

University of Groningen

## Super High-Throughput Screening of Enzyme Variants by Spectral Graph Convolutional Neural Networks

Ramírez-Palacios, Carlos; Marrink, Siewert J

*Published in:*  
Journal of Chemical Theory and Computation

*DOI:*  
[10.1021/acs.jctc.2c01227](https://doi.org/10.1021/acs.jctc.2c01227)

**IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.**

*Document Version*  
Publisher's PDF, also known as Version of record

*Publication date:*  
2023

[Link to publication in University of Groningen/UMCG research database](#)

*Citation for published version (APA):*

Ramírez-Palacios, C., & Marrink, S. J. (2023). Super High-Throughput Screening of Enzyme Variants by Spectral Graph Convolutional Neural Networks. *Journal of Chemical Theory and Computation*, 19(14), 4668–4677. Article 1227. Advance online publication. <https://doi.org/10.1021/acs.jctc.2c01227>

### Copyright

Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: <https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment>.

### Take-down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): <http://www.rug.nl/research/portal>. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

# Super High-Throughput Screening of Enzyme Variants by Spectral Graph Convolutional Neural Networks

Carlos Ramírez-Palacios and Siewert J. Marrink\*



Cite This: *J. Chem. Theory Comput.* 2023, 19, 4668–4677



Read Online

ACCESS |



Metrics & More

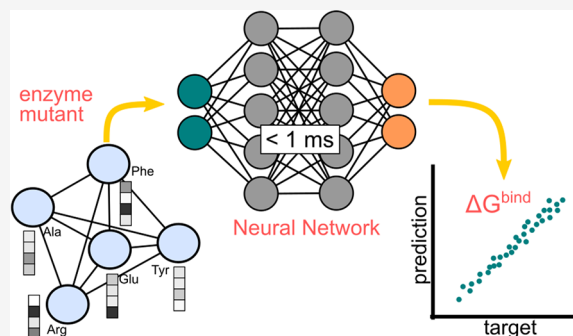


Article Recommendations



Supporting Information

**ABSTRACT:** Finding new enzyme variants with the desired substrate scope requires screening through a large number of potential variants. In a typical *in silico* enzyme engineering workflow, it is possible to scan a few thousands of variants, and gather several candidates for further screening or experimental verification. In this work, we show that a Graph Convolutional Neural Network (GCN) can be trained to predict the binding energy of combinatorial libraries of enzyme complexes using only sequence information. The GCN model uses a stack of message-passing and graph pooling layers to extract information from the protein input graph and yield a prediction. The GCN model is agnostic to the identity of the ligand, which is kept constant within the mutant libraries. Using a minuscule subset of the total combinatorial space ( $20^4$ – $20^8$  mutants) as training data, the proposed GCN model achieves a high accuracy in predicting the binding energy of unseen variants. The network's accuracy was further improved by injecting feature embeddings obtained from a language module pretrained on 10 million protein sequences. Since no structural information is needed to evaluate new variants, the deep learning algorithm is capable of scoring an enzyme variant in under 1 ms, allowing the search of billions of candidates on a single GPU.



## INTRODUCTION

Enzyme engineering is the process of tailoring existing enzymes to enhance some property through alteration of the sequence of amino acids that constitute the enzyme.<sup>1</sup> The property of interest is typically—but not limited to—catalytic activity, substrate specificity, enantioselectivity, or thermostability. The most well-established strategies for tailoring enzymatic properties are directed evolution<sup>2</sup> and rational design.<sup>3</sup> The two strategies require large amounts of experimental or computational effort to find suitable variants.

A third possibility to guide enzyme engineering efforts is through machine learning (ML), and in particular deep learning (DL) approaches. Two of the aspects that have hampered the application of ML in protein engineering are the large data sets needed for training ML models and the challenge of meaningfully representing molecules. The former has been addressed by more efficient architectures (e.g., few-shot or one-shot learning), by designing unsupervised learning tasks, or by using the data generated by classical molecular modeling methodologies (e.g., molecular dynamics trajectories) as training data. The latter aspect has been addressed by representing molecules as graphs,<sup>4</sup> sequences,<sup>5,6</sup> Cartesian coordinates,<sup>7</sup> or vectors obtained via representation learning.<sup>8</sup> ML-assisted protein engineering is thus an emerging field, with recent successes including classification and prediction of the location of the binding site,<sup>9</sup> prediction of the protein 3D structure from the sequence,<sup>10</sup> design of artificial enzymes,<sup>11</sup> de novo protein design,<sup>12</sup> scoring protein–protein docking

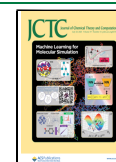
complexes,<sup>13</sup> scoring drug target interactions,<sup>4,14</sup> low-data drug discovery with one-shot learning,<sup>15</sup> predicting the native-like probability of every amino acid in a protein sequence,<sup>16</sup> and protein thermostability.<sup>17</sup>

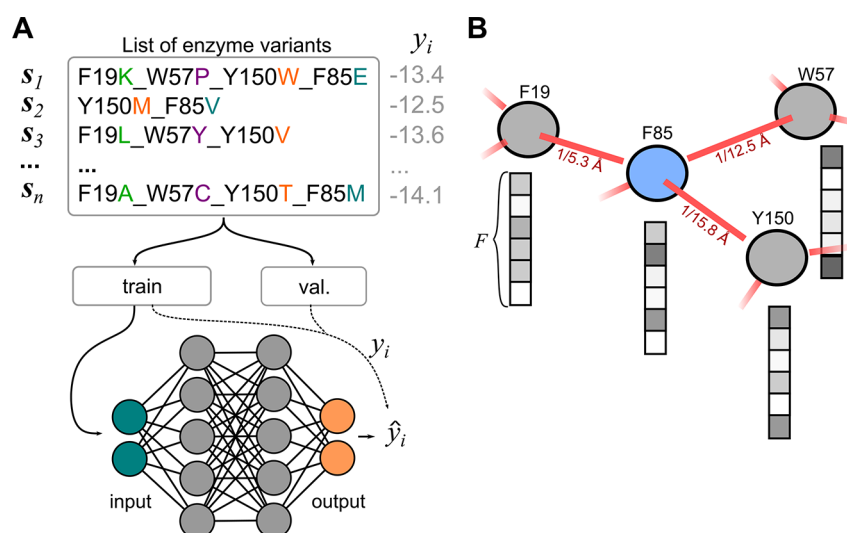
High-throughput screening of enzyme variants can be done experimentally or *in silico*. In wet-lab experiments, it is possible to screen  $>10^6$  variants per hour.<sup>18</sup> The number of variants that can be screened *in silico* depends on the specific methodology used, which determines the accuracy, and on the computational resources available. In either case, screening the entire combinatorial space may not be the most cost-effective strategy to find suitable candidate variants. A good alternative would be to train some algorithm, for example a neural network, that learns to combine mutations and predict the outcome at lightning-fast speed (super high-throughput screening). DL-assisted super high-throughput screening of enzyme variants has been attempted before with varying degree of success.<sup>19</sup> Cadet et al.<sup>20</sup> used a ML approach with protein spectra obtained via Fourier transform of the protein sequence to predict the enantioselectivity of variants of epoxide hydrolase.

**Special Issue:** Machine Learning for Molecular Simulation

**Received:** December 5, 2022

**Published:** March 24, 2023





**Figure 1.** A) Proposed methodology. In the data set, the enzyme variants ( $s_i$ ) are labeled with the binding energy of the variant in complex with the ligand ( $y_i$ ). The data set is used to train a NN, which learns the synergic relationships between mutations and can make predictions for unseen mutants. B) Graph representation of the query variant. Nodes (circles) represent the protein residues connected by edges (red lines). The nodes are characterized by a set of attributes (checked rectangles) representing physicochemical properties of the amino acid.

Their ML approach can evaluate  $2^{30}$  enzyme variants in less than 48 h ( $\sim 0.16$  ms per variant). Wu et al.<sup>21</sup> used shallow neural networks to evaluate combinatorial libraries from human GB1 binding protein (fitness) and a putative nitric oxide dioxygenase from *Rhodothermus marinus* (enantioselectivity). Liao et al.<sup>22</sup> tested the ability of eight ML algorithms to evaluate mutants of proteinase K using a small data set (<100 variants) with experimentally determined enzymatic activities.

The approach presented in this study uses deep Graph Convolutional Neural Networks (GCNs) to explore the fitness landscape of enzyme variants trained on large data sets ( $10^4$  variants) generated by traditional molecular modeling approaches, enabling the evaluation of  $10^8$  variants in less than 24 h.

The enzyme herewith explored is an  $\omega$ -transaminase from *Vibrio fluvialis* (*Vf*-TA) that catalyzes the conversion of ketones into chiral amines with high enantioselectivity.<sup>23</sup> *Vf*-TA is interesting from a protein engineering perspective for industrial production of chiral amines.<sup>24</sup> However, the substrate range of *Vf*-TA, and other  $\omega$ -transaminases, often needs to be tailored to produce the amine of interest.<sup>25,26</sup> Searching for enzyme variants that are better suited to accommodate the molecule of interest is a laborious task that can be aided by computational design approaches. Rational design of new *Vf*-TA variants can be performed *in silico*, in which new variants are generated and scored computationally, and only the best variants are selected for experimental verification. A computationally inexpensive strategy for scoring the goodness of *Vf*-TA variants is by measuring the Rosetta Interface Energy of the enzyme-ligand intermediate complex. The Rosetta Interface Energy (from now on referred to as the binding energy) has been shown to correlate well with the enzymatic activity of  $\omega$ -transaminases.<sup>27,28</sup> Libraries of Rosetta-generated mutants labeled with the binding energy ( $y_i$ ) were used to train NNs to predict the binding energy ( $\hat{y}_i$ ) of unseen variants.

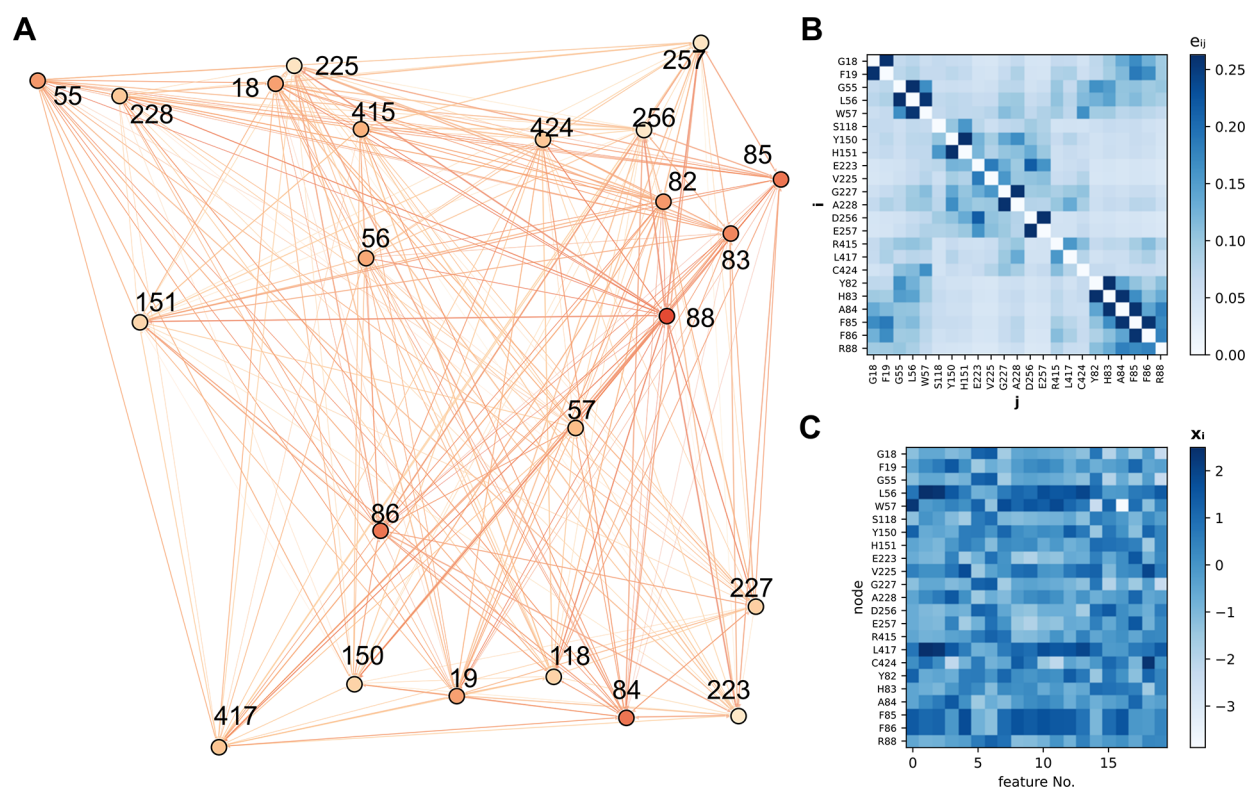
GCNs extend the convolution from regular (e.g., images) to irregular (graphs) representations.<sup>29,30</sup> Similar to classical Convolutional Neural Networks (CNNs), the pooling operation allows GCNs to learn increasingly abstract

representations and to discard unnecessary information. However, the non-Euclidean nature of graphs makes the convolution operation difficult to define and the application of GCNs has lagged behind respect to CNNs. GCNs have been used before to process graphs representing proteins to score protein–protein<sup>13,31</sup> or protein–ligand<sup>4</sup> complexes. The two main approaches to handling convolutions with irregular shapes are spatial and spectral convolution. The former attempts to perform convolution directly in the vertex domain typically by allowing information to flow between neighboring nodes through architecture-specific propagation rules. The latter attempts to perform convolutions in the frequency domain (Fourier-like basis), centering its attention in the graph Laplacian matrix,  $L = D - A$ , where  $D$  is the diagonal degree matrix,  $D_{ii} = \sum_{j=1}^n A_{ij}$ , and  $A$  is the adjacency matrix.

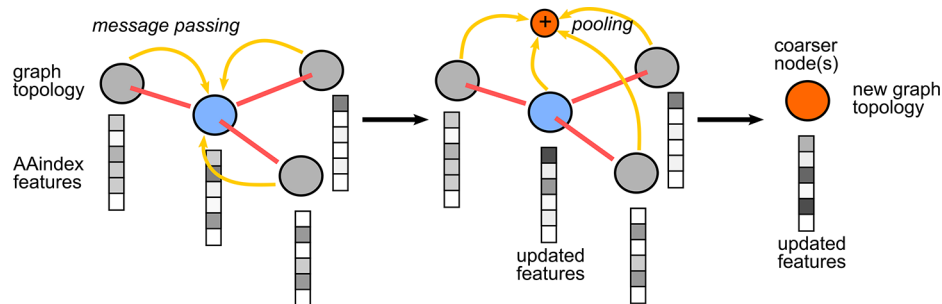
In this work, the spectral GCN proposed by Bianchi et al.<sup>32</sup> is used to analyze the *Vf*-TA variants. The architecture was chosen because the pooling operation clusters the nodes not just according to graph topology but also to node features, which is useful if the graph topology does not change across mutants but the node features do.

## METHODS

**Training Data Sets.** A randomly sampled library containing 10,000 *Vf*-TA variants was created to be used as training data. The variants were created by randomly mutating a set of predetermined  $N_{hot}$  hotspots, to generate  $L_{th}$ -order mutants ( $L = 1$ , single mutant;  $L = 2$ , double mutant; etc.), into one amino acid sampled from the set of 20 naturally occurring amino acids,  $AA = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$ . Each example,  $s_i$ , was labeled ( $y_i$ ) by calculating the binding energy between the enzyme variant and the ligand (Figure 1A). The ligand (Figure S1) was maintained constant within the data set of 10,000 variants. The full data set was split into training (80%), validation (10%), and test (10%) subsets. The standard data set referred to in the paper is  $\mathcal{D}_1$ . Additional data sets were generated by changing the number of hotspots allowed to mutate  $N_{hot} = \{4, 6, 8\}$ , the maximum  $L_{th}$ -order of the generated mutants, the number of



**Figure 2.** A) Example of a graph representation of the binding site of Vf-TA. B) Heatmap showing the edge weight matrix, where the weight of each edge ( $e_{ij}$ ) is the inverse pairwise distance between nodes  $i$  and  $j$ . The edge features were maintained constant across mutants. C) Heatmap showing the feature matrix, where each node (rows) is described by features (along the columns) which are the physicochemical properties of the constituent amino acid.



**Figure 3.** Spectral graph convolution as implemented by the employed layers. The message-passing operation consists of neighboring nodes passing information to each other, which also depends on edges. The pooling operation consists of clustering a subgroup of nodes together according to graph topology and node features. The purpose of these operations is to reduce complexity and yield a more abstract representation. The coarser node(s) (orange) contains information from all parent nodes (gray and blue).

amino acids allowed as target mutations  $|AAI| = \{10, 20\}$ , or the identity of the ligand  $LIG = \{E1, E2, E3, E4, E5\}$ .

**Data Labeling.** The enzyme variants were labeled using the binding energy of the query variant in complex with a ligand. The binding energy is the Rosetta Interface Energy and, unless otherwise indicated, the ligand is E4 (Figure S1). The procedure for performing molecular docking and measuring binding energies is the same as described in Ramírez-Palacios et al.<sup>33</sup> Very briefly, the external aldimine form of the query compound is docked into the binding site of Vf-TA and the binding energy is calculated and averaged over 10 replicas. The binding energies were calculated using the Rosetta suite (build number 57927).<sup>34</sup> The binding energies were whitened to have zero mean and standard deviation of one,  $y = (x - \text{mean})/\text{stdev}$ .

**Graph Representation of the Protein.** The enzyme variants were represented as graphs formed by protein residues (nodes) with pairwise connections (edges). Let  $\mathcal{G} = (\mathcal{X}, \epsilon)$  be a graph, where  $\mathcal{X}$  is the set of nodes with  $F$ -dimensional real attributes  $\mathcal{X} = \{x_i \in \mathbb{R}^F |_{i=1, \dots, N}\}$ , and  $\epsilon$  is the set of edges with  $S$ -dimensional real attributes  $\epsilon = \{e_{ij} \in \mathbb{R}^S |_{x_i, x_j \in \mathcal{X}}\}$  connecting nodes  $i$  and  $j$ . For computation, the graph  $\mathcal{G} = (\mathbf{A}, \mathbf{X}, \mathbf{E})$  is represented by a binary adjacency matrix  $\mathbf{A} \in \{0, 1\}^{N \times N}$ , node features  $\mathbf{X} \in \mathbb{R}^{N \times F}$ , and edge features  $\mathbf{E} \in \mathbb{R}^{N \times N \times S}$ , where  $N$  is the number of nodes (i.e., the number of protein residues).<sup>35</sup>

The graph representation of Vf-TA included only nodes representing residues near the binding site ( $N = 23$ ) (Figure 2A), and all other residues were truncated away. Including only

a small subset of protein residues to form the graph helps ease computations. A smaller subset of those 23 residues was allowed to mutate,  $N_{hot} \in N|_{N_{hot} = \{4,6,8\}}$ . Featurization of the node matrix,  $X$ , was carried out using  $F$  features taken from the AAindex<sup>36</sup> (Figure 2C and S4). The AAindex is a collection of 529 biophysical and biochemical properties of amino acids,  $F \in \mathbb{R}|_{1 \leq F \leq 529}$ . All nodes in the graph were connected to all other nodes. Finally, the edge attributes,  $E$ , were defined as the inverse of the pairwise distance between the protein residues  $C_\alpha$  atoms,  $e_{ij} = 1/\|e_i - e_j\|_2$ . Since the position of the protein  $C_\alpha$  atoms does not change significantly upon mutagenesis, the edge tensor,  $E$ , was set to be invariant to the identity of the mutant (Figure S2). The main advantage of keeping the edges constant across mutants is that evaluation of new variants can be done from the protein sequence alone, which massively reduces the computational cost.

**Graph Convolutions.** The input graph,  $\mathcal{G}$ , goes through a stack of convolutional layers to generate an increasingly more abstract representation of the input signal. The graph convolution layers used in this work, proposed by Bianchi et al.,<sup>32</sup> were obtained from the *Spektral* python library, and consist of a message-passing followed by a graph pooling layer.

The message passing operation is performed using the GCSCConv layer. In this operation, the graph topology remains intact while the features of each node are updated by rules learned through training (Figure 3). The output of the message-passing layer is

$$\bar{X} = \text{ReLU}(\tilde{A}X\Theta_m + X\Theta_s) \quad (1)$$

Where  $\Theta_m$  and  $\Theta_s$  are the trainable weights of the mixing and skip component, respectively.  $\tilde{A}$  is the symmetrically normalized adjacency matrix,  $\tilde{A} = D^{-1/2}AD^{-1/2}$ .

The pooling operation is performed using the MinCutPool layer on  $\bar{X}$ , which uses a skip connection to  $\tilde{A}$  and passes the input through an MLP, to cluster the graph and yield the cluster assignment matrix  $S$ . Graph clustering helps discard superfluous information, provides translation invariance, and reduces model complexity. The coarsened graph,  $(A^{pool}, X^{pool})$ , is thus a more abstract representation of the protein. The pooling operation clusters the nodes according to both graph topology and node features (nodes with similar features are more likely to be clustered together).

The message-passing and clustering operations can be performed several times on the newly generated graphs. At the end, the graph is linearized and passed through a single FC layer with one output unit.

$$\hat{y}_i = \text{ReLU}(WX^{pool} + b) \quad (2)$$

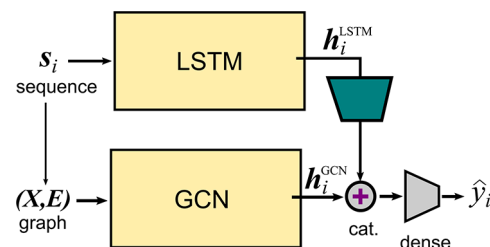
Predictions can then be evaluated using the Mean Squared Error (MSE).

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (3)$$

**Model Build and Training.** The graph convolutional network models were run using the *Spektral*<sup>35</sup> and *TensorFlow*<sup>37</sup> 2.4 libraries. The model is composed of an input layer, a GCSCConv layer, a maxpooling layer, a MLP layer, and an output layer.<sup>32</sup> The models were trained for 200 epochs using the ADAM optimizer with a learning rate of  $1 \times 10^{-4}$  and parameters  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ , and  $\hat{\epsilon} = 1 \times 10^{-7}$ .<sup>38</sup> The input

data was fed to the network in batches of size 1 for training. To avoid overfitting, early stopping with patience of 40 was used.

**Representation Learning Using a Pretrained LSTM Model.** To further improve the accuracy of the trained GCN models, protein sequence embeddings were fed to the NN (Figure 4). The sequence of the query variant,  $s_i \in \mathbb{R}^{452}$ , is



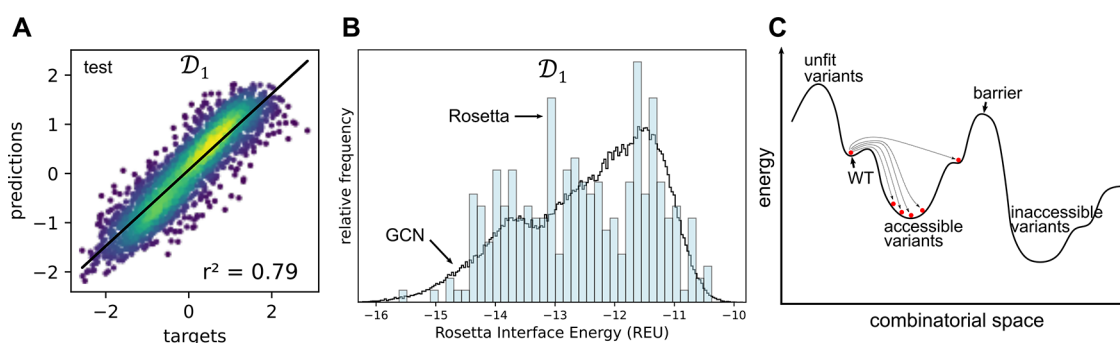
**Figure 4.** Representation learning approach. The query sequence is mapped into a latent space representation ( $s_i \rightarrow h_i^{LSTM}$ ) by a pretrained LSTM network. The same protein sequence can be represented as a graph ( $G_i = (X, E)$ ) and passed through a pretrained GCN network ( $G_i \rightarrow h_i^{GCN}$ ). The two latent space representations are then concatenated (cat.) and passed through a shallow network to yield a prediction. The trapezoid represents the dimensionality reduction via either an FC or a CNN module.

converted to a latent space representation,  $h_i^{LSTM} \in \mathbb{R}^{452 \times 512}$ , by a pretrained bidirectional LSTM model. The LSTM model was trained with  $\sim 10$  million protein sequences from the Pfam database (<https://github.com/flatironinstitute/DeepFRI>). The latent space representation is then reshaped to a low- $d$  vector by passing it through a 2D-CNN module that treats the vector embedding as an image. The hidden representation passes through two convolutional and two pooling layers to yield an 8- $d$  vector. The 8- $d$  vector is then concatenated to the output of the penultimate layer of the already-trained GCN module. The resulting 16- $d$  vector is finally passed through a single dense layer to produce the final 1- $d$  output vector,  $\hat{y}_i \in \mathbb{R}$ .

## RESULTS

### The Trained Spectral Graph Convolutional Network Can Predict Binding Energies with High Accuracy.

A neural network was trained on a mutant library of V<sub>f</sub>-TA enzyme-ligand complexes generated and scored by Rosetta. The data set is a collection of variants, in which a fixed number of positions ( $N_{hot} \in \{4,6,8\}$ ) were randomly mutagenized. The targets ( $y_i$ ) correspond to Rosetta binding energies, but any other type of binding energy would work. The network is unaware of the identity of the ligand used for training, and predictions ( $\hat{y}_i$ ) are based solely on the protein sequence ( $s_i$ ). The protein sequence is converted to a graph representation of the binding site,  $\mathcal{G} = (A, X, E)$ , to serve as input for the GCN. The neural network thus predicts the binding energies of unseen enzyme variants using the graph representation of the protein (Figure 2A). The trained model was not only able to predict with high accuracy the binding energies of unseen variants (Figure 5A and S3), but also does it 6 orders of magnitude faster than Rosetta. Evaluation takes approximately 1.36 ms per screened variant on a single GPU, but it can be parallelized with more workers within the same GPU. This massive speed-up opens the possibility of scanning billions of enzyme variants at a low computational cost. For example, the search space of all eighth-order variants (octuple mutants) is  $20^8 = 2.56 \times 10^{10}$ , which is virtually inaccessible by traditional



**Figure 5.** A) KDE scatter plot showing the correlation between the GCN-predicted binding energies ( $y$ -axis;  $\hat{y}_i$ ) and the Rosetta scores ( $x$ -axis;  $y_i$ ) with the test data set ( $n = 2,000$ ). B) Histogram showing the distribution of binding energies from the entire combinatorial space (black line) overlapped to the histogram obtained with designs proposed by Rosetta (not used for training) (blue bars) ( $n = 800$ ). C) When Rosetta is tasked with proposing new variants from a set of predefined mutable positions, the generated variants will tend to be nearby in combinatorial and conformational space. The variants easily accessible are located close to the wild-type enzyme. The variants that are farther away are less likely to be generated, especially if a barrier needs to be overcome.

methods. The proposed methodology enables screening of the entire search space (Figure 5B).

The GCN model trained on  $\mathcal{D}_1$  was used to screen the whole combinatorial space of  $\mathcal{D}_1$ , i.e., 160,000 mutants (Figure 5B black line). The distribution of the entire combinatorial space of  $\mathcal{D}_1$  was then compared to the distribution of mutants obtained when running Rosetta to propose new mutants ( $n = 2,000$ ). In the run, Rosetta is tasked with proposing mutants (Figure 5B blue bars) where positions 19, 57, 150, and 85 are mutated, without limitations or constraints as to the identity of the amino acids to which these four positions are allowed to mutate. The two distributions are similar, which suggests that the GCN model is able by brute force to propose as good mutants as Rosetta can but at a lower computational cost. Moreover, the brute force approach can explore variants that are not near the initial variant (wild-type) in combinatorial or conformational space (Figure 5C). The screening of the entire combinatorial space of  $\mathcal{D}_1$  by the trained GCN predicted the following five mutants to be the best: F85A\_W57F\_Y150F ( $-15.14$ ), F85P\_W57F\_F19Y ( $-15.15$ ), F85V\_W57F\_Y150F ( $-15.18$ ), F85G\_Y150W\_F19W ( $-15.19$ ), and F85P\_W57F\_Y150F\_F19Y ( $-15.29$ ). The number in parentheses is the energy predicted by the GCN model. Rosetta later scored these mutants with energies of  $-15.28$ ,  $-15.04$ ,  $-15.12$ ,  $-14.35$ , and  $-15.52$  kcal/mol, respectively. Except for the mutant with a score of  $-14.35$ , the other four mutants seem promising candidates. For reference, the training data set contained mutants in the range of  $-8.84$  to  $-15.38$ . How far the trained model can extrapolate to energies outside the range of the training data set is addressed later in the text and Figure S7.

The main ligand used in this study is the external aldimine intermediate of compound 4 (Figure S1). In addition to E4, other ligands were used for training and testing: E1, E2, E3, and E5. The ligands E1–E5 were chosen because it has been experimentally shown that the Rosetta score is a good predictor of the enzyme's activity toward these five compounds.<sup>27</sup> Overall, the trained models are accurate in making binding energy predictions with all five ligands (Figure S6).

The proposed architecture is ideal for the task because the MinCutPool layer assigns clusters based on both graph topology (edge attributes) and node features.<sup>32</sup> Since the majority of the residues in the protein are never mutated, the

graph topology is constant across variants. Because of that, clustering methods that heavily rely on graph topology would not perform ideally. For reference, a model using the classical architecture of Kipf and Welling<sup>39</sup> (Spektral layer: GCNConv) was trained and the results show that while the model is capable of scoring new protein variants, its performance is inferior to GCSCConv (Table 1). In general, the power of the graph representations is clear from the poor performance of a FC dense network, in which no learning occurred.

**Table 1. Results of the Trained Models on the Validation and Test Data Sets for Various Architectures<sup>a</sup>**

Model	Validation			Test		
	↑Acc.	↓Loss	↑ $r^2$	↑Acc.	↓Loss	↑ $r^2$
GCSCConv	0.44	0.23	0.76	0.46	0.23	0.77
GCSCConv (no edges) <sup>b</sup>	0.60	0.56	0.44	0.60	0.54	0.47
GCSCConv (extra nodes) <sup>c</sup>	0.75	0.72	0.31	0.75	0.66	0.34
GCNConv	0.59	0.46	0.57	0.59	0.45	0.55
Dense	0.00	1.00	-1.0	0.00	1.00	-1.00

<sup>a</sup>Data Set is  $\mathcal{D}_1$ . <sup>b</sup>All edges were simply set to 1. <sup>c</sup>Number of nodes was increased from 23 to 291. Evaluation time increases to 14.1 ms per variant.

Since by definition the edge matrix is the same across mutants (Figure S2), one could ask whether the pairwise distances are adding any useful information to the network. Thus, a model was trained without edge information (the weights of all edges were simply set to 1,  $e_{ij} = 1$ ), which resulted in a performance penalty (Table 1). This means that the extra information that the edge matrix provides to the network, i.e., the relative positions of the protein residues, does indeed improve performance. Additionally, one could ask whether the number of residues (nodes) selected beforehand to be part of the graph representation is optimal. After all, only 4–8 residues are allowed to mutate, and the other 15–19 residues are kept constant. The network may benefit from knowing the identities of more residues as this would provide additional information on the environment of the binding site. A model trained with 291 nodes did not improve performance, and on the contrary increased the evaluation time 10-fold and reduced the accuracy of the predictions (Table 1).

Table 2. Data Sets Used for Training<sup>a</sup>

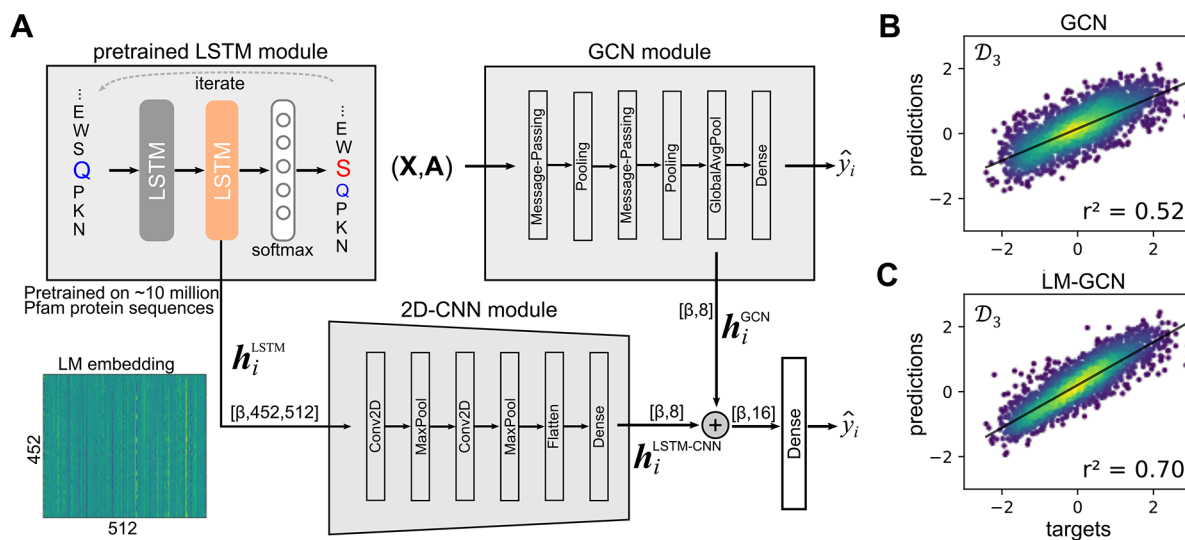
Name	$N_{hot}$	$N_{hot}$ positions	$L_{max}$ <sup>b</sup>	IAAI <sup>c</sup>	Search space <sup>d</sup>
D1	4	F19,W57,Y150,F85'	4	20	$1.6 \times 10^5$
D2	6	F19,W57,Y150,A228,R415,F85'	6	20	$6.4 \times 10^7$
D3	8	F19,W57,Y150,V225,A228,R415,F85',F86'	8	20	$2.56 \times 10^{10}$
D4	8	F19,W57,Y150,V225,A228,R415,F85',F86'	4	20	$1.12 \times 10^7$
D5	4	F19,W57,Y150,F85'	4	10	$1 \times 10^4$

<sup>a</sup>The number of examples per data set is 10,000 variants.  $\mathcal{D}_1$  is the main data set used in this study, it contains mutants of degrees 1, 2, 3, and 4 ( $L_{max} = 4$ ) in 4 positions ( $N_{hot} = 4$ ) that are allowed to mutate to any of the 20 standard amino acids (IAAI = 20). The search space of  $\mathcal{D}_1$  is therefore  $1.6 \times 10^5$ . Data set  $\mathcal{D}_3$  ( $L_{max} = 8$ ,  $N_{hot} = 8$ ) also resembles conditions relevant in enzyme design campaigns, with a search space of  $2.56 \times 10^{10}$ . In all cases, ligand is E4. <sup>b</sup>Maximum allowed mutant degree.  $L_{max} = 4$ , means that single ( $L = 1$ ), double ( $L = 2$ ), triple ( $L = 3$ ), and quadruple ( $L = 4$ ) mutants were allowed. <sup>c</sup>The number of amino acids allowed as target mutation, IAAI, was reduced to 10 in  $\mathcal{D}_5$ : AA = {A,C,D,E,G,H,I,K,L,M}, to see if the trained model could generalize to unseen amino acids, i.e., AA = {F,N,P,Q,R,S,T,V,W,Y}. <sup>d</sup>The search space for each data set was calculated with the following formula:  $C(N_{hot},L) \cdot IAAI^L$ , where  $C(N_{hot},L)$  is the combination of  $L$  items (mutant degree) taken from the set of size  $N_{hot}$  (number of hotspots), and IAAI is the number of amino acids allowed (normally 20).

Table 3. Performance of the GCN and LM-GCN Models on Various Data Sets<sup>a</sup>

Data set		GCN model				LM-GCN model			
		Train		Validation		Train		Validation	
Train	Eval.	$\uparrow r^2$	$\downarrow loss$	$\uparrow r^2$	$\downarrow loss$	$\uparrow r^2$	$\downarrow loss$	$\uparrow r^2$	$\downarrow loss$
D1	D1	0.819	0.182	0.813	0.188	0.808	0.158	0.805	0.194
D2	D2	0.574	0.427	0.552	0.435	0.667	0.310	0.651	0.356
D3	D3	0.502	0.498	0.497	0.489	0.642	0.347	0.625	0.368
D4	D4	0.596	0.405	0.588	0.405	0.601	0.399	0.597	0.405
D5	D5	0.786	0.213	0.778	0.227	0.852	0.152	0.855	0.147
D5	D1	–	–	0.602	0.403	–	–	0.638	0.361
D4	D3	–	–	0.396	0.587	–	–	0.449	0.542

<sup>a</sup>The architecture of the LM-GCN model is shown in Figure 6. The size of the data sets is 10,000 examples, with 80:20 split for training and validation, respectively.



**Figure 6.** LM-GCN model. A) The LM-GCN model consists of a pretrained LM module, a trained GCN module, and a 2D-CNN module. All layers from the LM and GCN modules are frozen when training the LM-GCN model during 200 epochs. The prediction reached by the combined LM-GCN model is better than the predictions of the GCN model alone.  $\beta$  represents the batch size. B) and C) are the correlations for the GCN and LM-GCN model, respectively. Training was done on the  $N_{hot} = 8$  data set. The 2D-CNN module using the LM embeddings ( $h_i^{LSTM} \in \mathbb{R}^{452 \times 512}$ , where 452 is the length of the enzyme sequence and 512 is the number of units in the pretrained LM) as input vectors can achieve a correlation  $r^2 = 0.43$  on its own, lower than the correlation of the GCN and LM-GCN models.

The data set insofar used for training and testing the NNs contained single, double, triple, and quadruple mutants ( $L = 1, 2, 3$ , and 4, respectively) on  $N_{hot} = 4$  positions (positions 19, 57, 150, and 85 are allowed to mutate). With  $N_{hot} = 4$ , the total combinatorial space is  $1.6 \times 10^5$  variants, which means that the

training data set of  $1 \times 10^4$  variants covered 6.25% of the total combinatorial space (Figure S5). Additional data sets with more hotspot positions and higher-order mutants were tested (Table 2). With  $N_{hot} = 6$ , the training data represents 0.015% of the total combinatorial space of  $6.4 \times 10^7$  variants. And with

$N_{hot} = 8$ , the training data represents a mere 0.000039% of the total search space of  $2.56 \times 10^{10}$  variants. Gratifyingly, the GCN models were still able to perform well on the prediction task but with a reduced accuracy ( $r^2 = 0.52$  for  $N_{hot} = 8$ ) (Table 3). An easy solution to increase the accuracy of the trained NN is to increase the number of examples in the training data set. Training the NN with a larger data set of  $5 \times 10^4$  variants (0.00019% of the total combinatorial space) marginally increased the accuracy to  $r^2 = 0.59$ . However, increasing the size of the training data set is not always the optimal solution, especially when more sophisticated and expensive methodologies are used to label the mutants. Instead, we increased accuracy by feeding vector embeddings from a pretrained language model (see below).

Unsurprisingly, the degree of the mutants in the training data set had an impact on the model's performance. A model trained and evaluated on a data set containing first–fourth order mutants (e.g.,  $\mathcal{D}_4$ ,  $r^2 = 0.588$ , loss = 0.405) performed better than a model trained and evaluated on a data set containing first–eighth order mutants (e.g.,  $\mathcal{D}_3$ ,  $r^2 = 0.497$ , loss = 0.489). Models trained with lower-order mutants did not generalize well to higher-order mutants, as shown by the results of the model trained on  $\mathcal{D}_4$  but evaluated on  $\mathcal{D}_3$  ( $r^2 = 0.396$  and loss = 0.587) (Table 3). Furthermore, the NN did not generalize to unseen amino acids either. A NN trained on the  $\mathcal{D}_5$  data set, which contained mutants with only 10 amino acids (Ala, Cys, Asp, Glu, Gly, His, Ile, Lys, Leu, or Met), was not particularly good at evaluating mutants from the  $\mathcal{D}_1$  data set, which contained mutants with all 20 amino acids ( $r^2 = 0.602$ , loss = 0.403) (Table 3).

**Vector Embeddings from a Pretrained Bidirectional LSTM Module Can Increase Accuracy.** The pretrained LSTM model maps the protein sequence,  $s_i$ , to a vector representation,  $h_i$ , that can be fed to the GCN model to improve predictions. The projection,  $s_i \rightarrow h_i$ , is made in the context of all the 10 million Pfam protein sequences with which the LSTM model was trained. Thereby, mutant sequences that are close to the sequences with which the LSTM was trained (e.g., mutant sequences that do not have a Pro residue within an  $\alpha$  helix region) will be close in the embedding space. The vector embeddings,  $h_i \in \mathbb{R}^{452 \times 512}$ , are passed through a 2D-CNN module to project them to a low dimensional 8- $d$  vector. The 8- $d$  vector is concatenated with the 8- $d$  vector produced by the trained GCN module, resulting in a 16- $d$  vector that can then be passed through as single dense layer with one output unit to generate a prediction,  $\hat{y}_i$ . The model presented in Figure 6 was trained keeping both the GCN and the pretrained LSTM modules frozen. After training for 200 epochs, the resulting LM-GCN model outputted better predictions than the GCN model alone (Figure 6B,C; Table 3). Originally, a representation learning strategy more similar to the one reported by Gligorijevic et al.<sup>9</sup> was attempted, using the hidden state representation from the last amino acid position of the input sequence of length  $|s_i| = 452$ . The mutations in the V $f$ -TA data set occur in positions 19, 57, 85, 86, 150, 255, 228, and 415, which means that the signal from the early substitutions (e.g., position 19) will have degraded by the time we arrive at the last position (Figure S8). Therefore, the embeddings coming from all positions were used, more similar to the strategy of Bepler and Berger.<sup>40</sup>

**Applicability in Protein Engineering Campaigns.** There are three limitations to consider for the application of

the presented methodology in protein engineering campaigns. 1) Generation of the training data set can be expensive. Scoring 10,000 mutants using Rosetta with 10 replicas each required 1,600 core-hours, which is equivalent to a runtime of 133 h on a desktop with 12 threads per CPU (HP workstation Z4 with an Intel Xeon W-2135 processor). Training the model only takes 0.5–2 h on a GPU (NVIDIA GTX 1080). Once the model has been trained, evaluation of new mutants takes less than 1 ms, which is  $1 \times 10^6$  times faster than Rosetta. 2) If *good mutants* are not already present in the training data set, the model is not expected to extrapolate to the region of good scores. However, we did find that a model trained only with bad mutants was still able to give the best scores it could to good mutants when tested (Figure S7). 3) Mutants that significantly change the protein backbone conformation might prove difficult because we set the edge attributes, i.e., the pairwise distances between protein residues  $C_\alpha$  atoms, to remain constant across mutants. In practice, we do not expect the protein backbone to change considerably with mutants of degree 1–8 (Figure S2) or Rosetta to be able to move the protein backbone to beyond  $\sim 1 \text{ \AA}$ .<sup>41</sup> Moreover, the NN was observed to still make predictions even when all edge attributes were set to 1.0 (Table 1), which means that while edge attributes help accuracy, they are not essential. While these limitations should generally not hinder applicability to protein engineering campaigns, we believe the importance of this study should be in the fact that a simple algorithm lacking much awareness about the identity of the molecules involved is capable of learning the intricate synergic relationships needed to assess new combinations of mutants.

## DISCUSSION

In this work, it has been shown that it is possible to train a neural network to learn the intricate synergic relationships needed to assess mutants resulting of the combination of two or more individual mutations. The main advantage of the presented deep learning strategy for predicting binding energies of protein–ligand complexes is the massive speed-up compared to traditional computational HTS screening methods. The trained DL model can, after  $\sim 2$  h of training, predict the binding energy of unseen enzyme variants in 1.36 ms with high accuracy, enabling super high-throughput screening. Computational strategies for fast screening of enzyme variants are useful because of the astronomic size of the combinatorial space that can be explored. Targeted mutagenesis efforts explore a limited number of variants in search of a good variant, typically by mutating 4–8 positions around the binding site, but the combinatorial space of 8 positions amounts to  $20^8$  possible variants. One possibility could be to generate all possible single mutants in 8 positions ( $8 \times 20 = 160$  variants) and combine the best amino acid from each position to generate double or higher-order mutants. However, such a strategy would not take correlations into account, i.e., the effect that mutating a position in the vicinity can have on the ability of the enzyme to catalyze the desired reaction. Another way to explore the combinatorial space without exhaustive screening of the entire combinatorial space is to generate library pathways, but it may produce a nonoptimal solution or a dead end.<sup>42</sup> By contrast, this work proposes to train a neural network to learn the combinatorial space from a few examples, and then use the trained network to scan the entire combinatorial space at high speed. Using NNs to solve the combinatorial libraries task has, thus, the



potential to reduce the experimental (or computational) effort required to explore the combinatorial space in search of good enzyme mutants.

Feature injection has been used before in biological problems to either achieve higher performance<sup>9,40</sup> or train with small data sets.<sup>4,15,43</sup> And, the idea of using pretrained models to learn continuous vector embeddings to be used in combinatorial libraries has previously been attempted.<sup>44</sup> In that early study, the method was unsuccessful and the authors note that protein design and protein classification (i.e., the task on which using the vector embeddings was successful) are two very distinct tasks and that the vector embeddings are only 100-*d*, which is not enough to represent a protein. In this study, the vector embeddings (sized [452,512]) come from a model pretrained on a data set of protein sequences 20 times larger.<sup>9,45</sup> The method hereby presented was successful in increasing the accuracy of the network upon injection. The idea was that the trained embeddings ( $h_i$ ) contain information about the query sequence ( $s_i$ ) in the context of all protein sequences seen during training ( $S = \{s_1, s_2, \dots, s_{\sim 10\,000}\}$ ). For example, mutation to Proline of a position within an  $\alpha$  helix would prevent the continuation of the  $\alpha$  helix, and decrease the ability of the LSTM model to predict the next residue in the sequence. The embeddings coming from this mutant sequence can be interpreted by the network to improve binding energy predictions (Figure S8).

In practice, the use-cases of the presented strategy are limited. The obvious application would be in exploring the entire combinatorial space to find the top-scoring variants (Figure 5C). However, the main problem is that the Rosetta-generated binding energies are never perfect, and the discovered variants may not necessarily work out in wet-lab experiments. The binding energies could be generated by more sophisticated or accurate methods (e.g., molecular dynamics simulations or experimental enzymatic activities) with the increased cost (computational or in lab equipment) in generating the training data set. Another possibility would be to train the model with a massive data set generated by a computationally inexpensive methodology (e.g., Rosetta) and later retrain it on the smaller but more accurate data set. Another application could be in exploring and comparing the entire combinatorial spaces of seemingly related enzymes (for example the  $\omega$ -TA from *Vibrio fluvialis* and *Chromobacterium violaceum*). Beyond potential use cases, the success of the presented methodology in evaluating unseen mutants by a neural network unaware of the identity of the ligand or any structural information also hints that the task may not be as complex as initially thought.

## CONCLUSIONS

The proposed deep learning strategy for predicting the binding energy of enzyme variants with the ligand of interest achieved a high accuracy after training with a data set sized a miniscule fraction of the total combinatorial space. The methodology leverages the high combinatorial power of neural networks to quickly learn patterns for combining mutants from a few examples. The naked GCN module achieved high accuracy when the number of mutation hotspots was small, but the accuracy decreased in data sets with a larger number of mutation hotspots. The combined LM-GCN module enhanced the predictions of the GCN module by injecting feature vectors generated by a pretrained LM module.

## ASSOCIATED CONTENT

### Data Availability Statement

Code is available at: <https://github.com/crp-mol/super-HTS>.

### Supporting Information

The Supporting Information is available free of charge at <https://pubs.acs.org/doi/10.1021/acs.jctc.2c01227>.

Additional figures of the structures of the ligands, protein complexes, training and testing procedures, extrapolation, and signal degradation (PDF)

## AUTHOR INFORMATION

### Corresponding Author

Siewert J. Marrink – Molecular Dynamics, Groningen Biomolecular Sciences and Biotechnology Institute (GBB), University of Groningen, 9747 AG Groningen, The Netherlands; [orcid.org/0000-0001-8423-5277](https://orcid.org/0000-0001-8423-5277); Email: [s.j.marrink@rug.nl](mailto:s.j.marrink@rug.nl)

### Author

Carlos Ramírez-Palacios – Molecular Dynamics, Groningen Biomolecular Sciences and Biotechnology Institute (GBB), University of Groningen, 9747 AG Groningen, The Netherlands; [orcid.org/0000-0003-2724-0760](https://orcid.org/0000-0003-2724-0760)

Complete contact information is available at: <https://pubs.acs.org/doi/10.1021/acs.jctc.2c01227>

### Notes

The authors declare no competing financial interest.

## ACKNOWLEDGMENTS

C.R.P. thanks CONACYT for the doctoral fellowship. We thank Dick Janssen for his support in this project.

## ABBREVIATIONS

GCN, graph convolutional network; ConvNet, convolutional neural network; DL, deep learning; MD, molecular dynamics;  $\omega$ -TA,  $\omega$ -transaminase; *Vf*-TA,  $\omega$ -transaminase from *Vibrio fluvialis*; LM, language model; LSTM, long short-term memory; CNN, convolutional neural network; NN, neural network; FFT, fast Fourier transform; MSE, mean squared error

## REFERENCES

- (1) Jemli, S.; Ayadi-Zouari, D.; Hlima, H. B.; Bejar, S. Biocatalysts: Application and Engineering for Industrial Purposes. *Crit. Rev. Biotechnol.* **2016**, *36* (2), 246–258.
- (2) Arnold, F. H. Directed Evolution: Bringing New Chemistry to Life. *Angew. Chem., Int. Ed.* **2018**, *57* (16), 4143–4148.
- (3) Acebes, S.; Fernandez-Fueyo, E.; Monza, E.; Lucas, M. F.; Almendral, D.; Ruiz-Dueñas, F. J.; Lund, H.; Martinez, A. T.; Guallar, V. Rational Enzyme Engineering Through Biophysical and Biochemical Modeling. *ACS Catal.* **2016**, *6* (3), 1624–1629.
- (4) Torng, W.; Altman, R. B. Graph Convolutional Neural Networks for Predicting Drug-Target Interactions. *J. Chem. Inf. Model.* **2019**, *59* (10), 4131–4149.
- (5) Sattarov, B.; Baskin, I. I.; Horvath, D.; Marcou, G.; Bjerrum, E. J.; Varnek, A. De Novo Molecular Design by Combining Deep Autoencoder Recurrent Neural Networks with Generative Topographic Mapping. *J. Chem. Inf. Model.* **2019**, *59* (3), 1182–1196.
- (6) Arús-Pous, J.; Johansson, S. V.; Prykhodko, O.; Bjerrum, E. J.; Tyrchan, C.; Reymond, J.-L.; Chen, H.; Engkvist, O. Randomized SMILES Strings Improve the Quality of Molecular Generative Models. *J. Cheminformatics* **2019**, *11* (1), 1–13.

- (7) Degiacomi, M. T. Coupling Molecular Dynamics and Deep Learning to Mine Protein Conformational Space. *Structure* **2019**, *27* (6), 1034–1040.e3.
- (8) Alley, E. C.; Khimulya, G.; Biswas, S.; AlQuraishi, M.; Church, G. M. Unified Rational Protein Engineering with Sequence-Based Deep Representation Learning. *Nat. Methods* **2019**, *16* (12), 1315–1322.
- (9) Gligorijević, V.; Renfrew, P. D.; Kosciolk, T.; Leman, J. K.; Berenberg, D.; Vatanen, T.; Chandler, C.; Taylor, B. C.; Fisk, I. M.; Vlamakis, H.; Xavier, R. J.; Knight, R.; Cho, K.; Bonneau, R. Structure-Based Protein Function Prediction Using Graph Convolutional Networks. *Nat. Commun.* **2021**, *12* (1), 3168.
- (10) Senior, A. W.; Evans, R.; Jumper, J.; Kirkpatrick, J.; Sifre, L.; Green, T.; Qin, C.; Židek, A.; Nelson, A. W. R.; Bridgland, A.; Penedones, H.; Petersen, S.; Simonyan, K.; Crossan, S.; Kohli, P.; Jones, D. T.; Silver, D.; Kavukcuoglu, K.; Hassabis, D. Improved Protein Structure Prediction Using Potentials from Deep Learning. *Nature* **2020**, *577* (7792), 706–710.
- (11) Dauparas, J.; Anishchenko, I.; Bennett, N.; Bai, H.; Ragotte, R. J.; Milles, L. F.; Wicky, B. I. M.; Courbet, A.; de Haas, R. J.; Bethel, N.; Leung, P. J. Y.; Huddy, T. F.; Pellock, S.; Tischer, D.; Chan, F.; Koepnick, B.; Nguyen, H.; Kang, A.; Sankaran, B.; Bera, A. K.; King, N. P.; Baker, D. Robust Deep Learning–Based Protein Sequence Design Using ProteinMPNN. *Science* **2022**, *378* (6615), 49–56.
- (12) Wang, J.; Lisanza, S.; Juergens, D.; Tischer, D.; Watson, J. L.; Castro, K. M.; Ragotte, R.; Saragovi, A.; Milles, L. F.; Baek, M.; Anishchenko, I.; Yang, W.; Hicks, D. R.; Expòsit, M.; Schlichthaele, T.; Chun, J.-H.; Dauparas, J.; Bennett, N.; Wicky, B. I. M.; Muenks, A.; DiMaio, F.; Correia, B.; Ovchinnikov, S.; Baker, D. Scaffolding Protein Functional Sites Using Deep Learning. *Science* **2022**, *377* (6604), 387–394.
- (13) Cao, Y.; Shen, Y. Energy-based Graph Convolutional Networks for Scoring Protein Docking Models. *Proteins Struct. Funct. Bioinforma.* **2020**, *88* (8), 1091–1099.
- (14) Zhao, T.; Hu, Y.; Valsdottir, L. R.; Zang, T.; Peng, J. Identifying Drug–Target Interactions Based on Graph Convolutional Network and Deep Neural Network. *Brief. Bioinform.* **2021**, *22* (2), 2141–2150.
- (15) Altae-Tran, H.; Ramsundar, B.; Pappu, A. S.; Pande, V. Low Data Drug Discovery with One-Shot Learning. *ACS Cent. Sci.* **2017**, *3* (4), 283–293.
- (16) Wang, J.; Cao, H.; Zhang, J. Z. H.; Qi, Y. Computational Protein Design with Deep Learning Neural Networks. *Sci. Rep.* **2018**, *8* (1), 6349.
- (17) Rodrigues, C. H.; Pires, D. E.; Ascher, D. B. DynaMut: Predicting the Impact of Mutations on Protein Conformation, Flexibility and Stability. *Nucleic Acids Res.* **2018**, *46* (W1), W350–W355.
- (18) Markel, U.; Essani, K. D.; Besirlioglu, V.; Schiffels, J.; Streit, W. R.; Schwaneberg, U. Advances in Ultrahigh-Throughput Screening for Directed Enzyme Evolution. *Chem. Soc. Rev.* **2020**, *49* (1), 233–262.
- (19) Singh, N.; Malik, S.; Gupta, A.; Srivastava, K. R. Revolutionizing Enzyme Engineering through Artificial Intelligence and Machine Learning. *Emerg. Top. Life Sci.* **2021**, *5*, 113–125.
- (20) Cadet, F.; Fontaine, N.; Li, G.; Sanchis, J.; Ng Fuk Chong, M.; Pandjaitan, R.; Vetrivel, I.; Offmann, B.; Reetz, M. T. A Machine Learning Approach for Reliable Prediction of Amino Acid Interactions and Its Application in the Directed Evolution of Enantioselective Enzymes. *Sci. Rep.* **2018**, *8* (1), 16757.
- (21) Wu, Z.; Kan, S. B. J.; Lewis, R. D.; Wittmann, B. J.; Arnold, F. H. Machine Learning-Assisted Directed Protein Evolution with Combinatorial Libraries. *Proc. Natl. Acad. Sci. U. S. A.* **2019**, *116* (18), 8852–8858.
- (22) Liao, J.; Warmuth, M. K.; Govindarajan, S.; Ness, J. E.; Wang, R. P.; Gustafsson, C.; Minshull, J. Engineering Proteinase K Using Machine Learning and Synthetic Genes. *BMC Biotechnol.* **2007**, *7* (1), 16.
- (23) Gomm, A.; O'Reilly, E. Transaminases for Chiral Amine Synthesis. *Curr. Opin. Chem. Biol.* **2018**, *43*, 106–112.
- (24) Kelly, S. A.; Pohle, S.; Wharry, S.; Mix, S.; Allen, C. C. R.; Moody, T. S.; Gilmore, B. F. Application of  $\omega$ -Transaminases in the Pharmaceutical Industry. *Chem. Rev.* **2018**, *118* (1), 349–367.
- (25) Park, E.-S.; Kim, M.; Shin, J.-S. Molecular Determinants for Substrate Selectivity of  $\omega$ -Transaminases. *Appl. Microbiol. Biotechnol.* **2012**, *93* (6), 2425–2435.
- (26) Tufvesson, P.; Lima-Ramos, J.; Jensen, J. S.; Al-Haque, N.; Neto, W.; Woodley, J. M. Process Considerations for the Asymmetric Synthesis of Chiral Amines Using Transaminases. *Biotechnol. Bioeng.* **2011**, *108* (7), 1479–1493.
- (27) Dong, L.; Meng, Q.; Ramírez-Palacios, C.; Wijma, H. J.; Marrink, S. J.; Janssen, D. B. Asymmetric Synthesis of Optically Pure Aliphatic Amines with an Engineered Robust  $\omega$ -Transaminase. *Catalysts* **2020**, *10* (11), 1310.
- (28) Meng, Q.; Ramírez-Palacios, C.; Capra, N.; Hooghwinkel, M. E.; Thallmair, S.; Rozeboom, H. J.; Thunnissen, A.-M. W. H.; Wijma, H. J.; Marrink, S. J.; Janssen, D. B. Computational Redesign of an  $\omega$ -Transaminase from *Pseudomonas jessenii* for Asymmetric Synthesis of Enantiopure Bulky Amines. *ACS Catal.* **2021**, *11*, 10733–10747.
- (29) Zhang, S.; Tong, H.; Xu, J.; Maciejewski, R. Graph Convolutional Networks: A Comprehensive Review. *Comput. Soc. Netw.* **2019**, *6* (1), 11.
- (30) Sun, M.; Zhao, S.; Gilvary, C.; Elemento, O.; Zhou, J.; Wang, F. Graph Convolutional Networks for Computational Drug Development and Discovery. *Brief. Bioinform.* **2020**, *21* (3), 919–935.
- (31) Johansson-Åkhe, I.; Mirabello, C.; Wallner, B. InterPepRank: Assessment of Docked Peptide Conformations by a Deep Graph Network. *Front. Bioinforma.* **2021**, *1*. DOI: 10.3389/fbinf.2021.763102
- (32) Bianchi, F. M.; Grattarola, D.; Alippi, C. Spectral Clustering with Graph Neural Networks for Graph Pooling. In *International Conference on Machine Learning*; PMLR, 2020; pp 874–883.
- (33) Ramírez-Palacios, C.; Wijma, H. J.; Thallmair, S.; Marrink, S. J.; Janssen, D. B. Computational Prediction of  $\omega$ -Transaminase Specificity by a Combination of Docking and Molecular Dynamics Simulations. *J. Chem. Inf. Model.* **2021**, *61* (11), 5569–5580.
- (34) Richter, F.; Leaver-Fay, A.; Khare, S. D.; Bjelic, S.; Baker, D. De Novo Enzyme Design Using Rosetta3. *PLoS One* **2011**, *6* (5), e19230.
- (35) Grattarola, D.; Alippi, C. Graph Neural Networks in TensorFlow and Keras with Spektral. *ArXiv200612138 Cs Stat* **2020**. DOI: 10.48550/arXiv.2006.12138.
- (36) Kawashima, S.; Pokarowski, P.; Pokarowska, M.; Kolinski, A.; Katayama, T.; Kanehisa, M. AAindex: Amino Acid Index Database, Progress Report 2008. *Nucleic Acids Res.* **2007**, *36* (suppl\_1), D202–D205.
- (37) Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; Kudlur, M.; Levenberg, J.; Monga, R.; Moore, S.; Murray, D. G.; Steiner, B.; Tucker, P.; Vasudevan, V.; Warden, P.; Wicke, M.; Yu, Y.; Zheng, X. TensorFlow: A System for Large-Scale Machine Learning. *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* **2016**, 265–283.
- (38) Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization. *ArXiv14126980 Cs* **2017**. DOI: 10.48550/arXiv.1412.6980.
- (39) Kipf, T. N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *ArXiv160902907 Cs Stat* **2017**. DOI: 10.48550/arXiv.1609.02907.
- (40) Bepler, T.; Berger, B. Learning Protein Sequence Embeddings Using Information from Structure. *ArXiv190208661 Cs Q-Bio Stat* **2019**. DOI: 10.48550/arXiv.1902.08661.
- (41) Lauck, F.; Smith, C. A.; Friedland, G. F.; Humphris, E. L.; Kortemme, T. RosettaBackrub—a Web Server for Flexible Backbone Protein Structure Modeling and Design. *Nucleic Acids Res.* **2010**, *38*, W569–W575.
- (42) Gumulya, Y.; Sanchis, J.; Reetz, M. T. Many Pathways in Laboratory Evolution Can Lead to Improved Enzymes: How to Escape from Local Minima. *ChemBioChem.* **2012**, *13* (7), 1060–1066.

- (43) Morris, P.; St. Clair, R.; Hahn, W. E.; Barenholtz, E. Predicting Binding from Screening Assays with Transformer Network Embeddings. *J. Chem. Inf. Model.* **2020**, *60* (9), 4191–4199.
- (44) Xu, Y.; Verma, D.; Sheridan, R. P.; Liaw, A.; Ma, J.; Marshall, N. M.; McIntosh, J.; Sherer, E. C.; Svetnik, V.; Johnston, J. M. Deep Dive into Machine Learning Models for Protein Engineering. *J. Chem. Inf. Model.* **2020**, *60* (6), 2773–2790.
- (45) Asgari, E.; Mofrad, M. R. K. Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics. *PLoS One* **2015**, *10* (11), e0141287.

## Recommended by ACS

### Interpretable Machine Learning of Amino Acid Patterns in Proteins: A Statistical Ensemble Approach

Anna Braghetto, Marco Baiesi, *et al.*

AUGUST 08, 2023  
JOURNAL OF CHEMICAL THEORY AND COMPUTATION

READ 

### QM/MM Modeling Aided Enzyme Engineering in Natural Products Biosynthesis

Fan Zhang, Ruibo Wu, *et al.*

AUGUST 09, 2023  
JOURNAL OF CHEMICAL INFORMATION AND MODELING

READ 

### Universal Approach to De Novo Drug Design for Target Proteins Using Deep Reinforcement Learning

Yunjiang Zhang, Shaorui Sun, *et al.*

FEBRUARY 05, 2023  
ACS OMEGA

READ 

### Mechanistic Insights into Enzyme Catalysis from Explaining Machine-Learned Quantum Mechanical and Molecular Mechanical Minimum Energy Pathways

Zilin Song, Peng Tao, *et al.*

MAY 18, 2022  
ACS PHYSICAL CHEMISTRY AU

READ 

Get More Suggestions >