state-of-the-art physics-based and unsupervised prediction methods. Across the different architectures tested, we find that networks that capture nonlinear interactions and share information across sequence positions display the greatest predictive performance. We explore what our neural network models have learned about proteins and how they comprehend the sequence–function mapping. The convolutional neural networks learn a protein sequence representation that

---

Significance

Understanding the relationship between protein sequence and function is necessary to design new and useful proteins with applications in bioenergy, medicine, and agriculture. The mapping from sequence to function is tremendously complex because it involves thousands of molecular interactions that are coupled over multiple lengths and timescales. We show that neural networks can learn the sequence–function mapping from large protein datasets. Neural networks are appealing for this task because they can learn complicated relationships from data, make few assumptions about the nature of the sequence–function relationship, and can learn general rules that apply across the length of the protein sequence. We demonstrate that learned models can be applied to design new proteins with properties that exceed natural sequences.

---

organizes sequences according to their structural and functional differences. In addition, the importance of input sequence features displays a strong correspondence to the protein's three-dimensional structure and known key residues. Finally, we used an ensemble of the supervised learning models to design five protein G B1 domain (GB1) sequences with varying distances from the wild type. We experimentally characterized these sequences and found the top design binds to immunoglobulin G (IgG) with at least an order of magnitude higher affinity than wild-type GB1.

Results

A Deep Learning Framework to Model the Sequence–Function Mapping. Neural networks are capable of learning complex, nonlinear input–output mappings; extracting meaningful, higher-level features from raw inputs; and generalizing from training data to new, unseen inputs (12). We develop a deep learning framework to learn from large-scale sequence–function data generated by deep mutational scanning. Deep mutational scanning data consist of thousands to millions of protein sequence variants that each have an associated score that quantifies their activity or fitness in a high-throughput function assay (13). We encode the protein sequences with a featurization that captures the identity and physicochemical properties of each amino acid at each position. Our approach encodes the entire protein sequence and thus can represent multimutation variants. We train a neural network to map the encoded sequences to their associated functional scores. After it is trained, the network generalizes and can predict functional scores for new, unseen protein variants (Fig. 1*A*).

We test four supervised learning models to explore how different internal representations influence the ability to learn the mapping from protein sequence to function: linear regression and fully connected, sequence convolutional, and graph

convolutional neural networks (Fig. 1*B*). Linear regression serves as a simple baseline because it cannot capture dependencies between sites, and thus, all residues make additive contributions to the predicted fitness. Fully connected networks incorporate multiple hidden layers and nonlinear activation functions, enabling them to learn complex nonlinearities in the sequence to function mapping. In contrast to linear regression, fully connected networks are capable of modeling how combinations of residues jointly affect function beyond simple additive effects. These nonadditive effects are known as mutational epistasis (14, 15). Neither linear regression nor fully connected networks are able to learn meaningful weights for amino acid substitutions that are not directly observed in the training set.

Convolutional neural networks have parameter sharing architectures that enable them to learn higher-level features that generalize across different sequence positions. They learn convolutional filters that identify patterns across different parts of the input. For example, a filter may learn to recognize the alternating pattern of polar and nonpolar amino acids commonly observed in β-strands. Applying this filter would enable the network to assess β-strand propensity across the entire input sequence and relate this higher-level information to the observed protein function. Importantly, the filter parameters are shared across all sequence positions, enabling convolutional networks to make meaningful predictions for mutations that were not directly observed during training. We develop a sequence-based convolutional network that integrates local sequence information by applying filters using a sliding window across the amino acid sequence. We also develop a structure-based graph convolutional network that integrates three-dimensional structural information and may allow the network to learn filters that correspond to structural motifs. The graph convolutional network applies filters to neighboring nodes in a graph representation of the protein's structure. The protein structure graph consists of a node for each residue and an
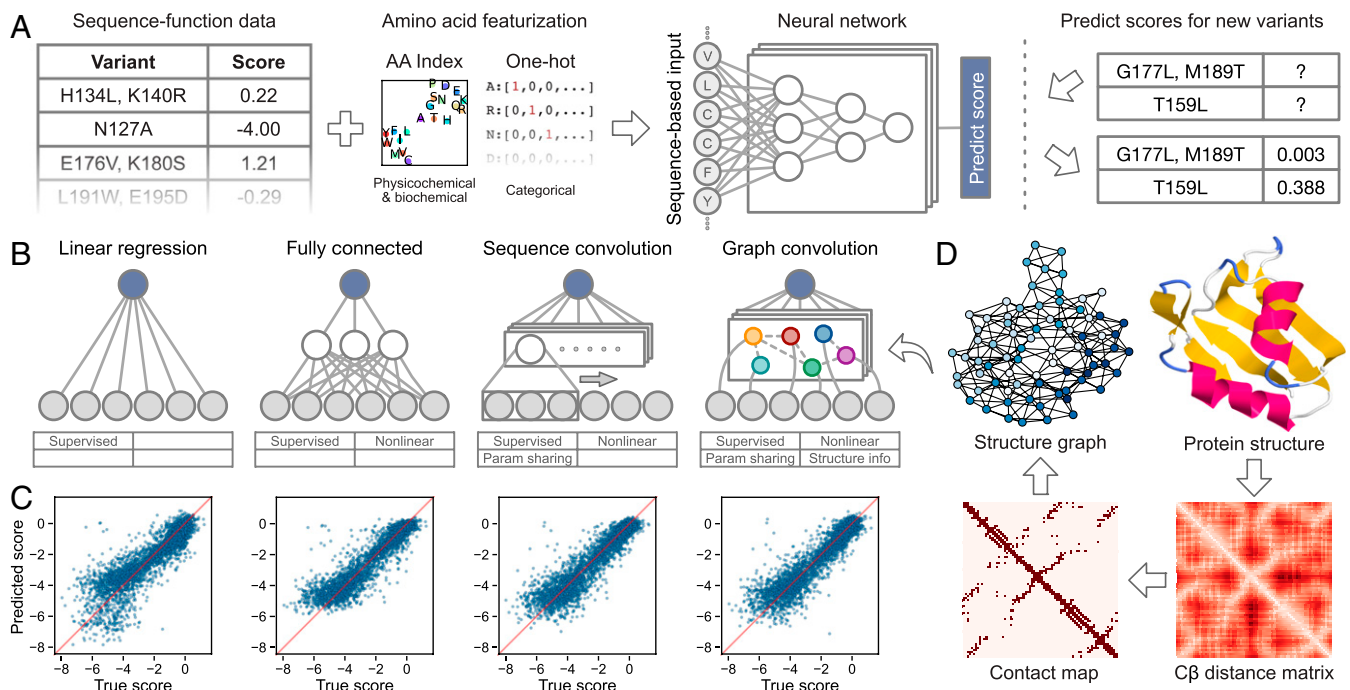

Fig. 1. Overview of our supervised learning framework. (*A*) We use sequence–function data to train a neural network that can predict the functional score of protein variants. The sequence-based input captures physicochemical and biochemical properties of amino acids and supports multiple mutations per variant. The trained network can predict functional scores for previously uncharacterized variants. (*B*) We tested linear regression and three types of neural network architectures: fully connected, sequence convolutional, and graph convolutional. (*C*) Scatterplots showing performance of trained networks on the Pab1 dataset. (*D*) Process of generating the protein structure graph for Pab1. We create the protein structure graph by computing a residue distance matrix from the protein's three-dimensional structure, thresholding the distances, and converting the resulting contact map to an undirected graph. The structure graph is the core part of the graph convolutional neural network.

edge between nodes if the residues are within a specified distance in three-dimensional space (Fig. 1*D*).

Evaluating Models Learned from Deep Mutational Scanning Data. We evaluated the predictive performance of the different network architectures on five diverse deep mutational scanning datasets representing proteins of varying sizes, folds, and functions: *Aequorea victoria* green fluorescent protein (avGFP), β-glucosidase (Bgl3), GB1, poly(A)-binding protein (Pab1), and ubiquitination factor E4B (Ube4b) (Fig. 2*A* and Table 1). These datasets range in size from ∼25,000 to ∼500,000

sequence-score examples. We randomly split each dataset into training, tuning, and testing sets to optimize hyperparameters and evaluate predictive performance on data that were not seen during training. The learned models displayed excellent test set predictions for most datasets, with Pearson's correlation coefficients ranging from 0.55 to 0.98 (Fig. 2*B*). The trends are generally similar using Spearman's correlation coefficient (*SI Appendix*, Fig. S1), although the differences between linear regression and the neural networks are smaller.

For comparison, we also evaluated the predictive performance of established physics-based and unsupervised learning methods
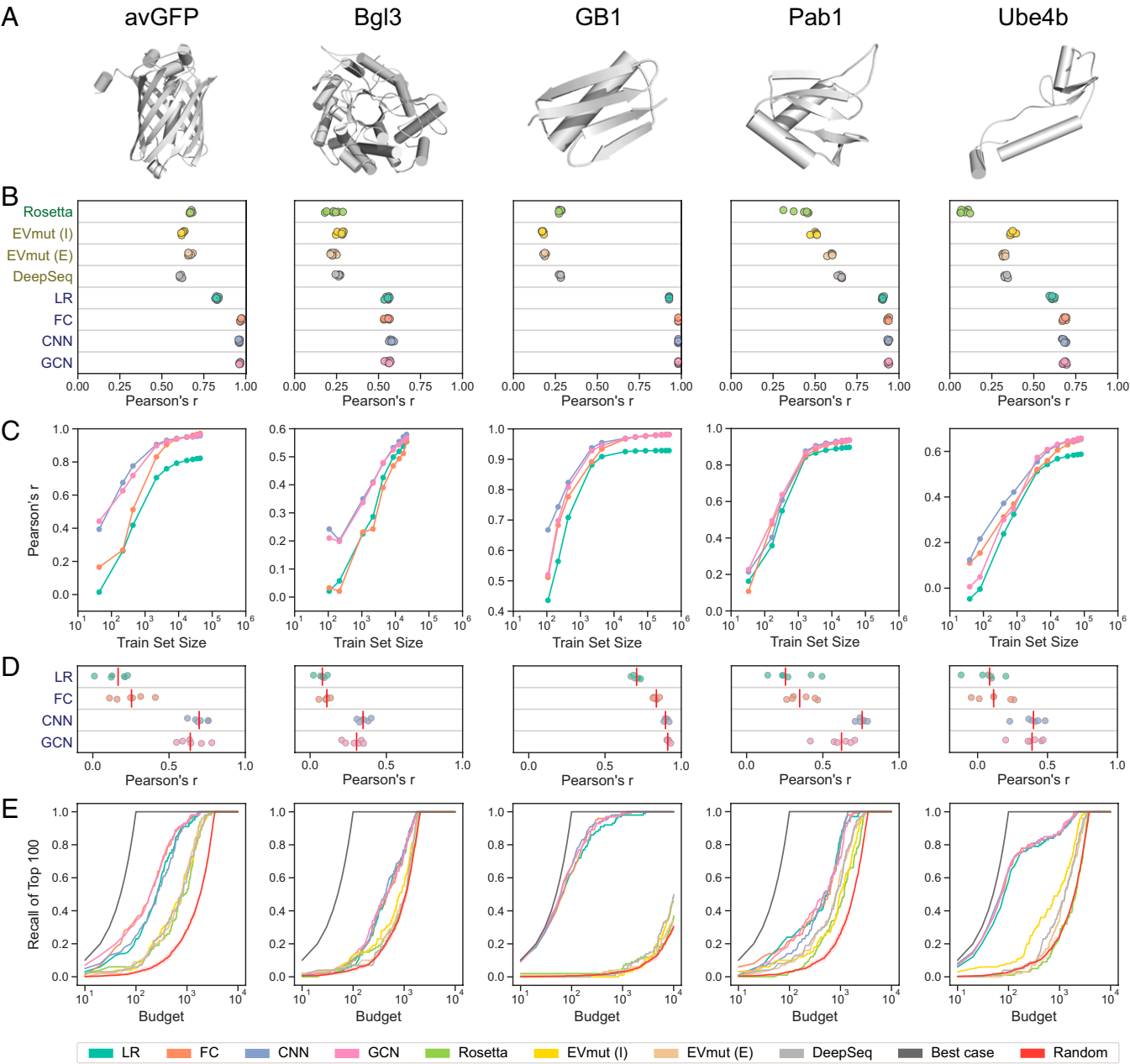

Fig. 2. Evaluation of neural networks and comparison with unsupervised methods. (*A*) Three-dimensional protein structures. (*B*) Pearson's correlation coefficient between true and predicted scores for Rosetta, EVmutation, DeepSequence, linear regression (LR), fully connected network (FC), sequence convolutional network (CNN), and graph convolutional network (GCN). EVmutation (I) refers to the independent formulation of the model that does not include pairwise interactions. EVmutation (E) refers to the epistatic formulation of the model that does include pairwise interactions. Each point corresponds to one of seven random train–tune–test splits. (*C*) Correlation performance of supervised models trained with reduced training set sizes. (*D*) Model performance when making predictions for variants containing mutations that were not seen during training (mutational extrapolation). Each point corresponds to one of six replicates, and the red vertical lines denote the medians. (*E*) The fraction of the true 100 best-scoring variants identified by each model's ranking of variants with the given budget. The random baseline is shown with the mean and a 95% CI.

Table 1. Deep mutational scanning datasets

	Description	Organism	Molecular function	Selection	Length	Variants	Ref.
avGFP	Green fluorescent protein	*A. victoria*	Fluorescence	Brightness	237	54,024	16
Bgl3	β-glucosidase	*Streptococcus* sp.	Hydrolysis of β-glucosidic linkages	Enzymatic activity	501	26,653	17
GB1	Protein G B1 domain	*Streptococcus* sp.	IgG binding	IgG-Fc binding	56	536,084	15
Pab1	Pab1 RNA recognition motif (RRM) domain	*Saccharomyces cerevisiae*	Poly(A) binding	Messenger RNA (mRNA) binding	75	40,852	18
Ube4b	Ubiquitination factor E4B U-box domain	*Mus musculus*	Ubiquitin-activating enzyme activity	Ubiquitin ligase activity	102	98,297	19

We evaluated the models on deep mutational scanning datasets representing proteins of varying sizes, folds, and functions.

Rosetta (20), EVmutation (1), and DeepSequence (2), which are not trained using the deep mutational scanning data. Our supervised learning approach achieves superior performance to these other methods on all five protein datasets, demonstrating the benefit of training directly on sequence–function data (Fig. 2B). This result is unsurprising because supervised models are tailored to the specific protein property and sequence distribution in the dataset. Rosetta predictions consider the energetics of the protein structure and therefore do not capture the more specific aspects of protein function. Unsupervised methods such as EVmutation and DeepSequence are trained on natural sequences and thus only capture aspects of protein function directly related to natural evolution. Despite their lower performance, physics-based and unsupervised methods have the benefit of not requiring large-scale sequence–function data, which are often difficult and expensive to acquire.

The different supervised models displayed notable trends in predictive performance across the datasets. The nonlinear models outperformed linear regression, especially on variants with low scores (*SI Appendix*, Fig. S2); high epistasis (*SI Appendix*, Fig. S3); and in the case of avGFP, larger numbers of mutations (*SI Appendix*, Fig. S4). The three nonlinear models performed similarly when trained and evaluated on the full training and testing sets. However, the convolutional networks achieved a better mean squared error when evaluating single-mutation variants in Pab1 and GB1 (*SI Appendix*, Fig. S4). For most proteins, the convolutional networks also had superior performance when trained on smaller training sets (Fig. 2C).

The quantitative evaluations described thus far involve test set variants that have similar characteristics to the training data. We also tested the ability of the models to extrapolate to more challenging test sets. In mutational extrapolation, the model makes predictions for variants containing mutations that were not seen during training. The model must generalize based on mutations that may occur in the same or other positions. The convolutional networks achieved strong performance for one dataset ($r > 0.9$), moderate performance for two additional datasets ($r > 0.6$), and outperformed linear regression and fully connected networks across all datasets (Fig. 2D). In positional extrapolation, the model makes predictions for variants containing mutations in positions that were never modified in the training data. The performance of all models is drastically reduced (*SI Appendix*, Fig. S5), highlighting the difficulty of this task (21). In theory, the parameter sharing inherent to convolutional networks allows them to generalize the effects of mutations across sequence positions. This capability may explain the convolutional networks' superior performance with reduced training set sizes and mutational extrapolation. However, it is still difficult for the convolutional networks to perform well when there are no training examples of mutations in a particular position, such as in positional extrapolation.

The sequence convolutional and graph convolutional networks displayed similar performance across all evaluation metrics, despite the inclusion of three-dimensional protein structure information in the graph topology. To assess the impact of integrating protein structure in the neural network architecture, we created graph convolutional networks with misspecified baseline graphs that were unrelated to the protein structure. These baseline graphs include shuffled, disconnected, sequential, and complete graph structures (*SI Appendix*, Fig. S6). We found that networks trained using these misspecified baseline graphs had accuracy similar to networks trained with the actual protein structure graph, indicating that protein structure information is contributing little to the model's performance (*SI Appendix*, Fig. S7). We also trained the convolutional networks with and without a final fully connected layer and found that this fully connected layer was more important than a correctly specified graph structure. In almost all cases, this final fully connected layer helps overcome the misspecified graph structure (*SI Appendix*, Fig. S7). Overall, these results suggest that the specific convolutional window is not as critical as sharing parameters across different sequence positions and integrating information with a fully connected layer.

The goal of protein engineering is to identify optimized proteins, and models can facilitate this process by predicting high-activity sequences from an untested pool of sequences. Pearson's correlation captures a model's performance across all variants, but it does not provide information regarding a model's ability to retrieve and rank high-scoring variants. We evaluated each model's ability to predict the highest-scoring variants within a given experimental testing budget (Fig. 2E). We calculated recall by asking each model to identify the top N variants from the test set, where N is the budget, and evaluating what fraction of the true top 100 variants was covered in this predicted set. The supervised models consistently achieve higher recall than Rosetta and the unsupervised methods, although the differences are small for Pab1 and Bgl3. In practice, the budget depends on the experimental costs of synthesizing and evaluating variants of the given protein. For GB1, a feasible budget may be 100 variants, and the supervised models can recall over 60% of the top 100 sequences with that budget.

Another important performance metric for protein engineering is the ability to prioritize variants that have greater activity than the wild-type protein. We calculated the mean and maximum scores of the top N predicted test set variants ranked by each model (*SI Appendix*, Figs. S8 and S9). We find that the variants prioritized by the supervised models have greater functional scores than the wild type on average, even when considering variants ranked beyond the top thousand sequences for some datasets. In contrast, Rosetta and the unsupervised models generally prioritize variants with mean scores worse than the wild type. The maximum score of the prioritized variants is also important because it represents the best variant suggested by the model. We find that nearly all models are able to prioritize variants with a maximum score greater than the wild type. The relative performance of each model is dependent on the dataset. Notably, the unsupervised methods perform very well on Bgl3, with EVmutation identifying the top variant with a budget of 20.

Meanwhile, the supervised methods perform very well on Ube4b, prioritizing a variant with the true maximum score with a budget as small as five variants.

Role of Data Quality in Learning Accurate Sequence–Function Models. The performance of the supervised models varied substantially across the five protein datasets. For example, the Pearson correlation for the Bgl3 models was ∼0.4 lower than the GB1 models. Although it is possible some proteins and protein families are intrinsically more difficult to model, practical considerations, such as the size and quality of the deep mutational scanning dataset, could also affect protein-specific performance. Deep mutational scanning experiments use a high-throughput assay to screen an initial gene library and isolate variants with a desired functional property. The initial library and the isolated variants are sequenced, and a fitness score is computed for each variant based on the frequency of reads in both sets. The quality of the calculated fitness scores depends on the sensitivity and specificity of the high-throughput assay, the number of times each variant was characterized in the high-throughput assay, and the number of DNA sequencing reads per variant. If any one of these factors is too low, the resulting fitness scores will not reflect the true fitness values of the characterized proteins, which will make it more difficult for a model to learn the underlying sequence to function mapping.

We assessed how experimental factors influence the success of supervised learning by resampling the full GB1 dataset to generate simulated datasets with varying protein library sizes and numbers of DNA sequencing reads. The library size is the number of unique variants screened in the deep mutational scan. The GB1 dataset is ideal for this analysis because it contains most of the possible single and double mutants and has a large number of sequencing reads per variant. We trained sequence convolutional models on each simulated dataset and tested each network's predictions on a "true" non-resampled test set (Fig. 3). Models trained on simulated datasets with small library sizes performed poorly because there were not sufficient examples to learn the sequence–function mapping. This result is expected and is in line with the performance of models trained on reduced training set sizes on the original GB1 dataset (Fig. 2C). Interestingly, we also found that datasets with large library sizes

Fig. 3. Trade-off between library size and the number of sequencing reads. Performance of sequence convolutional models trained on GB1 datasets that have been resampled to simulate different combinations of protein library size and number of sequencing reads in the deep mutational scan. An "X" signifies that the combination of library size and number of reads produced a dataset with fewer than 25 variants and was, therefore, excluded from the experiment. Having a large library size can be detrimental to supervised model performance if there are not enough reads to calculate reliable functional scores.

can perform poorly if there are not sufficient DNA sequencing reads to reliably estimate the frequency of each variant. This highlights a trade-off between the number of sequence–function examples in a dataset and the quality of its fitness scores. Given a fixed sequencing budget, there exists an optimal intermediate library size that balances these two competing factors. The Bgl3 dataset's poor performance may be the result of having too many unique variants without sufficient sequencing coverage, resulting in a low number of reads per variant and therefore unreliable fitness scores. Future deep mutational scanning libraries could be designed to maximize their size and diversity while ensuring that each variant will have sufficient reads within sequencing throughput constraints.

Learned Models Provide Insight into Protein Structure and Mechanism. Our neural networks transform the original amino acid features through multiple layers to map to an output fitness value. Each successive layer of the network constructs new latent representations of the sequences that capture important aspects of protein function. We can visualize the relationships between sequences in these latent spaces to reveal how the networks learn and comprehend protein function. We used Uniform Manifold Approximation and Projection (UMAP) (22) to visualize test set sequences in the latent space at the last layer of the GB1 sequence convolutional network (Fig. 4A). The latent space organizes the test set sequences based on their functional score, demonstrating that the network's internal representation, which was learned to predict function of the training set examples, also generalizes to capture the sequence–function relationship of the new sequences. The latent space features three prominent clusters of low-scoring variants that may correspond to different mechanisms of disrupting GB1 function. Two clusters, referred to as "G1" and "G2," contain variants with mutations in core residues near the protein's N and C termini, respectively (*SI Appendix*, Fig. S10). Mutations at these residues may disrupt the protein's structural stability and thus decrease the activity measured in the deep mutational scanning experiment (15). Residue cluster "G3" contains variants with mutations at the IgG binding interface, and these likely decrease activity by disrupting key binding interactions. This clustering of variants based on different molecular mechanisms suggests the network is learning biologically meaningful aspects of protein function.

We can also use the neural network models to understand which sequence positions have the greatest influence on protein function. We computed integrated gradients attributions (23) for all training set variants in Pab1 and mapped these values onto the three-dimensional structure (Fig. 4B). Pab1's sequence positions display a range of attributions spanning from negative to positive, where a negative attribution indicates that mutations at that position decrease the protein's activity. Residues at the RNA binding interface tend to display negative attributions, with the key interface residue N127 having the largest negative attribution. The original deep mutational scanning study found that residue N127 cannot be replaced with any other amino acid without significantly decreasing Pab1 binding activity (18). Position D151 has one of the largest positive attributions, which is consistent with the observation that aspartic acid (D) is uncommon at position 151 in naturally occurring Pab1 sequences (18). The sequence convolutional network is able to learn biologically relevant information directly from raw sequence–function data, without the need to specify detailed molecular mechanisms.

Finally, we used the Pab1 sequence convolutional network to make predictions for all possible single-mutation variants (Fig. 4C). The resulting heat map highlights regions of the Pab1 sequence that are intolerant to mutations and shows that mutations to proline are deleterious across most sequence positions. It also demonstrates the network's ability to predict scores for amino acids that were not directly observed in the

Gelman et al.
Neural networks to learn protein sequence–function relationships from deep mutational scanning data

PNAS | 5 of 12
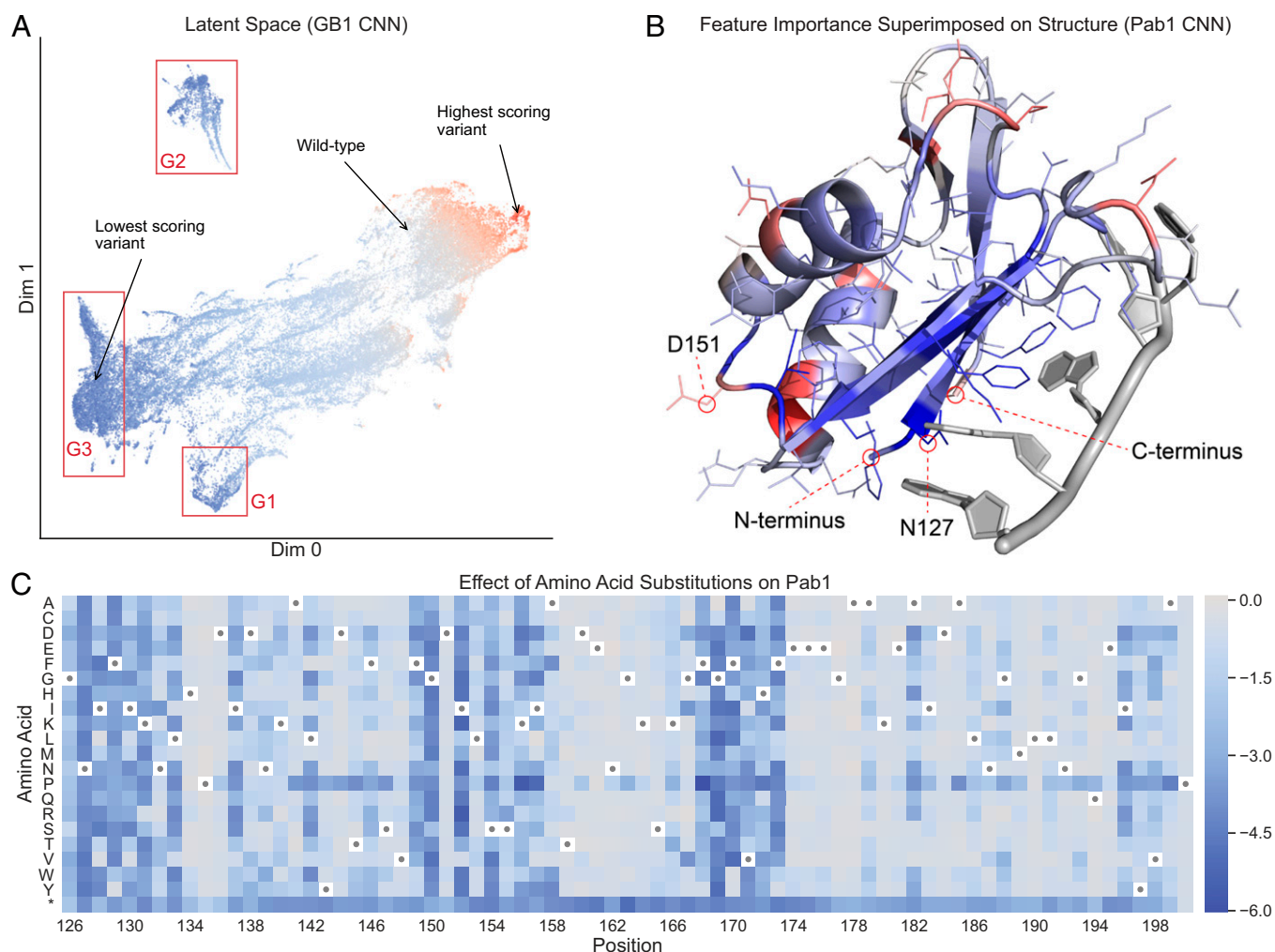https://doi.org/10.1073/pnas.2104878118

Fig. 4. Neural network interpretation. (*A*) A UMAP projection of the latent space of the GB1 sequence convolutional network (CNN), as captured at the last internal layer of the network. In this latent space, similar variants are grouped together based on the transformation applied by the network to predict the functional score. Variants are colored by their true functional score, where red represents high-scoring variants and blue represents low-scoring variants. The clusters marked G1 and G2 correspond to variants with mutations at core residues near the start and end of the sequence, respectively. Cluster G3 corresponds to variants with mutations at surface interface residues. (*B*) Integrated gradients feature importance values for the Pab1 CNN, aggregated at each sequence position and superimposed on the protein's three-dimensional structure. Blue represents positions with negative attributions, meaning mutations in those positions push the network to output lower scores, and red represents positions with positive attributions. (*C*) A heat map showing predictions for all single mutations from the Pab1 CNN. Wild-type residues are indicated with dots, and the asterisk is the stop codon. Most single mutations are predicted to be neutral or deleterious.

dataset. The original deep mutational scan characterized 1,244 single-mutation variants, yet the model can make predictions for all 1,500 possible single-mutation variants. For example, mutation F170P was not experimentally observed, but the model predicts it will be deleterious because proline substitutions at other positions are often highly deleterious. This generalization to amino acid substitutions not observed in the data is only possible with models that share parameters across sequence positions.

Designing Distant Protein Sequences with Learned Models. Our trained neural networks describe the mapping from sequence to function for a given protein family. These models can be used to design new sequences that were not observed in the original deep mutational scanning dataset and may have improved function. The protein design process involves extrapolating a model's predictions to distant regions of sequence space. Because the models were trained and evaluated only on sequences with local changes with respect to the wild type, it is unclear how these out-of-distribution predictions will perform.

We tested the ability of our supervised models to generalize beyond the training data by designing a panel of new GB1 variants with varying distances from the wild-type sequence (Fig. 5*A*). GB1 is a small 8-kDa domain from streptococcal protein G that binds to the fragment crystallizable (Fc) domain of mammalian IgG. GB1's structure is composed of one α-helix that is packed into a four-stranded β-sheet. GB1's interaction with IgG is largely mediated by residues in the α-helix and third β-strand.

The design process was guided by an ensemble of the four models (linear regression and fully connected, sequence convolutional, and graph convolutional networks) to fully leverage different aspects of the sequence–function mapping captured by each model. We used a random-restart hill-climbing algorithm to search over GB1 sequence space for designs that maximize the minimum predicted fitness over the four models. Maximizing the minimum predicted fitness over the four models ensures that every model predicts the designed sequences to have high fitness. We applied this sequence optimization method to design five GB1 variants with increasing numbers of mutations (10, 20, 30, 40, 50) from the wild type, representing sequence identities
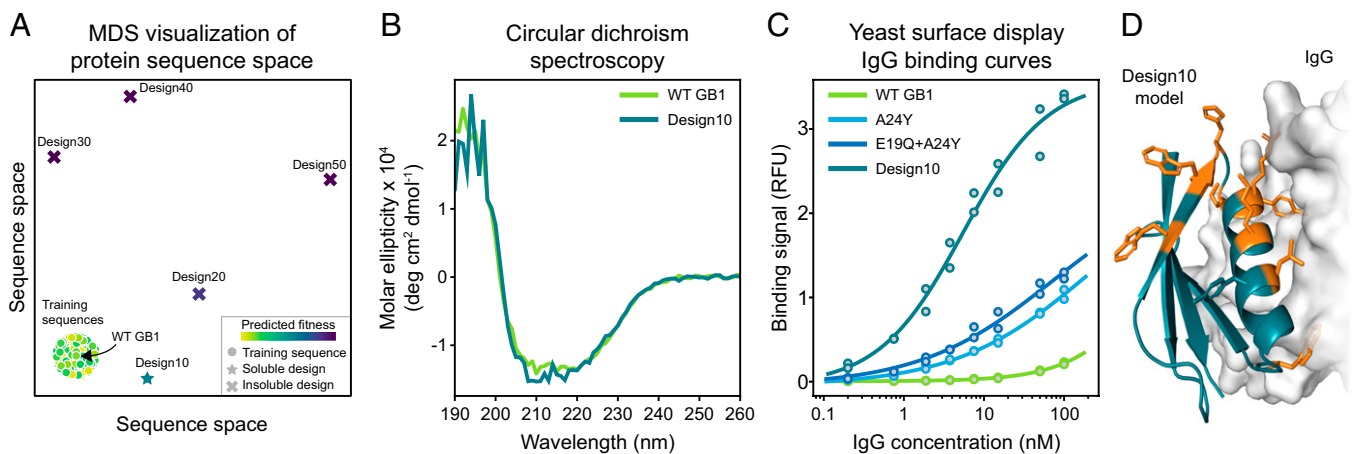
Fig. 5. Neural network–based protein design. (*A*) Multidimensional scaling (MDS) sequence space visualization of wild-type (WT) GB1 sequence, the GB1 training sequences, and the five designed proteins. Design10 to Design50 are progressively farther from the training distribution. Design10 is expressed as a soluble protein, while the more distant designs were insoluble. (*B*) Circular dichroism spectra of purified wild-type GB1 and Design10. Both proteins display highly similar spectra that are indicative of α-helical protein structures. (*C*) IgG binding curves of wild-type GB1 variants. Design10 displays substantially higher binding affinity than wild-type GB1, A24Y, and E19Q + A24Y. All measurements were performed in duplicate. Binding signal is reported in relative fluorescence units (RFU). (*D*) The locations of Design10's 10 mutations (shown in orange) relative to the IgG binding interface. The Design10 structure was predicted de novo using Rosetta.

spanning from 82 to 11% (*SI Appendix*, Table S1). We expect designs with fewer mutations to be more likely to fold and function because they are more similar to the training data.

We experimentally tested the five GB1 designs by synthesizing their corresponding genes and expressing them in *Escherichia coli*. We found that the 10-mutant design, referred to as Design10, was expressed as a soluble protein, but the more distant designs were insoluble (Fig. 5*A*). We were unable to further characterize Design20 to Design50 because their insoluble expression prevented downstream protein purification. We performed circular dichroism spectroscopy on Design10 and found that it had nearly identical spectra to wild-type GB1, suggesting they have similar secondary structure content (Fig. 5*B*).

The original GB1 deep mutational scan measured binding to the Fc region of IgG; therefore, our supervised models should capture a variant's binding affinity. We tested Design10's ability to bind to IgG using a yeast display binding assay. We also tested wild-type GB1 and the top single (A24Y) and double (E19Q + A24Y) mutants from the original deep mutational scanning dataset. We found that Design10 binds to IgG with a K_d of 5 nM, which is substantially higher affinity than wild-type GB1, A24Y, or E19Q + A24Y (Fig. 5*C*). We were unable to precisely determine wild-type GB1, A24Y, or E19Q + A24Y dissociation constants because our assay could not reliably measure binding at IgG concentrations above 100 nM. The data showed qualitative trends where wild type had the lowest affinity, followed by A24Y and then, E19Q + A24Y. Our measurements indicate that wild-type GB1's K_d is well above 100 nM, which is consistent with measurements from the literature that have found that this interaction is in the 250- to 900-nM range (24, 25). Based on our estimates and others' previous measurements, we conservatively estimate that Design10 binds human IgG with at least 20-fold higher affinity than wild-type GB1.

Closer inspection of the Design10 sequence revealed that it was not simply composed of the top 10 single mutations for enrichment and even included 4 mutations whose individual effects on the predicted functional score ranked below the top 300. In addition, Design10's predicted score was more than two times greater than the variant comprising the top 10 single mutations. This highlights the ability of the design process to capture nonlinear interactions and leverage synergies between sites. We also evaluated the robustness of our findings by rerunning the

10-mutant design process 100 independent times and evaluating the diversity of the designs (*SI Appendix*, Table S2).

We built a model of Design10's three-dimensional structure using Rosetta de novo structure prediction (Fig. 5*D*). Design10's predicted structure aligns to the wild-type GB1 crystal structure with 0.9 Å Cα rmsd. Design10's actual structure is likely very similar to wild-type GB1 given their high sequence identity, similar circular dichroism spectra, and the small deviation between Design10's de novo predicted structure and the experimental GB1 structure. Inspection of Design10's predicted structure revealed that many of its mutations were concentrated near the IgG binding interface, and this may help to explain its large increase in IgG binding affinity. We also evaluated Rosetta models for Design20 to Design50 and found no obvious reasons why they failed to express.

Discussion

We have presented a supervised learning framework to infer the protein sequence–function mapping from deep mutational scanning data. Our supervised models work best when trained with large-scale datasets, but they can still outperform physics-based and unsupervised prediction methods when trained with only hundreds of sequence–function examples. Unsupervised methods remain appealing for proteins with very little or no sequence–function data available. Among the supervised models, linear regression displayed the lowest performance due to its inability to represent interactions between multiple mutations. Despite that limitation, linear regression still performed fairly well because mutations often combine in an additive manner (26). The convolutional networks outperformed linear regression and fully connected networks when trained with fewer training examples and when performing mutational extrapolation. The parameter sharing inherent to convolutional networks can improve performance by allowing generalization of the effects of mutations across different sequence positions. However, in the five datasets we tested, even the convolutional networks could not accurately generalize when entire sequence positions were excluded from the training data. It was surprising that graph convolutions that incorporate protein structure did not improve performance over sequence-based convolutions. The comparable performance could be the result of the networks' ability to compensate with fully connected layers, the lack of sequence

Gelman et al.
Neural networks to learn protein sequence–function relationships from deep mutational scanning data

PNAS | 7 of 12
https://doi.org/10.1073/pnas.2104878118

diversity in the deep mutational scanning data, or the specific type of graph neural network architecture used. We are unable to determine which of these factors had the greatest influence.

Our analysis of how data quality influences the ability to learn the sequence–function mapping can be considered when designing future deep mutational scanning experiments. We found that a model's predictive performance is determined not only by the number of sequence–function training examples but also by the quality of the estimated functional scores. Therefore, in a deep mutational scanning experiment, it may be preferable to limit the total number of unique variants analyzed to ensure that each variant has sufficient sequencing reads to calculate accurate functional scores. Any missing mutations can then be imputed with a convolutional network to overcome the smaller dataset size.

Recent studies have examined supervised learning methods capable of scaling to deep mutational scanning datasets. One study benchmarked combinations of supervised learning methods and protein sequence encodings (7). Consistent with our results, it found that sequence convolutional neural networks with amino acid property-based features tended to perform better than alternatives. Some algorithms specialize in modeling epistasis. Epistatic Net (27) introduced a neural network regularization strategy to limit the number of epistatic interactions. Other approaches focused on the global epistasis that arises due to a nonlinear transformation from a latent phenotype to the experimentally characterized function (28, 29). Protein engineering with UniRep (30) showed that general global protein representations can support training function-specific supervised models with relatively few sequence–function examples. ECNet pioneered an approach for combining global protein representations, local information about residue coevolution, and protein sequence features (31). Across tens of deep mutational scanning datasets, ECNet was almost always superior to unsupervised learning models and models based only on a global protein representation. Future work can explore how to best combine global protein representations, local residue coevolution features, and graph encodings of protein structure to learn predictive models for specific protein functions, including for proteins that have little experimental data available. Despite their similar performance to sequence convolutional networks in our study, graph convolutional networks that integrate three-dimensional structural information remain enticing because of successes on other protein modeling tasks (32–35) and rapid developments in graph neural network architectures (36).

Another challenging future direction will be assessing how well trained models extrapolate to sequences with higher-order mutations (21, 37). As a proof of concept, we designed distant GB1 sequences with tens of mutations from the wild type. The 10-mutant design (Design10) had substantially stronger IgG binding affinity than wild-type GB1, but the four sequences with more mutations did not express as soluble proteins. The tremendous success of Design10 is encouraging considering how few designed sequences we tested and the many opportunities to improve upon our limited exploration of model-guided design. The model predictions can be improved through more sophisticated ensembling and uncertainty estimation. Our hill-climbing sequence optimization strategy can be replaced by specialized methods that allow supervised models to efficiently explore new parts of a sequence space (38–41).

Machine learning is revolutionizing our ability to model and predict the complex relationships between protein sequence, structure, and function (42, 43). Supervised models of protein function are currently limited by the availability and quality of experimental data but will become increasingly accurate and general as researchers continue to experimentally characterize protein sequence space (44). Other important machine learning advances relevant to protein engineering include generative modeling to sample nonnatural protein sequences (34, 45, 46), language models to learn protein representations from diverse natural sequences (47–50), and strategies to incorporate machine learning predictions into directed evolution experiments (51–53). These approaches are enabling the next generation of data-driven protein engineering.

Materials and Methods

Datasets. We tested our supervised learning approach on five deep mutational scanning datasets: avGFP (16), Bgl3 (17), GB1 (15), Pab1 (18), and Ube4b (19). We selected these publicly available datasets because they correspond to diverse proteins and contain variants with multiple amino acid substitutions. The avGFP, Pab1, and Ube4b datasets were published with precomputed functional scores, which we used directly as the target scores for our method. For GB1 and Bgl3, we computed functional scores from raw sequencing read counts using Enrich2 (54). We filtered out variants with fewer than five sequencing reads and ran Enrich2 using the "Log Ratios (Enrich2)" scoring method and the "Wild Type" normalization method. Table 1 shows additional details about the datasets.

Protein Sequence Encoding. We encoded each variant's amino acid sequence using a sequence-level encoding that supports multiple substitutions per variant. Each amino acid is featurized with its own feature vector, and the full encoded variant consists of the concatenated amino acid feature vectors. We featurize each amino acid using a two-part encoding made up of a one-hot encoding and an amino acid index (AAIndex) encoding. One-hot encoding captures the specific amino acid at each position. It consists of a length 21 vector where each position represents one of the possible amino acids or the stop codon. All positions are zero except the position of the amino acid being encoded, which is set to a value of one. AAindex encoding captures physicochemical and biochemical properties of amino acids from the AAindex database (55). These properties include simple attributes, such as hydrophobicity and polarity, as well as more complex characteristics, such as average nonbonded energy per atom and optimized propensity to form a reverse turn. In total, there are 566 such properties that were taken from literature. These properties are partially redundant because they are aggregated from different sources. Therefore, we used principle component analysis to reduce the dimensionality to a length 19 vector, capturing 100% of the variance. We concatenated the one-hot and AAindex encodings to form the final representation for each amino acid. One benefit of this encoding is that it enables the use of convolutional networks, which leverage the spatial locality of the raw inputs to learn higher-level features via filters. Other types of encodings that do not have a feature vector for each residue, such as those that embed full amino acid sequences into fixed-size vectors, would not be as appropriate for convolutional networks because they do not have locality in the input that can be exploited by convolutional filters.

Convolutional Neural Networks. We tested two types of convolutional neural networks: sequence convolutional and graph convolutional. These networks extract higher-level features from the raw inputs using convolutional filters. Convolutional filters are sets of learned weights that identify patterns in the input and are applied across different parts of the input. The filters can output higher or lower values depending on whether the given input matches the pattern that the filters have learned to identify. We implemented a sequence convolutional network where the input is a one-dimensional amino acid sequence. The network applies filters using a sliding window across the input sequence, integrating information from amino acid sequence neighbors. The network applies filters at all valid sequence positions and does not pad the ends of the sequence with zeros.

Graph convolutional neural networks are similar to traditional convolutional networks, except graph convolutional networks operate on arbitrary graph structures rather than linear sequences or two-dimensional grids. Graph filters still capture spatial relationships in the input data, but those relationships are determined by neighboring nodes in the graph rather than neighboring characters in a sequence or neighboring pixels in a two-dimensional grid. In our case, we use a graph derived from the protein's wild-type three-dimensional structure. This allows the network to more easily learn features that correspond to patterns of amino acid residues that are nearby in physical space.

We use the order-independent graph convolution operator described by Fout et al. (32). It is considered order independent because it does not impose an ordering on neighbor nodes. In an order-dependent formulation, different neighbor nodes would have different weights, but in the order-independent formulation, all neighbor nodes are treated identically and

share the same weights. Each filter consists of a weight vector for the center node and a weight vector for the neighbor nodes that is shared among the neighbor nodes. For a set of filters, the output z_i at a center node i is calculated using Eq. **1**, where W_C is the center node's weight matrix, W_N is the neighbor nodes' weight matrix, and b is the vector of biases, one for each filter. Additionally, x_i is the feature vector at node i, N_i is the set of neighbors of node i, and σ is the activation function:

$$z_i = \sigma\left(W_C \cdot x_i + \frac{1}{|N_i|}\sum_{j \in N_i}(W_N \cdot x_j) + b\right). \qquad [1]$$

In this formulation, a graph consisting of nodes and edges is incorporated into each convolutional layer. Input features are placed at the graph's nodes in the first layer. Outputs are computed at the node level using input features from a given center node and corresponding neighbor nodes. Because output is computed for each node, graph structure is preserved between subsequent graph layers. The incoming signal from neighbor nodes is averaged to account for the variable numbers of neighbors. The window size of the filter is limited to the immediate neighbors of the current center node. Information from more distant nodes is incorporated through multiple graph convolutional layers. The final output of the network is computed at the graph level with a single function score prediction for the entire graph.

Protein Structure as a Graph. We encoded each protein's wild-type structure as a graph and incorporated it into the architecture of the graph convolutional neural network (Fig. 1D). The protein structure graph is an undirected graph with a node for each amino acid residue and an edge between nodes if the residues are within a specified distance threshold in three-dimensional space. The distance threshold is a hyperparameter with a range of 4 to 10 Å and was selected independently for each dataset during the hyperparameter optimization. We measure distances between residues via distances of the β-carbon atoms (Cβ) in angstroms. The protein structure graph for GB1 is based on Protein Data Bank (PDB) structure 2QMT. The protein structure graphs for the other four proteins are based on structures derived from Rosetta comparative modeling, using the RosettaCM protocol (56) with the default options. For the comparative modeling, we selected template structures from PDB that most closely matched the reference sequence of the deep mutational scanning data. In addition to the standard graph based on the protein's structure, we tested four baseline graphs: a shuffled graph based on the standard graph but with shuffled node labels, a disconnected graph with no edges, a sequential graph containing only edges between sequential residues, and a complete graph containing all possible edges (*SI Appendix*, Fig. S6). We used NetworkX (57) v2.3 to generate all protein structure and baseline graphs.

Complete Model Architectures. We implemented linear regression and three types of neural network architectures: fully connected, sequence convolutional, and graph convolutional. Linear regression is implemented as a fully connected neural network with no hidden layers. It has a single output node that is fully connected to all input nodes. The other networks all have multiple layers. The fully connected network consists of some number of fully connected layers, and each fully connected layer is followed by a dropout layer with a 20% dropout probability. Finally, there is a single output node. The sequence and graph convolutional networks consist of some number of convolutional layers, a single fully connected layer with 100 hidden nodes, a dropout layer with a 20% dropout probability, and a single output node. We also trained sequence and graph convolutional networks without the fully connected layer or dropout layer for the analyses in *SI Appendix*, Fig. S7. We used the leaky rectified linear unit as the activation function for all hidden layers. A hyperparameter sweep determined the other key aspects of the model architectures, such as the number of layers, the number of filters, and the kernel size of filters (*SI Appendix*, Fig. S11). We used Python v3.6 and TensorFlow (58) v1.14 to implement the models.

Model Training. We trained the networks using the Adam optimizer and mean squared error loss. We set the Adam hyperparameters to the defaults except for the learning rate and batch size, which were selected using hyperparameter sweeps. We used early stopping for all model training with a patience of 15 epochs and a minimum delta (the minimum amount by which loss must decrease to be considered an improvement) of 0.00001. We set the maximum possible number of epochs to 300. The original implementation overweighted the last examples in an epoch when calculating the tuning set loss. This could have affected early stopping but had little to no effect in practice. We trained the networks on graphics processing units (GPUs) available at the University of Wisconsin–Madison via the Center for High Throughput Computing and the workload management system HTCondor

(59). We also used GPU resources from Argonne National Laboratory's Cooley cluster. The GPUs we used included Nvidia GeForce GTX 1080 Ti, GeForce RTX 2080 Ti, and Tesla K80.

Main Experiment Setup. We split each dataset into random training, tuning, and testing sets. The tuning set is sometimes referred to as the validation set and is used for hyperparameter optimization. This allowed us to train the models; tune hyperparameters; and evaluate performance on separate, nonoverlapping sets of data. The training set was 81% of the data, the tuning set was 9%, and the testing set was 10%. This strategy supports the objective of training and evaluating models that fully leverage all available sequence–function data and make predictions for variants that have characteristics similar to the training data. There are other valid strategies that more directly test the ability of a model to generalize to mutations or positions that were not present in the training data, which we describe below.

We performed a hyperparameter grid search for each dataset using all possible combinations of the hyperparameters in *SI Appendix*, Fig. S11. The hyperparameters selected for one dataset did not influence the hyperparameters selected for any other dataset. For each type of supervised model (linear regression, fully connected, sequence convolutional, and graph convolutional), we selected the set of hyperparameters that resulted in the smallest mean squared error on the tuning set. The selected hyperparameters are listed in *SI Appendix*, Table S3, and the number of trainable parameters in each selected model is listed in *SI Appendix*, Table S4. This is the main set of hyperparameters. For any subsequently trained models, such as those with reduced training sets, we performed smaller hyperparameter sweeps to select a learning rate and batch size, but all other hyperparameters that specify the network architecture were set to those selected in the main run.

To assess the robustness of the original train–tune–test splits, we created six additional random splits for each dataset. We tuned the learning rate and batch size independently for each split; however, the network architectures were fixed to those selected using the original split. There was a risk of overestimating performance on the new splits because the data used to tune the architectures from the original split may be present in the test sets of the new splits. However, the results showed no evidence of this type of overfitting. We report the performance on these new splits in Fig. 2B and *SI Appendix*, Fig. S1A. All other experiments used the original train–tune–test split.

For the random baseline in *SI Appendix*, Figs. S8 and S9, we generated 1,000 random rankings of the entire test set. Then, for each ranking threshold N, we selected the first N variants from each ranking as the prioritized variants. We computed the mean (*SI Appendix*, Fig. S8) and the maximum (*SI Appendix*, Fig. S9) of each random ranking's prioritized variants. Finally, we show the 95% CI calculated as $\pm$ 1.96 times the SD.

Reduced Training Size Setup. For the reduced training size experiment, we used the same 9% tuning and 10% testing sets as the main experiment. The reduced training set sizes were determined by percentages of the original 81% training pool. For each reduced size, we sampled five random training sets of the desired size from the 81% training pool. These replicates are needed to mitigate the effects of an especially strong or weak training set, which could be selected by chance, especially with the smaller training set sizes. We reported the median metrics of these five replicates.

Mutational and Positional Extrapolation. We tested the ability of the models to generalize to mutations and positions not present in the training data using two dataset splitting strategies referred to as mutational and positional extrapolation. For each of these splitting strategies, we created six replicate train–tune–test splits and tuned the learning rate and batch size independently for each split. We report the Pearson's correlation on the test set for each split in Fig. 2D and *SI Appendix*, Fig. S5.

For mutational extrapolation, we designated 80% of single mutations present in the dataset as training and 20% as testing. We then divided the variants into three pools: training, testing, or overlap, depending on whether the variants contained only mutations designated as training, only mutations designated as testing, or mutations from both sets. We discarded the variants in the overlap pool to ensure there was no informational overlap between the training and testing data. We split the training pool into a 90% training set and a 10% tuning set. We used 100% of the variants in the testing pool as the test set.

For positional extrapolation, we followed a similar procedure as mutational extrapolation. We designated 80% of sequence positions as training and 20% as testing. We divided variants into training, testing, and overlap pools, depending on whether the variants contained mutations only in positions designated as training, only in positions designated as testing, or both in positions designated as training and testing. We discarded the

Gelman et al.
Neural networks to learn protein sequence–function relationships from deep mutational scanning data

PNAS | 9 of 12
https://doi.org/10.1073/pnas.2104878118

variants in the overlap pool, split the training pool into a 90% training set and a 10% tuning set, and used 100% of the variants in the testing pool as the test set.

Comparison with EVmutation and DeepSequence. We generated multiple sequence alignments using the EVcouplings web server (60) according to the protocol described for EVmutation (1). We used Jackhmmer (61) to generate an initial alignment with five search iterations against UniRef100 (62) and a sequence inclusion threshold of 0.5 bits per residue. If the alignment had < 80% sequence coverage, we increased the threshold in steps of 0.05 bits per residue until coverage was ≥ 80%. If the number of effective sequences in the alignment was < 10 times the length of the sequence, we decreased the threshold until the number of sequences was ≥ 10 times the length. If the objectives were conflicting, we gave priority to the latter. We set all other parameters to EVcouplings defaults. We trained EVmutation via the "mutate" protocol from EVcouplings. We executed EVmutation locally using EVcouplings v0.0.5 with configuration files generated by the EVcouplings web server. We trained DeepSequence using the same fixed architecture and hyperparameters described in the original work (2). We fit a DeepSequence model to each alignment and calculated the mutation effect prediction using 2,000 evidence lower-bound samples.

Comparison with Rosetta. We computed Rosetta scores for every variant using Rosetta's FastRelax protocol with the talaris2014 score function (Rosetta v3.10). First, we created a base structure for each wild-type protein. We generated 10 candidate structures by running relax on the same structure used to generate the protein structure graph, described above. We selected the lowest-energy structure to serve as the base. Next, we ran mutate and relax to generate a single structure and compute the corresponding energy for each variant. We set the residue selector to a neighborhood of 10 Å. We took the negative of the computed energies to compute the final score for each variant.

GB1 Resampling Experiment. We performed a resampling experiment on the GB1 dataset to assess how the quality of deep mutational scanning–derived functional scores impacts performance of supervised learning models. In this case, quality refers to the number of sequencing reads per variant used to estimate the fitness scores. The number of reads per variant depends on the number of variants and the total number of reads in the deep mutational scanning experiment. Raw deep mutational scanning data consist of two sets of variants: an input set (prescreening) and a selected set (postscreening). Both sets have associated sequencing read counts for each variant, and the functional score for each variant is calculated from these read counts. We resampled the original GB1 data to generate datasets corresponding to 99 different combinations of protein library size and number of reads (*SI Appendix*, Fig. S12). The library size refers to the number of unique variants being screened in the deep mutational scan. Note that the final dataset may have fewer unique variants than the protein library. This occurs when there is a low number of sequencing reads relative to the size of the library. In that scenario, not all generated variants will get sequenced, even though they were screened as part of the function assay.

First, we created a filtered dataset by removing any variants with zero reads in either the input set or selected set of the original deep mutational scanning data. We generated Enrich2 scores for this filtered dataset using the same approach described in the above section on datasets. We randomly selected 10,000 variants from this dataset to serve as a global testing set. Next, we used the filtered dataset, minus the testing set variants, as a base to create the resampled datasets. For each library size in the heat map in Fig. 3, we randomly selected that many variants from the base dataset to serve as the library. Then, for each library, we created multinomial probability distributions giving the probability of generating a read for a given variant. We created these probability distributions for both the input and selected sets by dividing the original read counts of each variant by the total number of reads in the set. The multinomial distributions allowed us to sample new input and selected sets based on the read counts in the heat map in Fig. 3. To determine how many reads should be sampled from the input set vs. the selected set, we computed the fraction of reads in the input set and selected set in the base dataset and sampled reads based on that fraction. Finally, we generated Enrich2 scores for each resampled dataset using the same approach described in the above section on datasets. To account for potential skewing from random sampling, we generated five replicates for each of the 99 combinations of library size and numbers of reads. Counting the replicates, we created 495 resampled datasets in total.

We trained the supervised learning models on each resampled dataset, as long as the dataset had at least 25 total variants in each of its five replicates.

Of the 99 combinations of library size and number of reads, 7 did not have enough variants across the replicate datasets and were thus excluded from this experiment. Although the libraries of these 7 combinations had more than 25 variants, there were not enough reads to estimate scores for all of them, and thus, the final datasets ended up with less than 25 variants. We split each resampled dataset into 80% training and 20% tuning sets. The tuning sets were used to select the learning rate and batch size hyperparameters. The network architectures and other parameters were set to those selected during the main experiment described above. We evaluated each model using the held-out testing set with non-resampled fitness scores. This type of evaluation ensures that although the models are trained on resampled datasets with potentially unreliable fitness scores, they are evaluated on high-confidence fitness scores from the non-resampled dataset. We report the mean Pearson's correlation coefficient across the five replicates for each combination of library size and number of reads.

UMAP Projection of Latent Space. Each neural network encodes a latent representation of the input in its last internal layer before the output node. The last internal layer in the convolutional networks is a dense fully connected layer with 100 hidden nodes. Thus, the latent representation of each variant at this layer is a length 100 vector. We used UMAP (22) to project the latent representation of each variant into a two-dimensional space to make it easier to visualize while still preserving spatial relationships between variants. We used the umap-learn package v0.4.0 to compute the projection with default hyperparameters (n_neighbors = 15, min_dist = 0.1, and metric = "euclidean"). The two-dimensional visualization shows how the network organizes variants internally prior to predicting a functional score. We colored each variant by its score to show that the network efficiently organizes the variants. Variants grouped close together in the UMAP plot have similar functional scores. We also annotated a few key variants, such as the highest- and lowest-scoring variants.

Integrated Gradients. To determine which input features were important for making predictions, we generated integrated gradients feature attributions (23) for all variants. The attributions quantify the effects of specific feature values on the network's output. A positive attribution means the feature value pushes the network to output a higher score relative to the given baseline, and a negative attribution means the feature value pushes the network to output a lower score relative to the given baseline. We used the wild-type sequence as the baseline input. Integrated gradients attributions are computed on a per-variant basis, meaning attributions are specific to the feature values of the given variant. Due to nonlinear effects captured by the nonlinear models, a given feature value might have a positive attribution in one variant but a negative attribution in a different variant. We computed attributions for all variants in the training set. Examining the training set is analogous to other model interpretation techniques that compute attributions directly from the weights or parameters of models that were trained using training sets. We summed the attributions for all features at each sequence position, allowing us to see which mutations pushed the network to output a higher or lower score for each individual variant. We also summed the attributions across all the variants in the training set to see which sequence positions were typically tolerant or intolerant to mutations. We used DeepExplain (63) v0.3 to compute the integrated gradients attributions with "steps" set to 100.

Model-Guided Design of GB1 Variants. We used a random-restart hill-climbing algorithm to design sequences with a set number of mutations (*n*) from wild-type GB1 that maximized the minimum predicted functional score from an ensemble of four models (linear regression, fully connected, sequence convolutional, and graph convolutional):

$$\arg\max_{x \in \mathbb{S}^n} \min_{model \in LR, FC, CNN, GCN} f_{model}(x),$$

where x is a sequence, $\mathbb{S}^n$ is the set of all sequences n mutations from the wild type, and $f_{model}(x)$ is a model's predicted score for sequence x. This design objective ensures that all models predict that the sequence will have a high functional score. We initialized a hill-climbing run with a randomly selected sequence containing n point mutations and performed a local search by exchanging each of these n mutations with each other possible single-point mutation. Exchanging mutations ensured that we only search sequences a fixed distance n from the wild type. We then moved to the mutation-exchanged variant with the highest objective, which became our new reference point, and repeated this hill-climbing process until a local optimum was reached. We performed each sequence optimization with 10 random initializations and took the design with the highest overall objective value. We applied this procedure to design one sequence at each level of diversity, where $n = 10, 20, 30, 40, 50$. We visualized the sequence space

using multidimensional scaling with the Hamming distance as the distance metric between sequences.

We predicted the three-dimensional structure of Design10 using Rosetta Abinitio (64). We used the Rosetta Fragment Server to generate the fragments for the Design10 sequence. We generated 100 structures using Rosetta 3.12 and AbinitioRelax and selected the structure with the lowest total score. The predicted structure for Design10 aligns to the wild-type GB1 crystal structure with 0.9 Å Cα rmsd.

The experimental methods to characterize these designs are described in *SI Appendix*.

Data Availability. We provide a cleaned version of our code that can be used to retrain the models from this article or train new models with different network architectures or for different datasets. We also provide pretrained models that use the latest code and are functionally equivalent to the ones from this article. The pretrained models can be used to make predictions for new variants. Our code is freely available on GitHub and is licensed under the MIT license (https://github.com/gitter-lab/nn4dms) (65). The software is also archived on Zenodo (https://doi.org/10.5281/zenodo.4118330)

(66). *SI Appendix*, Table S5 shows software dependencies and their versions.

1. T. A. Hopf *et al.*, Mutation effects predicted from sequence co-variation. *Nat. Biotechnol.* **35**, 128–135 (2017).
2. A. J. Riesselman, J. B. Ingraham, D. S. Marks, Deep generative models of genetic variation capture the effects of mutations. *Nat. Methods* **15**, 816–822 (2018).
3. R. J. Fox *et al.*, Improving catalytic function by ProSAR-driven enzyme evolution. *Nat. Biotechnol.* **25**, 338–344 (2007).
4. H. Song, B. J. Bremer, E. C. Hinds, G. Raskutti, P. A. Romero, Inferring protein sequence-function relationships with large-scale positive-unlabeled learning. *Cell Syst.* **12**, 92–101.e8 (2021).
5. P. A. Romero, A. Krause, F. H. Arnold, Navigating the protein fitness landscape with Gaussian processes. *Proc. Natl. Acad. Sci. U.S.A.* **110**, E193–E201 (2013).
6. V. E. Gray, R. J. Hause, J. Luebeck, J. Shendure, D. M. Fowler, Quantitative missense variant effect prediction using large-scale mutagenesis data. *Cell Syst.* **6**, 116–124.e3 (2018).
7. Y. Xu *et al.*, Deep dive into machine learning models for protein engineering. *J. Chem. Inf. Model.* **60**, 2773–2790 (2020).
8. R. Vaser, S. Adusumalli, S. N. Leng, M. Sikic, P. C. Ng, SIFT missense predictions for genomes. *Nat. Protoc.* **11**, 1–9 (2016).
9. I. A. Adzhubei *et al.*, A method and server for predicting damaging missense mutations. *Nat. Methods* **7**, 248–249 (2010).
10. M. Hecht, Y. Bromberg, B. Rost, Better prediction of functional effects for sequence variants. *BMC Genomics* **16** (suppl. 8), S1 (2015).
11. B. Wang, E. R. Gamazon, Modeling mutational effects on biochemical phenotypes using convolutional neural networks: Application to SARS-CoV-2. bioRxiv [Preprint] (2021). https://doi.org/10.1101/2021.01.28.428521 (Accessed 8 February 2021).
12. T. Ching *et al.*, Opportunities and obstacles for deep learning in biology and medicine. *J. R. Soc. Interface* **15**, 20170387 (2018).
13. D. M. Fowler, S. Fields, Deep mutational scanning: A new style of protein science. *Nat. Methods* **11**, 801–807 (2014).
14. T. N. Starr, J. W. Thornton, Epistasis in protein evolution. *Protein Sci.* **25**, 1204–1218 (2016).
15. C. A. Olson, N. C. Wu, R. Sun, A comprehensive biophysical description of pairwise epistasis throughout an entire protein domain. *Curr. Biol.* **24**, 2643–2651 (2014).
16. K. S. Sarkisyan *et al.*, Local fitness landscape of the green fluorescent protein. *Nature* **533**, 397–401 (2016).
17. P. A. Romero, T. M. Tran, A. R. Abate, Dissecting enzyme function with microfluidic-based deep mutational scanning. *Proc. Natl. Acad. Sci. U.S.A.* **112**, 7159–7164 (2015).
18. D. Melamed, D. L. Young, C. E. Gamble, C. R. Miller, S. Fields, Deep mutational scanning of an RRM domain of the *Saccharomyces cerevisiae* poly(A)-binding protein. *RNA* **19**, 1537–1551 (2013).
19. L. M. Starita *et al.*, Activity-enhancing mutations in an E3 ubiquitin ligase identified by high-throughput mutagenesis. *Proc. Natl. Acad. Sci. U.S.A.* **110**, E1263–E1272 (2013).
20. R. F. Alford *et al.*, The Rosetta all-atom energy function for macromolecular modeling and design. *J. Chem. Theory Comput.* **13**, 3031–3048 (2017).
21. A. C. Mater, M. Sandhu, C. Jackson, The NK landscape as a versatile benchmark for machine learning driven protein engineering. bioRxiv [Preprint] (2020). https://doi.org/10.1101/2020.09.30.319780 (Accessed 6 October 2020).
22. L. McInnes, J. Healy, UMAP: Uniform manifold approximation and projection for dimension reduction. arXiv [Preprint] (2020). https://arxiv.org/abs/1802.03426 (Accessed 18 September 2020).
23. M. Sundararajan, A. Taly, Q. Yan, Axiomatic attribution for deep networks. arXiv [Preprint] (2017). https://arxiv.org/abs/1703.01365 (Accessed 13 June 2017).
24. R. K. Jha, T. Gaiotto, A. R. Bradbury, C. E. Strauss, An improved Protein G with higher affinity for human/rabbit IgG Fc domains exploiting a computationally designed polar network. *Protein Eng. Des. Sel.* **27**, 127–134 (2014).
25. H. Watanabe *et al.*, Histidine-mediated intramolecular electrostatic repulsion for controlling pH-dependent protein-protein interaction. *ACS Chem. Biol.* **14**, 2729–2736 (2019).
26. J. A. Wells, Additivity of mutational effects in proteins. *Biochemistry* **29**, 8509–8517 (1990).
27. A. Aghazadeh *et al.*, Epistatic Net allows the sparse spectral regularization of deep neural networks for inferring fitness functions. *Nat. Commun.* **12**, 5225 (2021).
28. A. Tareen *et al.*, MAVE-NN: Learning genotype-phenotype maps from multiplex assays of variant effect. bioRxiv [Preprint] (2021). https://doi.org/10.1101/2020.07.14.201475 (Accessed 27 June 2021).
29. J. Otwinowski, D. M. McCandlish, J. B. Plotkin, Inferring the shape of global epistasis. *Proc. Natl. Acad. Sci. U.S.A.* **115**, E7550–E7558 (2018).
30. S. Biswas, G. Khimulya, E. C. Alley, K. M. Esvelt, G. M. Church, Low-N protein engineering with data-efficient deep learning. *Nat. Methods* **18**, 389–396 (2021).
31. Y. Luo *et al.*, Evolutionary context-integrated deep sequence modeling for protein engineering. bioRxiv [Preprint] (2020). https://doi.org/10.1101/2020.01.16.908509 (Accessed 17 January 2020).
32. A. Fout, J. Byrd, B. Shariat, A. Ben-Hur, "Protein interface prediction using graph convolutional networks" in *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, I. Guyon *et al.*, Eds. (Curran Associates, Inc., Red Hook, NY, 2017), vol. 30, pp. 6530–6539.
33. V. Gligorijević *et al.*, Structure-based protein function prediction using graph convolutional networks. *Nat. Commun.* **12**, 3168 (2021).
34. A. Strokach, D. Becerra, C. Corbi-Verge, A. Perez-Riba, P. M. Kim, Fast and flexible protein design using deep graph neural networks. *Cell Syst.* **11**, 402–411.e4 (2020).
35. S. Sanyal, I. Anishchenko, A. Dagar, D. Baker, P. Talukdar, ProteinGCN: Protein model quality assessment using Graph Convolutional Networks. bioRxiv [Preprint] (2020). https://doi.org/10.1101/2020.04.06.028266 (Accessed 7 April 2020).
36. Z. Wu *et al.*, A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **32**, 4–24 (2021).
37. D. H. Bryant *et al.*, Deep diversification of an AAV capsid protein by machine learning. *Nat. Biotechnol.* **39**, 691–696 (2021).
38. C. Angermueller *et al.*, Population-based black-box optimization for biological sequence design. arXiv [Preprint] (2020). https://arxiv.org/abs/2006.03227 (Accessed 11 July 2020).
39. C. Fannjiang, J. Listgarten, Autofocused oracles for model-based design. arXiv [Preprint] (2020). https://arxiv.org/abs/2006.08052 (Accessed 24 October 2020).
40. D. H. Brookes, H. Park, J. Listgarten, Conditioning by adaptive sampling for robust design. arXiv [Preprint] (2021). https://arxiv.org/abs/1901.10060 (Accessed 12 May 2021).
41. J. Linder, G. Seelig, Fast differentiable DNA and protein sequence optimization for molecular design. arXiv [Preprint] (2020). https://arxiv.org/abs/2005.11275 (Accessed 20 December 2020).
42. K. K. Yang, Z. Wu, F. H. Arnold, Machine-learning-guided directed evolution for protein engineering. *Nat. Methods* **16**, 687–694 (2019).
43. M. Torrisi, G. Pollastri, Q. Le, Deep learning methods in protein structure prediction. *Comput. Struct. Biotechnol. J.* **18**, 1301–1310 (2020).
44. D. Esposito *et al.*, MaveDB: An open-source platform to distribute and interpret data from multiplexed assays of variant effect. *Genome Biol.* **20**, 223 (2019).
45. A. Hawkins-Hooker *et al.*, Generating functional protein variants with variational autoencoders. *PLOS Comput. Biol.* **17**, e1008736 (2021).
46. A. Madani *et al.*, ProGen: Language modeling for protein generation. bioRxiv [Preprint] (2020). https://doi.org/10.1101/2020.03.07.982272 (Accessed 13 March 2020).
47. E. Asgari, M. R. K. Mofrad, Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLoS One* **10**, e0141287 (2015).
48. K. K. Yang, Z. Wu, C. N. Bedbrook, F. H. Arnold, Learned protein embeddings for machine learning. *Bioinformatics* **34**, 2642–2648 (2018).
49. E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, G. M. Church, Unified rational protein engineering with sequence-based deep representation learning. *Nat. Methods* **16**, 1315–1322 (2019).
50. A. Rives *et al.*, Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl. Acad. Sci. U.S.A.* **118**, e2016239118 (2021).
51. S. Biswas *et al.*, Toward machine-guided design of proteins. bioRxiv [Preprint] (2018). https://doi.org/10.1101/337154 (Accessed 2 June 2018).

Gelman et al.
Neural networks to learn protein sequence–function relationships from deep mutational scanning data

52. Y. Saito *et al.*, Machine-learning-guided mutagenesis for directed evolution of fluorescent proteins. *ACS Synth. Biol.* **7**, 2014–2022 (2018).

53. B. J. Wittmann, Y. Yue, F. H. Arnold, Machine learning-assisted directed evolution navigates a combinatorial epistatic fitness landscape with minimal screening burden. *Cell Syst.*, 10.1016/j.cels.2021.07.008 (2021).

54. A. F. Rubin *et al.*, A statistical framework for analyzing deep mutational scanning data. *Genome Biol.* **18**, 150 (2017).

55. S. Kawashima *et al.*, AAindex: Amino acid index database, progress report 2008. *Nucleic Acids Res.* **36**, D202–D205 (2008).

56. Y. Song *et al.*, High-resolution comparative modeling with RosettaCM. *Structure* **21**, 1735–1742 (2013).

57. A. A. Hagberg, D. A. Schult, P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX" in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, J. Millman, Eds. (SciPy, 2008), pp. 11–15.

58. M. Abadi *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems (2015). https://www.tensorflow.org/. Accessed 18 June 2019.

59. D. Thain, T. Tannenbaum, M. Livny, Distributed computing in practice: The Condor experience. *Concurr. Comput. Pract. Exp* **17**, 323–356 (2005).

60. T. A. Hopf *et al.*, The EVcouplings Python framework for coevolutionary sequence analysis. *Bioinformatics* **35**, 1582–1584 (2019).

61. S. R. Eddy, Accelerated profile HMM searches. *PLoS Comput. Biol.* **7**, e1002195 (2011).

62. B. E. Suzek, Y. Wang, H. Huang, P. B. McGarvey, C. H. Wu, UniRef clusters: A comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics* **31**, 926–932 (2015).

63. M. Ancona, E. Ceolini, C. Öztireli, M. Gross, Towards better understanding of gradient-based attribution methods for deep neural networks. arXiv [Preprint] (2018). https://arxiv.org/abs/1711.06104 (Accessed 7 March 2018).

64. K. T. Simons, R. Bonneau, I. Ruczinski, D. Baker, Ab initio protein structure prediction of CASP III targets using ROSETTA. *Proteins* **37**, 171–176 (1999).

65. S. Gelman, S. A. Fahlberg, P. A. Romero, A. Gitter, Neural networks for deep mutational scanning data (2020). GitHub. https://github.com/gitter-lab/nn4dms. Deposited 22 October 2020.

66. S. Gelman, S. A. Fahlberg, P. A. Romero, A. Gitter, Neural networks for deep mutational scanning data (2020). Zenodo. https://doi.org/10.5281/zenodo.4118330. Deposited 22 October 2020.