bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

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

Sam Gelman[1, 2], **Philip A. Romero**[3, *], **and Anthony Gitter**[1, 2, 4, *]

[1]Department of Computer Sciences, University of Wisconsin-Madison
[2]Morgridge Institute for Research
[3]Department of Biochemistry, University of Wisconsin-Madison
[4]Department of Biostatistics and Medical Informatics, University of Wisconsin-Madison
[*]These authors contributed equally to this work

ABSTRACT

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 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. Our software is available from https://github.com/gitter-lab/nn4dms.

Introduction

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 this 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], or making predictions only for single-mutation variants [6]. 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 [7–9]. There is a current need for general supervised learning methods that can leverage large sequence-function datasets to predict specific molecular phenotypes with the high accuracy required for protein design.

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 to state-of-the-art physics-based and unsupervised prediction methods. Across the different architectures tested, we find networks that capture nonlinear interactions and share information across sequence positions display the greatest predictive performance. Finally, we explore what our neural network models have learned about proteins and how they comprehend the sequence-function mapping. We find the convolutional neural networks learn a protein sequence representation that 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.

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

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 [10]. We develop a deep learning framework to learn from large-scale sequence-function data generated by deep mutational scanning. Deep mutational scanning data consists 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 [11]. 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 multi-mutation variants. We train a neural network to map the encoded sequences to their associated functional scores. Once trained, the network generalizes and can predict functional scores for new, unseen protein variants (Fig. 1a).

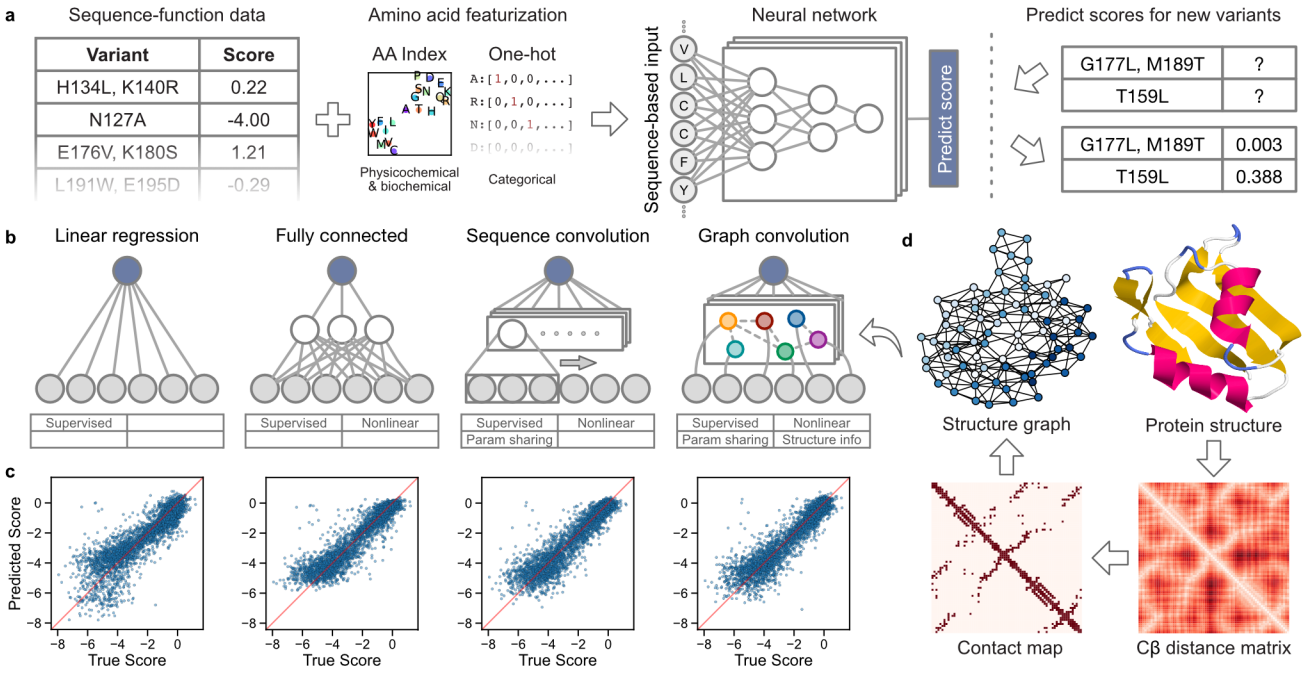

Figure 1. Overview of our supervised learning framework. (a) We use sequence-function data to train a neural network that can predict the functional score of protein variants. The sequence-based input captures physicochemical and biochemical properties of amino acids and supports multiple mutations per 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) Scatter plots 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.

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. 1b). 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 non-additive effects are known as mutational epistasis and are widespread in the protein fitness landscape [12, 13]. 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 beta strands. Applying this filter would enable the network to assess beta strand propensity across the entire input sequence

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

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

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: avGFP, Bgl3, GB1, Pab1, and Ube4b (Table S1). 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 was 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).

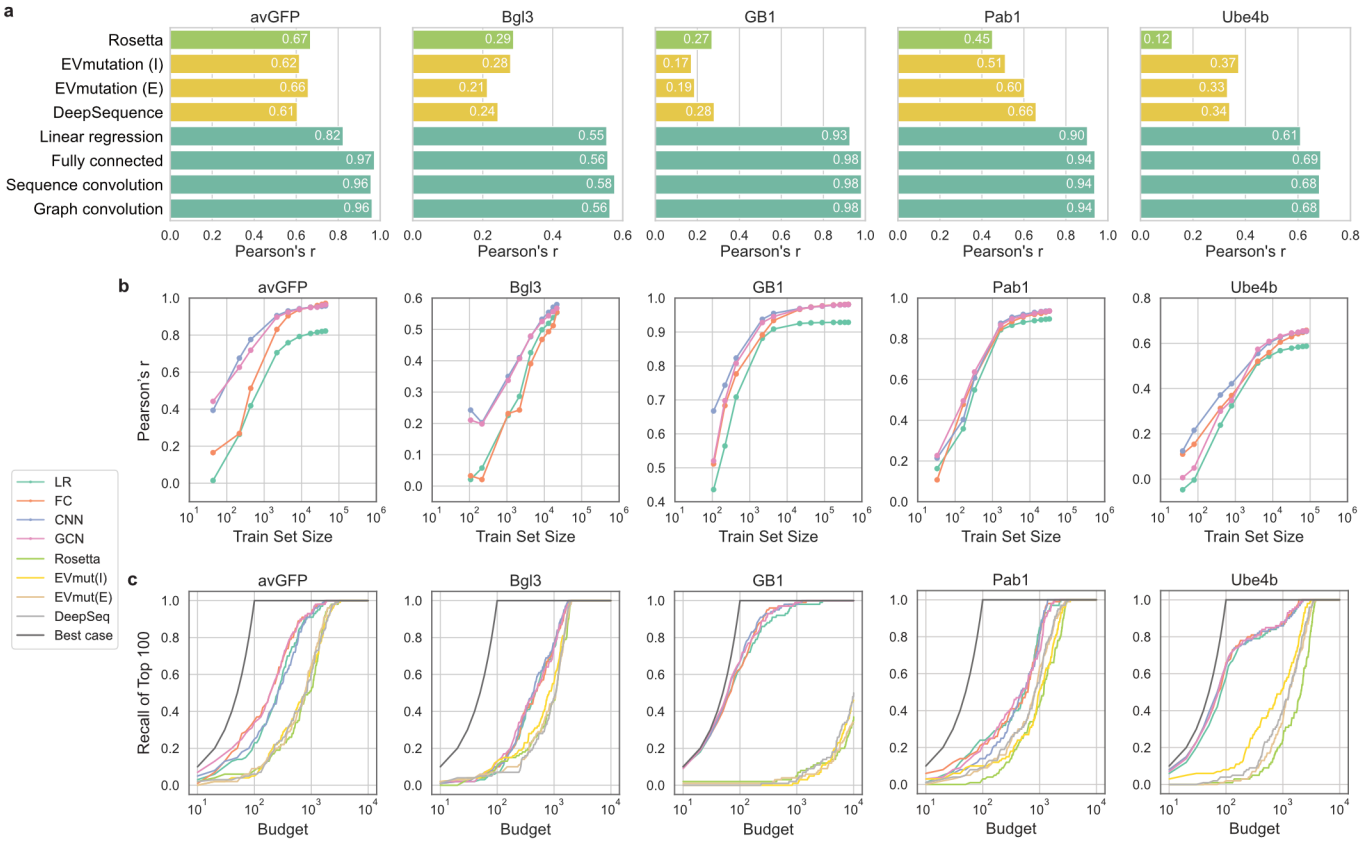

Figure 2. Evaluation of neural networks and comparison to unsupervised methods. (a) Bar plots showing performance, as measured by Pearson's correlation coefficient between true and predicted scores, for Rosetta, EVmutation, DeepSequence, linear regression, and the neural network models. EVmutation (I) 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. The models are grouped by color, where green represents Rosetta, a physics-based method, yellow represents the unsupervised methods, and turquoise represents the supervised methods. (b) Correlation performance of supervised models trained with reduced training set sizes: linear regression (LR), fully connected neural network (FC), sequence convolutional neural network (CNN), and graph convolutional neural network (GCN). (c) The fraction of the true 100 best-scoring variants identified by each model's ranking of variants with the given budget.

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

For comparison, we also evaluated the predictive performance of established physics-based and unsupervised learning methods Rosetta [14], 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. 2a). 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 is 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 (Fig. S1), high epistasis (Fig. S2), and in the case of avGFP, larger numbers of mutations (Fig. S3). The three nonlinear models performed similarly when trained and evaluated on the full training set and testing sets. However, the convolutional networks achieved a better mean squared error when evaluating single-mutation variants in Pab1 and GB1 (Fig. S3). For most proteins, the convolutional networks also had superior performance when trained on smaller training sets (Fig. 2b). The convolutional networks achieve better performance with smaller training set sizes due to their parameter sharing architecture, which allows them to generalize the effects of mutations across different sequence positions and make meaningful predictions for mutations that were not observed in the training data. This may allow convolutional networks to learn from a wider range of sequence-function datasets containing fewer variants.

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 (Fig. S4). We found networks trained using these misspecified baseline graphs had similar accuracy to networks trained with the actual protein structure graph, indicating that protein structure information is contributing little to the model's performance (Fig. S5). We also found the convolutional networks' final fully connected layer was more important than a correctly specified graph structure, and in almost all cases this final fully connected layer helps overcome the misspecified graph structure. Overall, these results suggest 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. 2c). 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 were covered in this predicted set. The supervised models consistently achieve higher recall than Rosetta and the unsupervised methods, although the differences are relatively 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 of a model to prioritize variants that have greater activity than the wild-type protein. We calculated the mean and max scores of the top N predicted test set variants ranked by each model (Figs. S6 and S7). We find the variants prioritized by the supervised models have greater functional scores than 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 wild-type. The max score of the prioritized variants is also important because it represents the best variant suggested by the model. We find nearly all models are able to prioritize variants with a max score greater than 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 max score with a budget as small as 5 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 $\sim$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

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

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 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. 2b). Interestingly, we also found datasets with large library sizes 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.

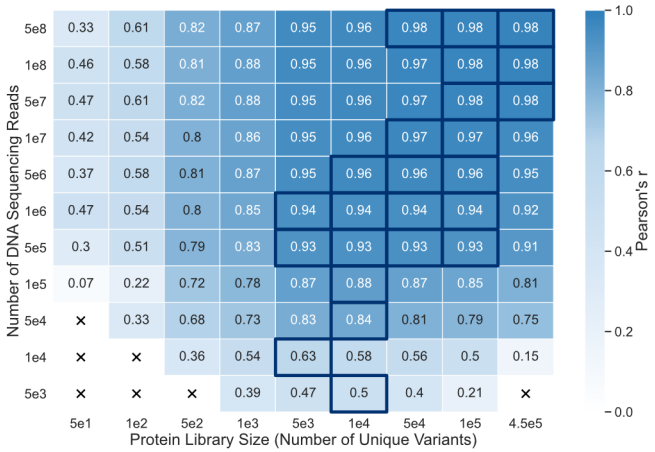

Figure 3. Trade-off between library size and number of sequencing reads. Performance of sequence convolutional models trained on GB1 datasets that have been resampled to simulate different combinations of protein library size and number of sequencing reads in the deep mutational scan. An "X" signifies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

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) [15] 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 terminus, respectively (Fig. S8). Mutations at these residues may disrupt the protein's structural stability and thus decrease the activity measured in the deep mutational scanning experiment [13]. Residue cluster "G3" contains variants with mutations at the IgG-binding interface, and these likely decrease activity by disrupting key binding

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

interactions. This clustering of variants based on different molecular mechanisms suggests the network is learning biologically meaningful aspects of protein function.

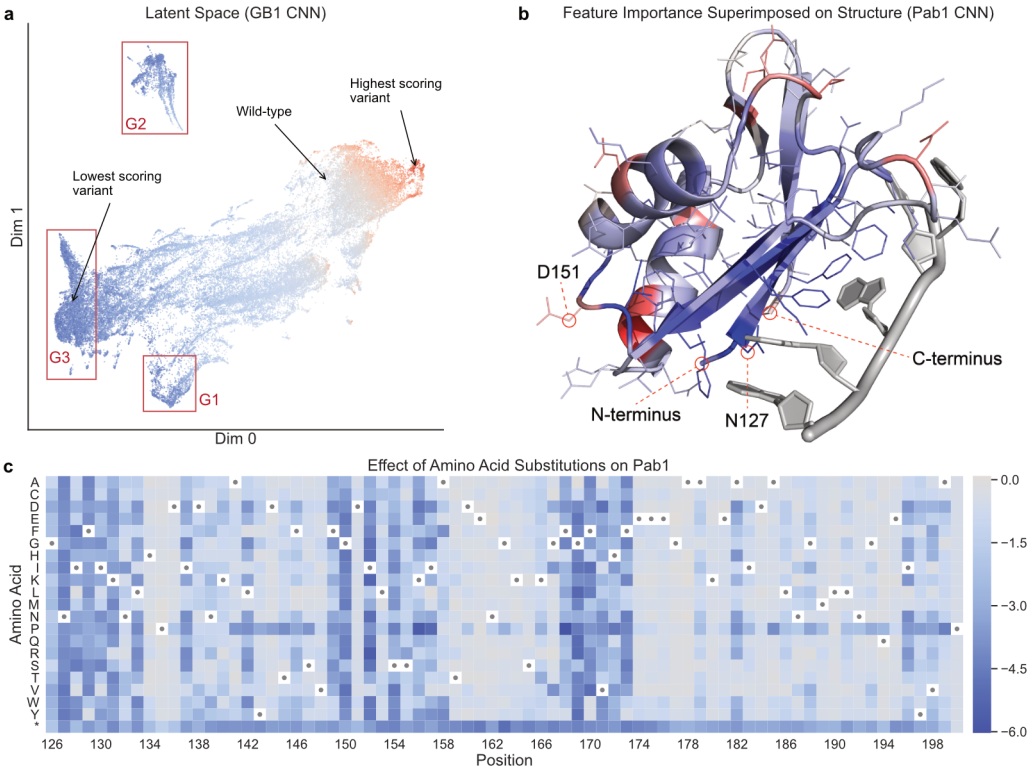

Figure 4. Neural network interpretation. (a) A UMAP projection of the latent space of the GB1 sequence convolutional network, as captured at the last internal layer of the network. In this latent space, similar variants are grouped together based on the transformation applied by the network to predict 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 sequence convolutional network, 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 sequence convolutional network. Wild-type residues are indicated with dots, and * is the stop codon. Most single mutations are predicted to be neutral or deleterious.

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 [16] 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 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 residue N127 cannot be replaced with any other amino acid without significantly decreasing Pab1 binding activity [17]. 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 [17]. 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 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

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

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.

Discussion

We have presented a new 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. The parameter sharing inherent to convolutional networks improves performance by allowing generalization of the effects of mutations across different sequence positions. This capability is especially important when training models with fewer examples because the limited sequence-function information can be applied to unobserved mutations.

Recent studies have examined supervised deep learning methods capable of scaling to deep mutational scanning datasets. One study benchmarked combinations of supervised learning methods and protein sequence encodings [18]. Consistent with our results, it found sequence convolutional neural networks with amino acid property-based features tended to perform better than alternatives. Protein engineering with UniRep [19] 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 co-evolution, and protein sequence features [20]. 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 co-evolution 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 [21–24] and rapid developments in graph neural network architectures [25].

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 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. Another important area of future work will be testing how well our neural network models can extrapolate to sequences outside the training distribution, such as variants with higher-order mutations [26].

Machine learning is revolutionizing our ability to model and predict the complex relationships between protein sequence, structure, and function [27, 28]. 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. Other important machine learning advances relevant to protein engineering include generative modeling to sample non-natural protein sequences [23, 29, 30], language models to learn protein representations from diverse natural sequences [31–34], and strategies to incorporate machine learning predictions into directed evolution experiments [35, 36]. Coupled with optimization methods that allow supervised models to efficiently explore new parts of the sequence space [37–40], these approaches could make possible a new generation of data-driven protein engineering.

Methods

Datasets. We tested our supervised learning approach on five deep mutational scanning datasets: avGFP [41], Bgl3 [42], GB1 [13], Pab1 [17], and Ube4b [43]. 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 [44]. 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. See Table S1 for 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 AAindex encoding. One-hot encoding captures the specific amino acid at each position. It consists of a

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

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 [45]. These properties include simple attributes, such as hydrophobicity and polarity, as well as more complex characteristics, such as average non-bonded 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 concatenate 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 in [21]. 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 Equation 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.

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

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 4Å-10Å and was selected independently for each dataset during the hyperparameter optimization. We measure distances between residues via distances of the beta 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 [46] 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 (Fig. S4). We used NetworkX [47] v2.3 to generate all protein structure and baseline graphs.

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

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 Figure S5. We used the leaky rectified linear unit (LReLU) 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 (Fig. S9). We used Python v3.6 and TensorFlow [48] 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 over-weighted 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 [49]. We also used 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, non-overlapping sets of data. The training set was 81% of the data, the tuning set was 9%, and the testing set was 10%. We performed a hyperparameter grid search for each dataset using all possible combinations of the hyperparameters in Figure S9. 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 Table S2. This is the main set of hyperparameters. For any subsequently trained models, such as those with reduced training set sizes, 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.

For the random baseline in Figures S6 and S7, 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 (Fig. S6) and the max (Fig. S7) of each random ranking's prioritized variants. Finally, we show the 95% confidence interval calculated as $\pm$ 1.96 times the standard deviation.

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.

Comparison to EVmutation and DeepSequence. We generated multiple sequence alignments using the EVcouplings web server [50] according to the protocol described for EVmutation [1]. We used Jackhmmer [51] to generate an initial alignment with five search iterations against UniRef100 [52] and a sequence inclusion threshold of 0.5 bits/residue. If the alignment had $<$ 80% sequence coverage, we increased the threshold in steps of 0.05 bits/residue until coverage was $\geq$ 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 $\geq$ 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 to 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

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

compute the corresponding energy for each variant. We set the residue selector to a neighborhood of 10Å. 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 consists of two sets of variants: an input set (pre-screening) and a selected set (post-screening). 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 (Fig. S10). The library size refers to the number of unique variants being screened in the deep mutational scan. Note 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 Figure 3 heat map, 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 Figure 3 heat map. To determine how many reads should be sampled from the input set versus 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 5 replicates. Out 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 run 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 [15] 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 v0.1.1 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 [16] 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

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

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 [53] v0.3 to compute the integrated gradients attributions with "steps" set to 100.

Code availability. We provide a cleaned version of our code that can be used to retrain the models from this publication or train new models with different network architectures or for different datasets. We also provide pre-trained models that use the latest code and are functionally equivalent to the ones from this publication. The pre-trained 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`. The software is also archived on Zenodo at `https://doi.org/10.5281/zenodo.4118330`. See Table S3 for software dependencies and their versions.

Acknowledgements

We thank Zhiyuan Duan for his assistance running Rosetta.

Funding

This research was supported by National Institutes of Health award 5R35GM119854, National Institutes of Health training grant 5T32HG002760, a Predoctoral Fellowship from the PhRMA Foundation, and the John W. and Jeanne M. Rowe Center for Research in Virology at the Morgridge Institute for Research. In addition, this research benefited from the use of credits from the National Institutes of Health Cloud Credits Model Pilot, a component of the Big Data to Knowledge program, and used resources of the Argonne Leadership Computing Facility, which is a Department of Energy Office of Science User Facility supported under contract DE-AC02-06CH11357. The research was performed using the compute resources and assistance of the University of Wisconsin-Madison Center for High Throughput Computing in the Department of Computer Sciences.

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

Supplement

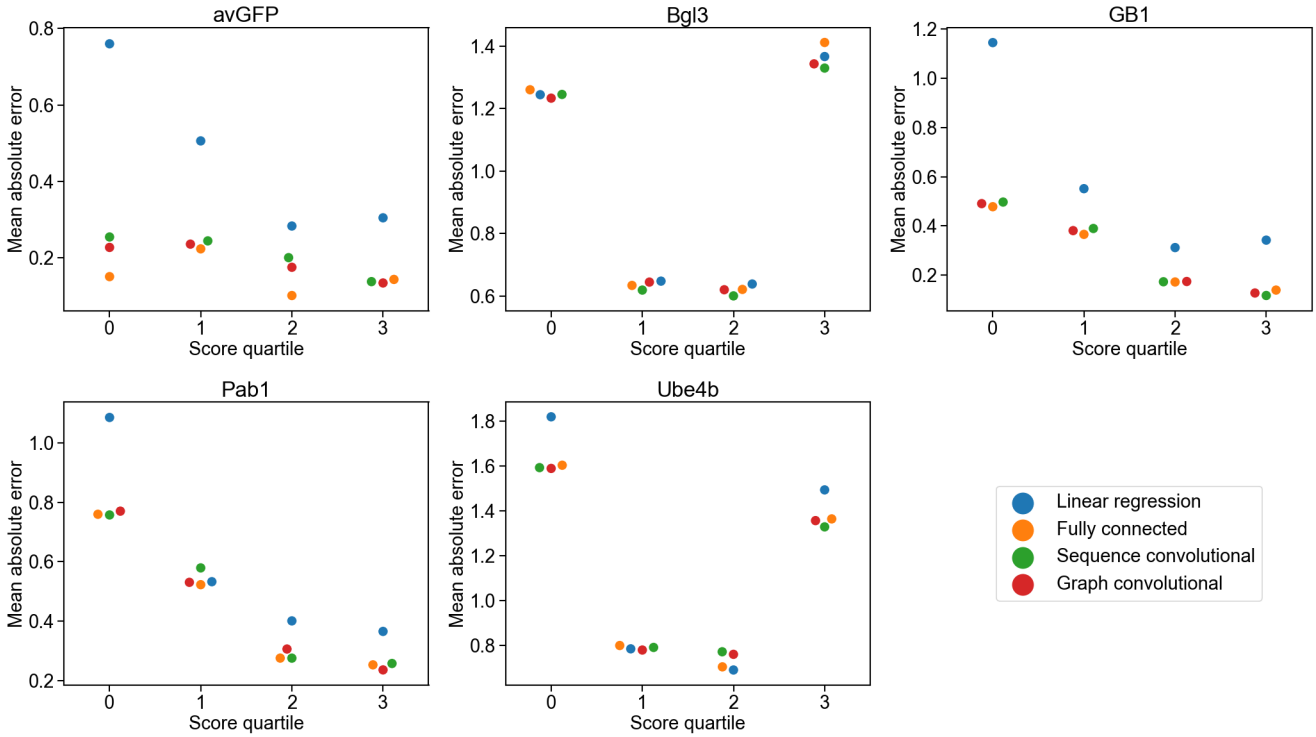

Figure S1. Mean absolute error vs. score quartile. The mean absolute error in the models' predictions grouped by score quartile. Linear regression has a substantial jump in error for low-scoring variants compared to the other models in avGFP, GB1, Pab1, and Ube4b.

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

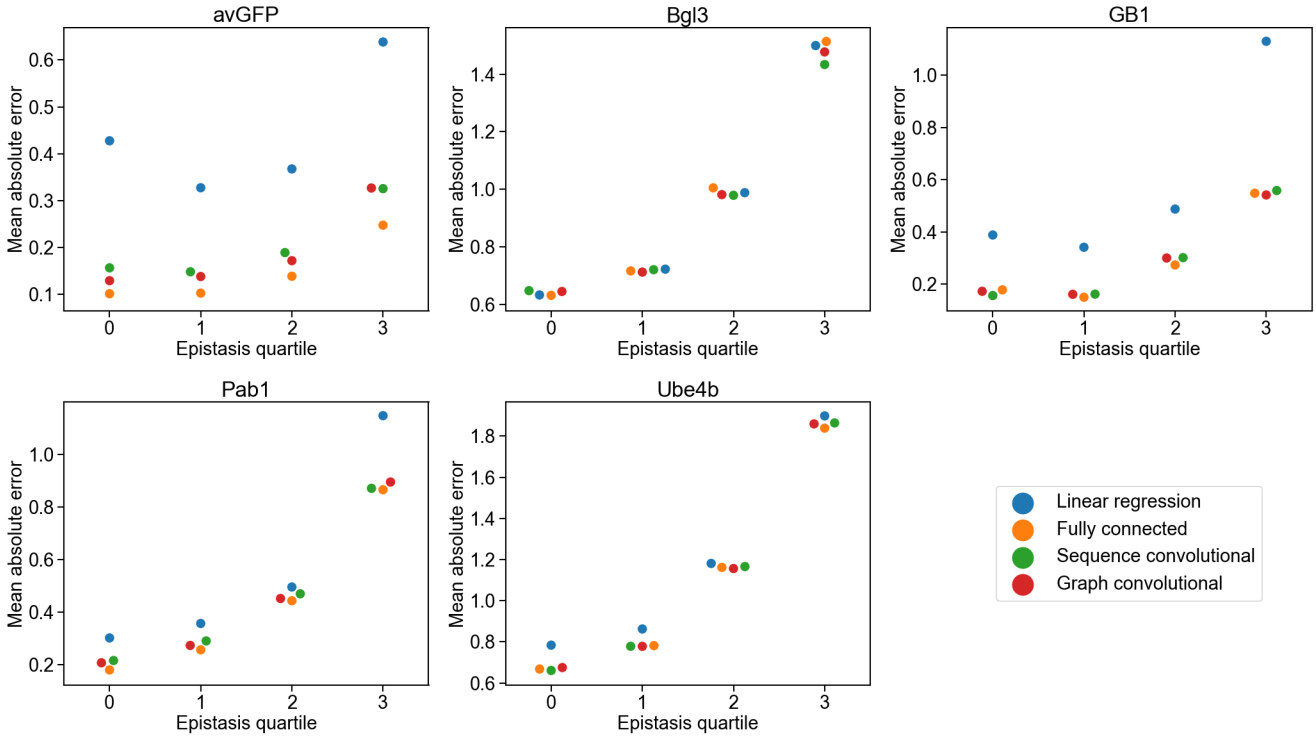

Figure S2. Mean absolute error vs. epistasis quartile. The mean absolute error in the models' predictions grouped by absolute epistasis quartile. We compute epistasis by subtracting the expected score for the multi-mutant sequence from the true score. The expected score for the multi-mutant sequence is the sum of the corresponding single-mutant scores, truncated to the observed minimum or maximum in the dataset. Linear regression has a substantial jump in error for high-epistasis variants compared to the other models in avGFP, GB1, and Pab1.

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

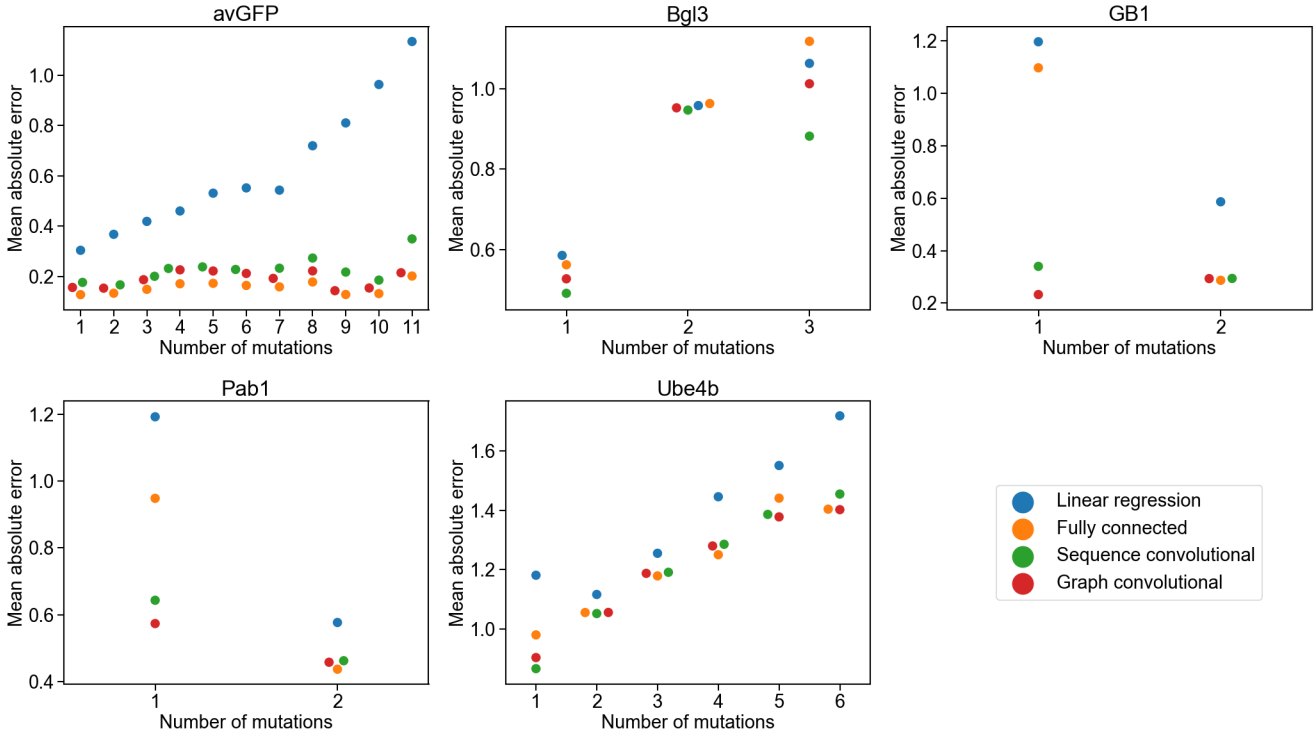

Figure S3. Mean absolute error vs. number of mutations. The absolute error in the models' predictions for each variant grouped by the number of mutations in the variant. Linear regression struggles with increasing numbers of mutations in avGFP. The convolutional networks perform better than linear regression and the fully connected network on single-mutation variants in GB1 and Pab1.

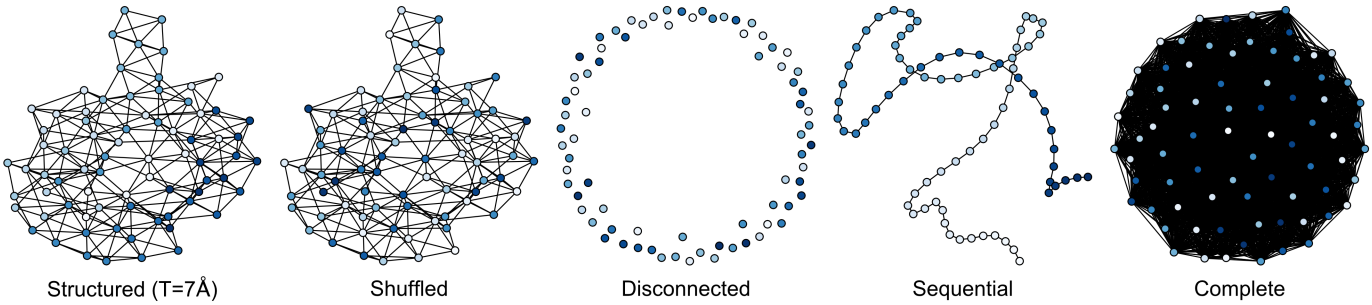

Figure S4. Protein structure graphs for Pab1. The graph convolutional network uses a graph of the protein's structure to determine which residues are close together. In addition to the standard graph based on the protein's actual structure, we tested four baseline graphs: a shuffled graph based on the 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. The graphs pictured are for the Pab1 dataset. The structured graph uses a distance threshold of 7Å to determine which residues should be connected with edges (selected by hyperparameter sweep). The nodes are colored according to each residue's sequence position, with light colors corresponding to residues at the start of the sequence and dark blue colors corresponding to residues at the end of the sequence.

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

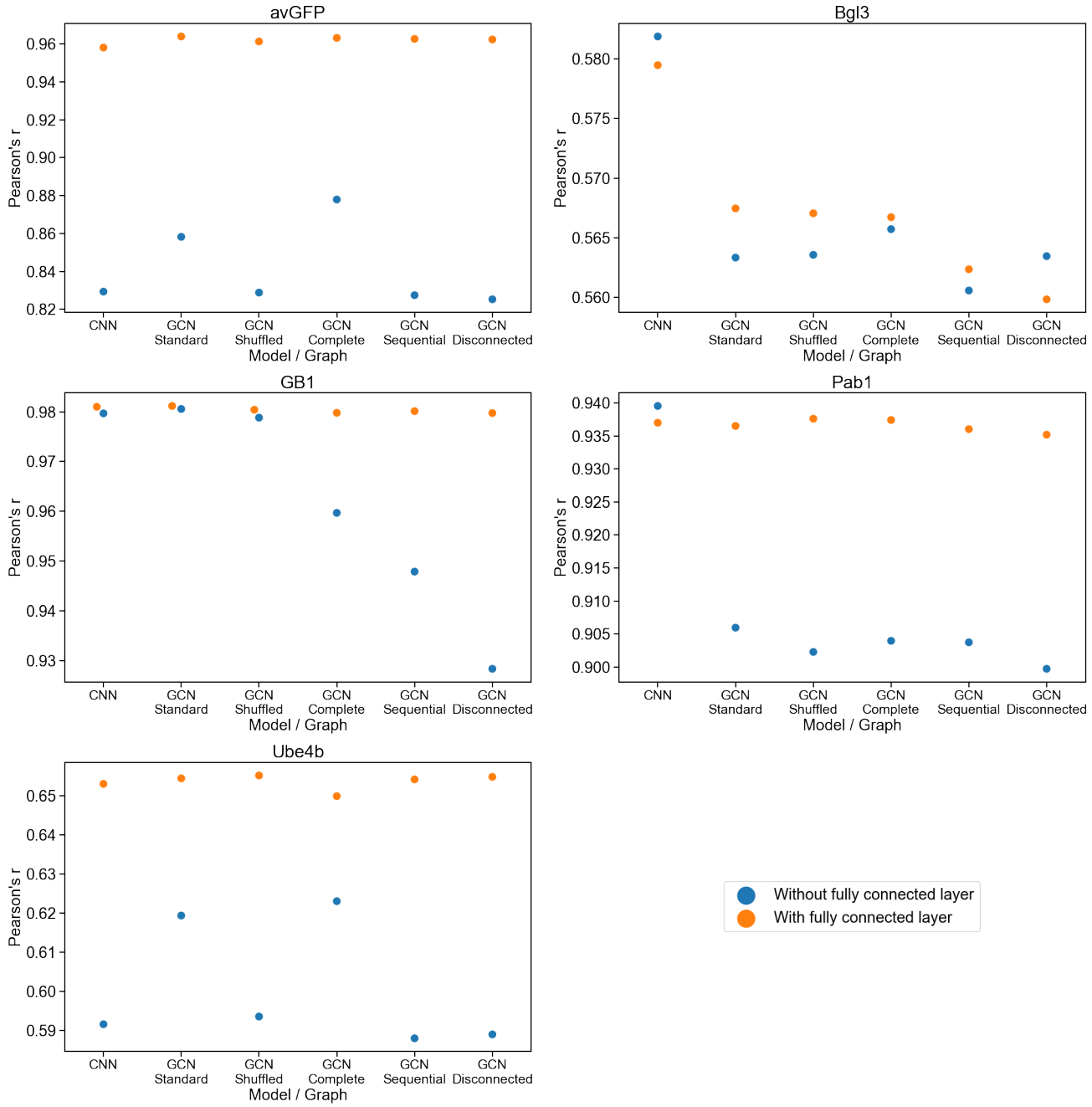

Figure S5. Convolutional networks with and without a fully connected layer. The correlation performance of a sequence convolutional network and graph convolutional networks trained with various baseline structure graphs, with and without a final fully connected layer. The standard graph is based on the protein's actual structure. The shuffled graph is a version of the regular structured graph with shuffled node labels. The complete graph contains all possible edges between residues. The sequential graph only contains edges between sequential residues. The disconnected graph contains no edges. The fully connected layer at the end of the network compensates for apparent differences in performance caused by type of convolutional network or different graph structures.

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

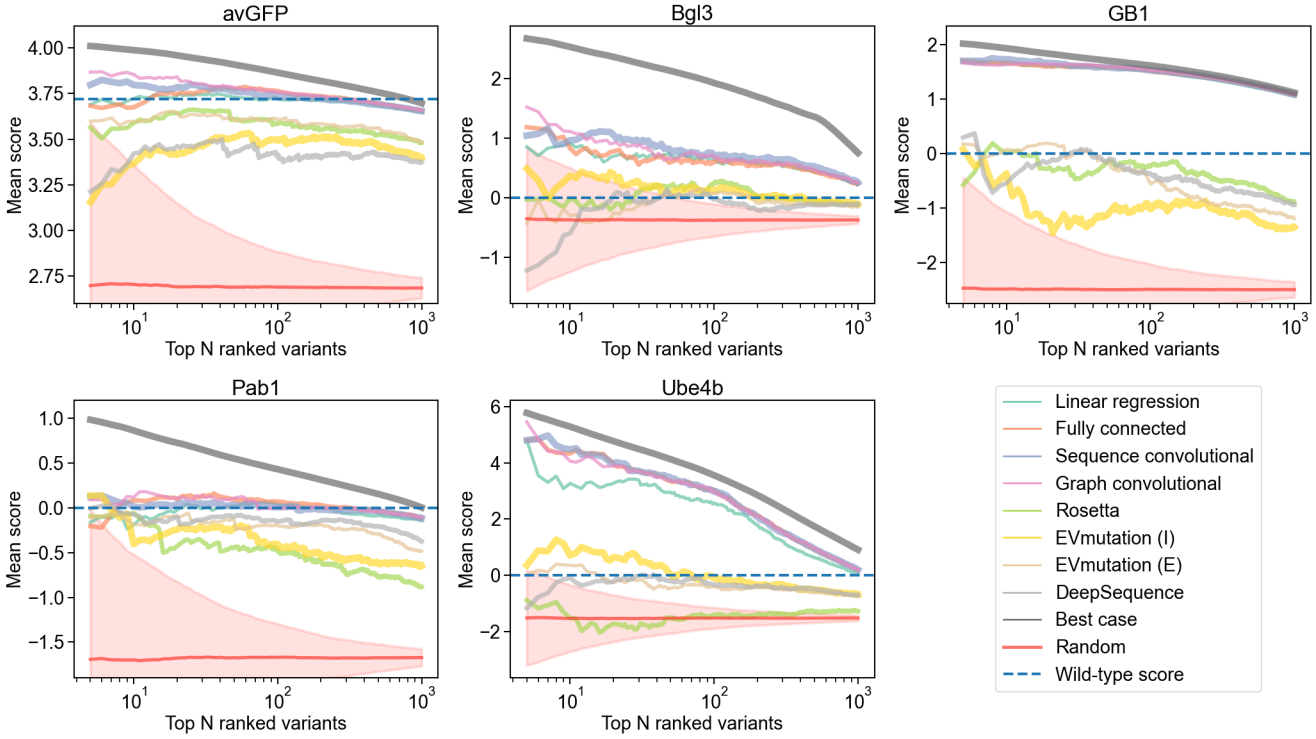

Figure S6. Mean score of highest ranked variants. The mean score of each model's ranking of the highest scoring test set variants. For the most part, the supervised models prioritize variants whose average score is higher than the wild-type. The random baseline is shown with the mean and 95% confidence interval.

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

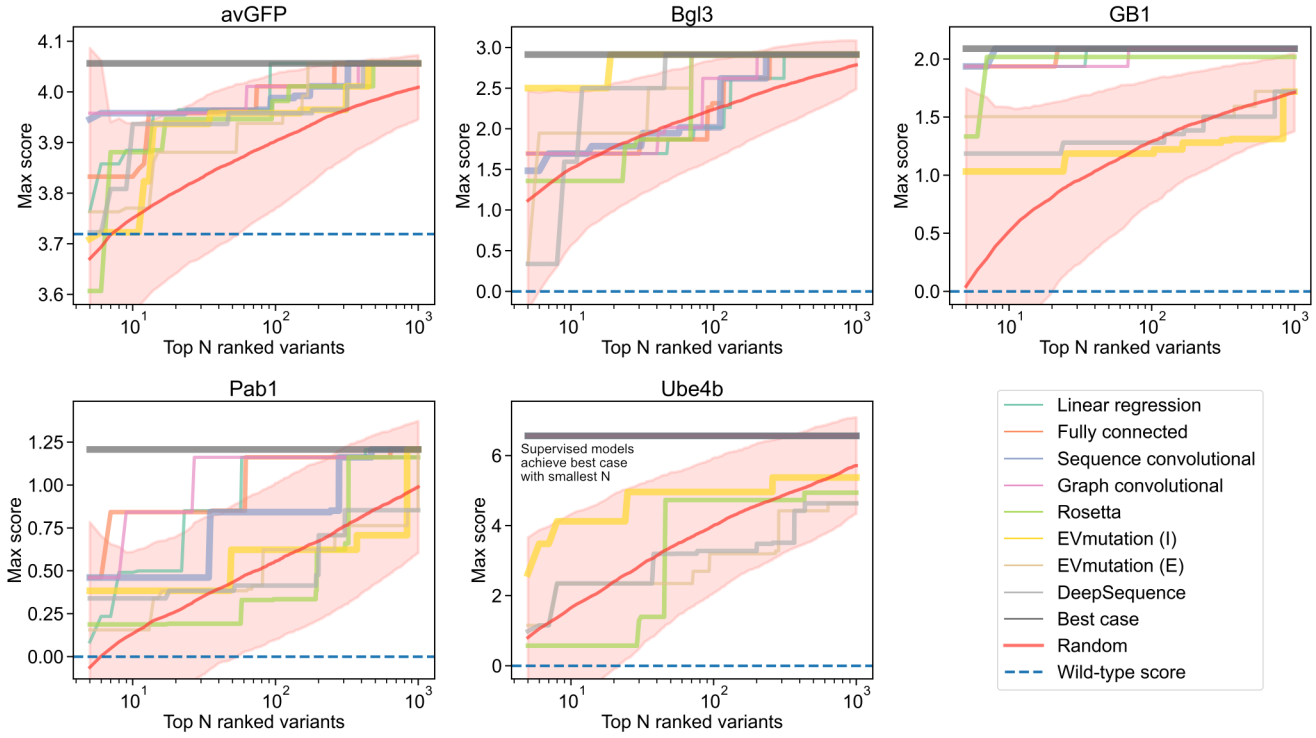

Figure S7. Max score of highest ranked variants. The max score in each model's ranking of the highest scoring test set variants. For Ube4b, the supervised models prioritize a variant with the true max score with the smallest tested budget (N=5), thus all the lines corresponding to the supervised models are hidden behind the line for the true score. Nearly all models across all datasets prioritize variants whose max score is higher than the wild-type. The random baseline is shown with the mean and 95% confidence interval.

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

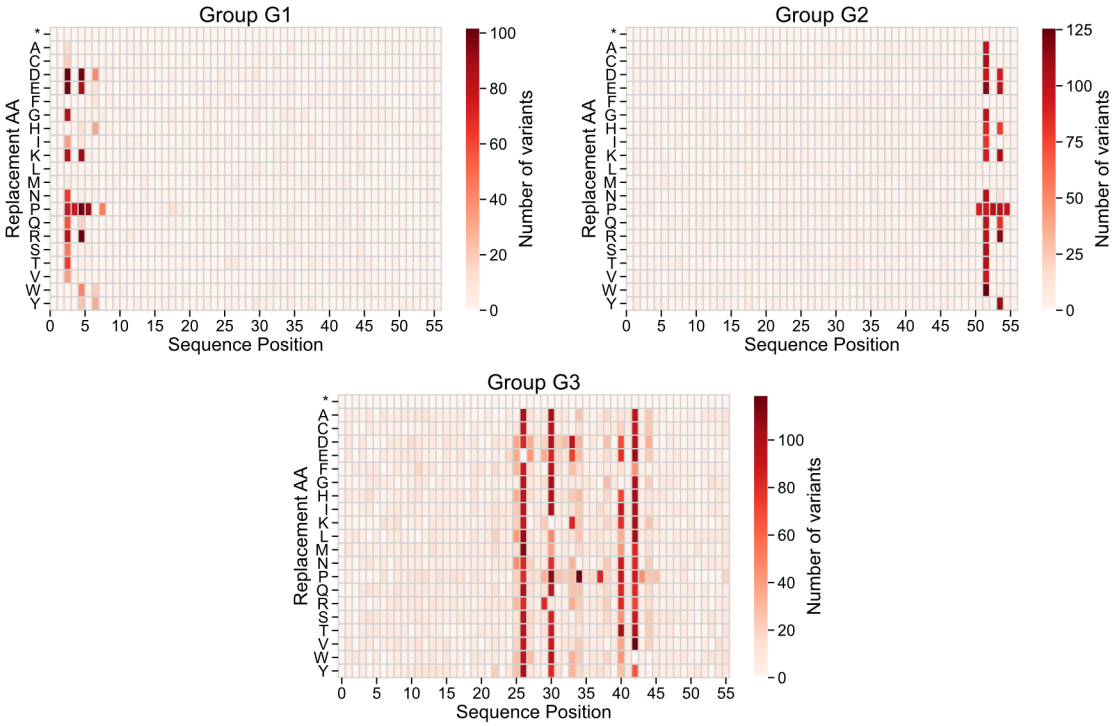

Figure S8. Mutations in GB1 latent space groups. Heat maps showing the number of occurrences of mutations for each annotated group in the GB1 latent space in Figure 4a. Groups G1 and G2 contain variants with mutations at core residues near the start and end of the sequence, respectively. Group G3 contains variants with mutations at surface interface residues.

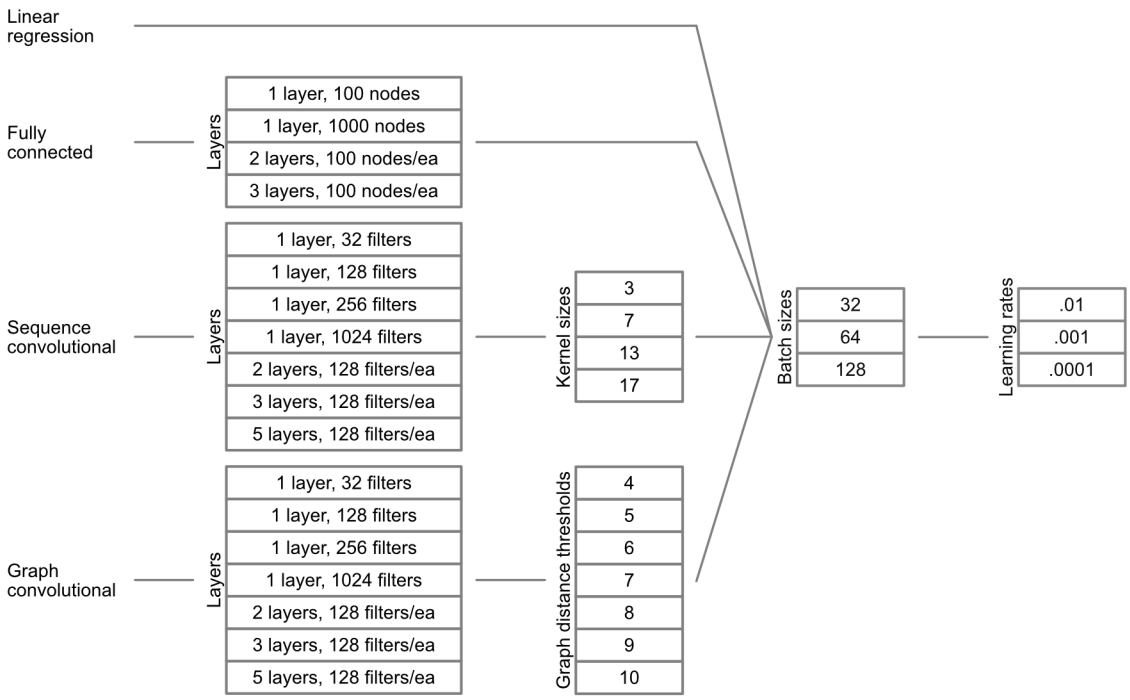

Figure S9. Hyperparameter sweep. We performed an exhaustive hyperparameter sweep for each dataset and type of model using all possible combinations of these hyperparameters.

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

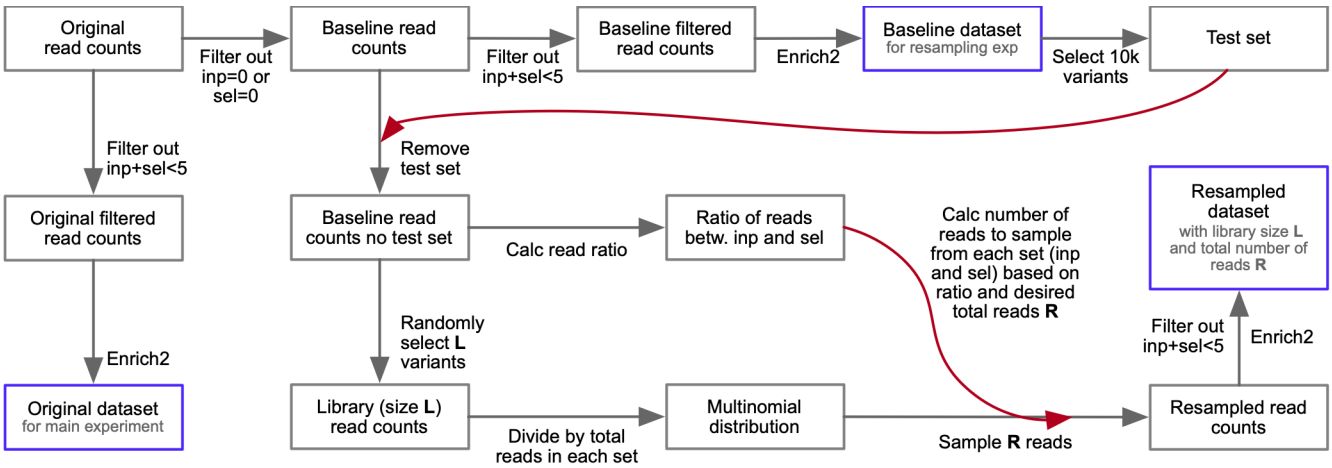

Figure S10. Generation of resampled GB1 datasets. Flowchart showing how we created resampled GB1 datasets corresponding to different library sizes and numbers of reads.

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

Table S1. Datasets. We evaluated our approach on five diverse DMS datasets, representing proteins of varying sizes, folds, and functions.

	Description	Organism	Molecular function	Selection	Year	Ref
avGFP	Green fluorescent protein	Aequorea victoria	Fluorescence	Brightness	2016	[41]
Bgl3	Beta glucosidase	Escherichia coli	Hydrolysis of β-glucosidic linkages	Enzymatic activity	2015	[42]
GB1	IgG-binding domain of protein G	Streptococcus sp	IgG-binding	IgG-FC binding	2014	[13]
Pab1	RRM domain of Pab1	Saccharomyces cerevisiae	poly(A)-binding	mRNA binding	2013	[17]
Ube4b	U-box domain of E4B	Mus musculus	Ubiquitin activating enzyme activity	Ubiquitin ligase activity	2013	[43]

	Length	Variants	Single mutants	Structure from
avGFP	237	54024	1114	Rosetta CM
Bgl3	501	26653	3393	Rosetta CM
GB1	56	536084	1045	PDB ID 2QMT
Pab1	75	40852	1244	Rosetta CM
Ube4b	102	98297	932	Rosetta CM

Table S2. Selected hyperparameters. The hyperparameters selected by a hyperparameter sweep for the main experiment. There are additional parts of the architecture that were not part of the hyperparameter sweep. For example, the fully connected networks have a dropout layer after every dense layer. The convolutional networks a dense layer and a dropout layer before the output node. Experiments with reduced training set sizes and GB1 resampling used the same architectures selected for the main experiment, but they had their own sweeps for learning rate and batch size.

Dataset	Model type	Key parts of architecture	Learning rate	Batch size	Epochs
avGFP	Linear regression	Linear regression	0.0001	128	90
	Fully connected	3 layers, 100 hidden units each	0.0001	32	134
	Sequence convolutional	5 layers, kernel size 3, 128 filters	0.001	64	113
	Graph convolutional	2 layers, 7Å threshold, 128 filters	0.0001	32	130
Bgl3	Linear regression	Linear regression	0.0001	128	164
	Fully connected	2 layers, 100 hidden units each	0.0001	32	187
	Sequence convolutional	1 layer, kernel size 17, 32 filters	0.0001	64	102
	Graph convolutional	1 layer, 6Å threshold, 32 filters	0.0001	128	129
GB1	Linear regression	Linear regression	0.0001	128	27
	Fully connected	1 layer, 1000 hidden units	0.0001	64	110
	Sequence convolutional	3 layers, kernel size 17, 128 filters	0.0001	32	27
	Graph convolutional	5 layers, 7Å threshold, 128 filters	0.0001	32	109
Pab1	Linear regression	Linear regression	0.001	128	47
	Fully connected	3 layers, 100 hidden units each	0.001	128	108
	Sequence convolutional	3 layers, kernel size 17, 128 filters	0.0001	128	42
	Graph convolutional	1 layer, 7Å threshold, 32 filters	0.0001	128	232
Ube4b	Linear regression	Linear regression	0.0001	64	73
	Fully connected	3 layers, 100 hidden units each	0.0001	64	124
	Sequence convolutional	5 layers, kernel size 3, 128 filters	0.0001	128	29
	Graph convolutional	3 layers, 7Å threshold, 128 filters	0.0001	128	92

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

Table S3. Main software packages. The main libraries and version numbers used to train and evaluate models.

Library	Version
python	3.6.8
cudatoolkit	10.0.130
cudnn	7.6.0
tensorflow-gpu	1.14.0
gast	0.2.2
numpy	1.16.4
joblib	0.13.2
matplotlib	3.1.1
networkx	2.3
pandas	0.25.0
scikit-learn	0.21.2
scipy	1.3.0
seaborn	0.9.0
enrich2	1.2.1

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

References

1. Hopf, T. A. *et al.* Mutation effects predicted from sequence co-variation. *Nature Biotechnology* **35,** 128–135. doi:10.1038/nbt.3769 (2017).
2. Riesselman, A. J., Ingraham, J. B. & Marks, D. S. Deep generative models of genetic variation capture the effects of mutations. *Nature Methods* **15,** 816–822. doi:10.1038/s41592-018-0138-4 (2018).
3. Fox, R. J. *et al.* Improving catalytic function by ProSAR-driven enzyme evolution. *Nature Biotechnology* **25,** 338–344. doi:10.1038/nbt1286 (2007).
4. Song, H., Bremer, B. J., Hinds, E. C., Raskutti, G. & Romero, P. A. Inferring protein sequence-function relationships with large-scale positive-unlabeled learning. *bioRxiv.* doi:10.1101/2020.08.19.257642 (2020).
5. Romero, P. A., Krause, A. & Arnold, F. H. Navigating the protein fitness landscape with Gaussian processes. *Proceedings of the National Academy of Sciences* **110,** E193. doi:10.1073/pnas.1215251110 (2013).
6. Gray, V. E., Hause, R. J., Luebeck, J., Shendure, J. & Fowler, D. M. Quantitative Missense Variant Effect Prediction Using Large-Scale Mutagenesis Data. *Cell Systems* **6,** 116–124.e3. doi:10.1016/j.cels.2017.11.003 (2018).
7. Vaser, R., Adusumalli, S., Leng, S. N., Sikic, M. & Ng, P. C. SIFT missense predictions for genomes. *Nature Protocols* **11,** 1–9. doi:10.1038/nprot.2015.123 (2016).
8. Adzhubei, I. A. *et al.* A method and server for predicting damaging missense mutations. *Nature Methods* **7,** 248–249. doi:10.1038/nmeth0410-248 (2010).
9. Hecht, M., Bromberg, Y. & Rost, B. Better prediction of functional effects for sequence variants. *BMC Genomics* **16,** S1. doi:10.1186/1471-2164-16-S8-S1 (2015).
10. Ching, T. *et al.* Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface* **15,** 20170387. doi:10.1098/rsif.2017.0387 (2018).
11. Fowler, D. M. & Fields, S. Deep mutational scanning: a new style of protein science. *Nature Methods* **11,** 801–807. doi:10.1038/nmeth.3027 (2014).
12. Starr, T. N. & Thornton, J. W. Epistasis in protein evolution. *Protein Science* **25,** 1204–1218. doi:10.1002/pro.2897 (2016).
13. Olson, C. A., Wu, N. C. & Sun, R. A Comprehensive Biophysical Description of Pairwise Epistasis throughout an Entire Protein Domain. *Current Biology* **24,** 2643–2651. doi:10.1016/j.cub.2014.09.072 (2014).
14. Alford, R. F. *et al.* The Rosetta All-Atom Energy Function for Macromolecular Modeling and Design. *Journal of Chemical Theory and Computation* **13,** 3031–3048. doi:10.1021/acs.jctc.7b00125 (2017).
15. McInnes, L., Healy, J. & Melville, J. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction* arXiv: 1802.03426 [stat.ML] (2020).
16. Sundararajan, M., Taly, A. & Yan, Q. *Axiomatic Attribution for Deep Networks* arXiv: 1703.01365 [cs.LG] (2017).
17. Melamed, D., Young, D. L., Gamble, C. E., Miller, C. R. & Fields, S. Deep mutational scanning of an RRM domain of the Saccharomyces cerevisiae poly(A)-binding protein. *RNA* **19,** 1537–1551. doi:10.1261/rna.040709.113 (2013).
18. Xu, Y. *et al.* Deep Dive into Machine Learning Models for Protein Engineering. *Journal of Chemical Information and Modeling* **60,** 2773–2790. doi:10.1021/acs.jcim.0c00073 (2020).
19. Biswas, S., Khimulya, G., Alley, E. C., Esvelt, K. M. & Church, G. M. Low-N protein engineering with data-efficient deep learning. *bioRxiv.* doi:10.1101/2020.01.23.917682 (2020).
20. Luo, Y. *et al.* Evolutionary context-integrated deep sequence modeling for protein engineering. *bioRxiv.* doi:10.1101/2020.01.16.908509 (2020).
21. Fout, A., Byrd, J., Shariat, B. & Ben-Hur, A. *Protein interface prediction using graph convolutional networks* in *Advances in Neural Information Processing Systems 30* (eds Guyon, I. *et al.*) 6530–6539 (Curran Associates, Inc., 2017).
22. Gligorijevic, V. *et al.* Structure-Based Protein Function Prediction using Graph Convolutional Networks. *bioRxiv.* doi:10.1101/786236 (2020).
23. Strokach, A., Becerra, D., Corbi-Verge, C., Perez-Riba, A. & Kim, P. M. Fast and Flexible Protein Design Using Deep Graph Neural Networks. *Cell Systems.* doi:10.1016/j.cels.2020.08.016.
24. Sanyal, S., Anishchenko, I., Dagar, A., Baker, D. & Talukdar, P. ProteinGCN: Protein model quality assessment using Graph Convolutional Networks. *bioRxiv.* doi:10.1101/2020.04.06.028266 (2020).

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

25. Wu, Z. *et al.* A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems,* 1–21. doi:10.1109/TNNLS.2020.2978386 (2020).

26. Mater, A. C., Sandhu, M. & Jackson, C. The NK Landscape as a Versatile Benchmark for Machine Learning Driven Protein Engineering. *bioRxiv.* doi:10.1101/2020.09.30.319780 (2020).

27. Yang, K. K., Wu, Z. & Arnold, F. H. Machine-learning-guided directed evolution for protein engineering. *Nature Methods* **16,** 687–694. doi:10.1038/s41592-019-0496-6 (2019).

28. Torrisi, M., Pollastri, G. & Le, Q. Deep learning methods in protein structure prediction. *Computational and Structural Biotechnology Journal* **18,** 1301–1310. doi:10.1016/j.csbj.2019.12.011 (2020).

29. Hawkins-Hooker, A. *et al.* Generating functional protein variants with variational autoencoders. *bioRxiv.* doi:10.1101/2020.04.07.029264 (2020).

30. Madani, A. *et al.* ProGen: Language Modeling for Protein Generation. *bioRxiv.* doi:10.1101/2020.03.07.982272 (2020).

31. Asgari, E. & Mofrad, M. R. K. Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics. *PLOS ONE* **10,** e0141287. doi:10.1371/journal.pone.0141287 (2015).

32. Yang, K. K., Wu, Z., Bedbrook, C. N. & Arnold, F. H. Learned protein embeddings for machine learning. *Bioinformatics* **34,** 2642–2648. doi:10.1093/bioinformatics/bty178 (2018).

33. Alley, E. C., Khimulya, G., Biswas, S., AlQuraishi, M. & Church, G. M. Unified rational protein engineering with sequence-based deep representation learning. *Nature Methods* **16,** 1315–1322. doi:10.1038/s41592-019-0598-1 (2019).

34. Rives, A. *et al.* Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv.* doi:10.1101/622803 (2020).

35. Biswas, S. *et al.* Toward machine-guided design of proteins. *bioRxiv.* doi:10.1101/337154 (2018).

36. Saito, Y. *et al.* Machine-Learning-Guided Mutagenesis for Directed Evolution of Fluorescent Proteins. *ACS Synthetic Biology* **7,** 2014–2022. doi:10.1021/acssynbio.8b00155 (2018).

37. Angermueller, C. *et al.* *Population-Based Black-Box Optimization for Biological Sequence Design* arXiv: 2006.03227 [cs.LG] (2020).

38. Fannjiang, C. & Listgarten, J. *Autofocused oracles for model-based design* arXiv: 2006.08052 [cs.LG] (2020).

39. Brookes, D. H., Park, H. & Listgarten, J. *Conditioning by adaptive sampling for robust design* arXiv: 1901.10060 [cs.LG] (2020).

40. Linder, J. & Seelig, G. *Fast differentiable DNA and protein sequence optimization for molecular design* arXiv: 2005.11275 [cs.LG] (2020).

41. Sarkisyan, K. S. *et al.* Local fitness landscape of the green fluorescent protein. *Nature* **533,** 397–401. doi:10.1038/nature17995 (2016).

42. Romero, P. A., Tran, T. M. & Abate, A. R. Dissecting enzyme function with microfluidic-based deep mutational scanning. *Proceedings of the National Academy of Sciences* **112,** 7159. doi:10.1073/pnas.1422285112 (2015).

43. Starita, L. M. *et al.* Activity-enhancing mutations in an E3 ubiquitin ligase identified by high-throughput mutagenesis. *Proceedings of the National Academy of Sciences* **110,** E1263. doi:10.1073/pnas.1303309110 (2013).

44. Rubin, A. F. *et al.* A statistical framework for analyzing deep mutational scanning data. *Genome Biology* **18,** 150. doi:10.1186/s13059-017-1272-5 (2017).

45. Kawashima, S. *et al.* AAindex: amino acid index database, progress report 2008. *Nucleic Acids Research* **36,** D202–D205. doi:10.1093/nar/gkm998 (suppl_1 2008).

46. Song, Y. *et al.* High-Resolution Comparative Modeling with RosettaCM. *Structure* **21,** 1735–1742. doi:10.1016/j.str.2013.08.005 (2013).

47. Hagberg, A. A., Schult, D. A. & Swart, P. J. *Exploring network structure, dynamics, and function using NetworkX* in *Proceedings of the 7th Python in Science Conference* (eds Varoquaux, G., Vaught, T. & Millman, J.) (Pasadena, CA USA, 2008), 11–15.

48. Abadi, M. *et al.* TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org (2015).

bioRxiv preprint doi: https://doi.org/10.1101/2020.10.25.353946; this version posted October 25, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY 4.0 International license.

49. Thain, D., Tannenbaum, T. & Livny, M. Distributed computing in practice: the Condor experience. *Concurrency and Computation: Practice and Experience* **17,** 323–356. doi:10.1002/cpe.938 (2-4 2005).

50. Hopf, T. A. *et al.* The EVcouplings Python framework for coevolutionary sequence analysis. *Bioinformatics* **35,** 1582–1584. doi:10.1093/bioinformatics/bty862 (2019).

51. Eddy, S. R. Accelerated Profile HMM Searches. *PLOS Computational Biology* **7,** e1002195. doi:10.1371/journal.pcbi.1002195 (2011).

52. Suzek, B. E. *et al.* UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics* **31,** 926–932. doi:10.1093/bioinformatics/btu739 (2015).

53. Ancona, M., Ceolini, E., Öztireli, C. & Gross, M. *Towards better understanding of gradient-based attribution methods for Deep Neural Networks* arXiv: 1711.06104 [cs.LG] (2018).