This is an open access article published under an ACS AuthorChoice License, which permits copying and redistribution of the article or any adaptations for non-commercial purposes.

JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

pubs.acs.org/jcim Article

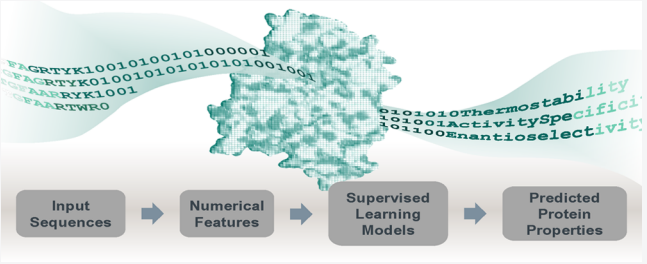
Deep Dive into Machine Learning Models for Protein Engineering

Yuting Xu,* Deeptak Verma, Robert P. Sheridan, Andy Liaw, Junshui Ma, Nicholas M. Marshall, John McIntosh, Edward C. Sherer, Vladimir Svetnik, and Jennifer M. Johnston*

Cite This: J. Chem. Inf. Model. 2020, 60, 2773−2790 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: Protein redesign and engineering has become an important task in pharmaceutical research and development. Recent advances in technology have enabled efficient protein redesign by mimicking natural evolutionary mutation, selection, and amplification steps in the laboratory environment. For any given protein, the number of possible mutations is astronomical. It is impractical to synthesize all sequences or even to investigate all functionally interesting variants. Recently, there has been an increased interest in using machine learning to assist protein redesign, since prediction models can be used to virtually screen a large number of novel sequences. However, many state-of-the-art machine learning models, especially deep learning models, have not been extensively explored. Moreover, only a small selection of protein sequence descriptors has been considered. In this work, the performance of prediction models built using an array of machine learning methods and protein descriptor types, including two novel, single amino acid descriptors and one structure-based three-dimensional descriptor, is benchmarked. The predictions were evaluated on a diverse collection of public and proprietary data sets, using a variety of evaluation metrics. The results of this comparison suggest that Convolution Neural Network models built with amino acid property descriptors are the most widely applicable to the types of protein redesign problems faced in the pharmaceutical industry.



1. INTRODUCTION

Protein redesign by directed evolution experiments has been widely used in the pharmaceutical industry to develop proteins with improved properties.[1] For example, enzymes are often used for the large-scale synthesis of drugs and drug precursors and must be good catalysts often under conditions different from those that would be optimal for natural enzymes. The directed evolution of proteins consists of multiple rounds of experiments that mimic natural selection. In each round various mutated sequences are created from the best available parent sequence, and desired properties (for example, percentage substrate conversion) are measured in vitro. Desirable variants, for example those that synthesize the most product at a given set of conditions, are carried forward to future rounds of evolution. The process of directed evolution is essentially an optimization problem in the ultrahigh dimensional protein sequence space. Identifying desirable mutations with a finite number of experiments remains challenging.[2] Opportunities to improve the efficiency of identifying desirable mutants through the application of predictive modeling are apparent. Machine learning models can learn relationships between sequences and properties from tested sequences in order to make predictions of the properties for virtual sequences. Computational approaches can therefore guide the design of future rounds of experiments to synthesize only the most promising sequences.[2] Recently the application of machine learning models in this field has attracted increasing attention.[3−5]

The application of machine learning to protein optimization requires descriptors that are appropriate for the information rich protein sequences. Some protein sequence descriptors have already been successfully applied to sequence-based protein classification and ligand docking problems, wherein descriptors such as amino acid composition (dipeptide, tripeptide composition),[6] predicted secondary structure and predicted solvent accessibility,[7] histograms of torsion angles density and amino acid distances density,[8] k-Spaced Amino Acid Pairs and Conjoint Triad,[9] and 3D grid protein−ligand structures[10] have been employed.

Despite this success for protein classification and ligand docking, the prediction of protein function, usually a continuous value measured by experiments, is quite a different task.[11] Machine learning guided protein engineering is still a new field; and there are few studies with demonstrable success in the current literature,[5,12−15] and these have tended to include only a small number of proteins and models.

Both Kimothi and colleagues[16] and Yang et al.[11] have applied the word embedding model doc2vec[17] to large protein sequence

Received: January 22, 2020
Published: April 6, 2020

Downloaded via 80.71.142.54 on December 26, 2022 at 12:41:12 (UTC).
See https://pubs.acs.org/sharingguidelines for options on how to legitimately share published articles.

© 2020 American Chemical Society

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790

data sets. These methods take cues from language processing, treating the protein sequence as documents and fragments of the sequences as words. Yang et al. used a single machine learning method, Gaussian process (GP) regression, to build predictive models with various input descriptors, including this type of embedding, alongside one-hot encoding, mismatch kernel, ProFET, and AAIndex properties, and compared the performance on four public data sets.

A more recent contribution by Wu et al.[5] demonstrated that a high throughput in silico model found improved mutants with reduced experimental effort for guided directed evolution. Several supervised learning methods were used; however, the input descriptors of the protein sequences were not discussed. Yang et al.[4] authored a review article covering some basic concepts of using machine learning for protein engineering, and this article gives two application examples. However, the discussion and recommendation for choosing a machine-learning sequence-function model for proteins are based on literature review, which can hamper quantitative interpretation of the results.

Aside from the variety of descriptors to select from, predicting biological properties from protein sequence is a challenging task because of the low signal-to-noise ratio data from experiments. The experimental measurement involves multiple steps of selection or amplification of in vitro protein samples. The resulting values from these low-volume, high-throughput experiments can show high variability. Promising variants are often reconfirmed or followed up at larger scale. Another challenge is extrapolation of predictions to include mutants that have not yet been seen in the existing data. Thus, it is critically important to develop accurate prediction models that perform well given the state of real-world data.

In this work, the predictive performance of 44 machine learning method and descriptor combinations is compared across a diverse collection of public and proprietary data sets using common evaluation metrics. Descriptor sets used herein include both widely used amino acid descriptors from the literature and novel, three-dimensional structure-based descriptors that have been developed specifically to address protein redesign problems. In addition, a Convolution Neural Network (CNN) model is proposed, which performed well in many examples herein.

2. METHODS

2.1. Protein Descriptors. There are two major types of protein descriptors based on the amino acid sequence of the protein subject: protein-level global descriptors and amino acid-level descriptors. The protein-level descriptors summarize key features of the protein, such as amino acid composition,[18] which is the fraction of each amino acid type within a protein sequence, or the autocorrelation descriptors,[19] describing the distribution of amino acid properties along the sequence. Sequence variants in directed evolution experiments via protein engineering are derived from a single parent backbone. Hence, the resulting variants are of the same length as the parent with a few amino acid changes. These highly similar aligned sequences enable the use of position-specific amino acid-level descriptors for model building. In this work, exploring alignment-based amino acid descriptors is prioritized in order to better understand the influence of mutations at specific locations, thus better reflecting the experimental objectives of protein evolution. We did not investigate many protein-level descriptors, such as ProFET or amino acid composition descriptors, since they are known not to perform well for protein engineering predictive tasks.[11]

A large number of single amino acid descriptor sets capturing different aspects of amino acids have been proposed in the literature.[20] Here, two popular single amino acid descriptor sets have been selected for comparison, the zScales and VHSE. In addition to existing widely used protein descriptors or amino acid descriptors, two sets of novel single amino acid descriptors, named "PCscores" and "sScales", are presented, which are derived from the AA-index database (https://www.genome.jp/aaindex/).[21] The PCscores descriptors are created by a data-driven dimension reduction approach from a full set of AA-indices, whereas the sScales descriptor represents the sum of a subset of AA-indices selected using domain knowledge. Details of sequence alignment-based descriptors can be found in Section 2.1.1.

In addition to sequence-based descriptors, a structure-based descriptor that employs the AA-index amino acid pairwise contact potential is also described for the first time. This novel descriptor, called sPairs, is introduced in Section 2.1.2. A further descriptor generation method, ProtVec,[22] that allows the construction of sequence descriptors using an embedding technique, is included and explained in Section 2.1.3. Finally, a mutation identity descriptor, MutInd, used as a baseline for comparison is described in Section 2.1.4.

2.1.1. Sequence-Based Feature Vector. Given the full data set of aligned sequences and a set of single amino acid descriptors, each protein sequence can be converted into a feature vector or matrix, as illustrated in Figure 1 (top). Each column represents a residue position in the alignment, and each row represents properties for the amino acid in that position. The number of rows depends on the nature and type of descriptor. For instance, for the identity descriptor, there would be 20 rows (one for each natural amino acid). If the residue was "A", the vector position corresponding to "A" would be "1", and all the other positions would be "0". For amino acid properties, the rows represent the scalar values of the amino acid in that position (size, hydrophobicity, charge, etc.) For principal components of properties, for instance zScales, there would be 5 rows each representing the component scalar value for that amino acid. Positions (columns) with no mutations exhibit the same vector value throughout the data set, whereas the positions with mutations exhibit different vector scores according to the nature of the mutation. These differences reflect the changes in the underlying biochemical and physicochemical properties of the data set. The protein descriptor matrix will be used as the input feature for CNN models. For other ML methods, a protein descriptor vector is constructed by concatenating the columns with mutations across the data set. Details of the single amino acid descriptor sets used in this study are described below. The names of single amino acid descriptor sets are used to refer to the corresponding sequence-based protein descriptors.

Identity. This descriptor is the one-hot-encoding binary vector of the 20 natural amino acid names.

zScales. This descriptor was proposed in ref 23 and is represented by a five-dimensional vector descriptor for each amino acid. These vectors were carefully extracted from a set of 26 descriptors reported in the literature (including those describing lipophilic, steric, and electronic properties of the amino acids) and condensed using the principal component analysis (PCA) approach.

VHSE. Similar to zScales described above, the VHSE (principal components score Vectors of Hydrophobic, Steric, and Electronic properties) descriptor was originally proposed in ref 24. The eight-dimensional vector is derived from PCA on

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790

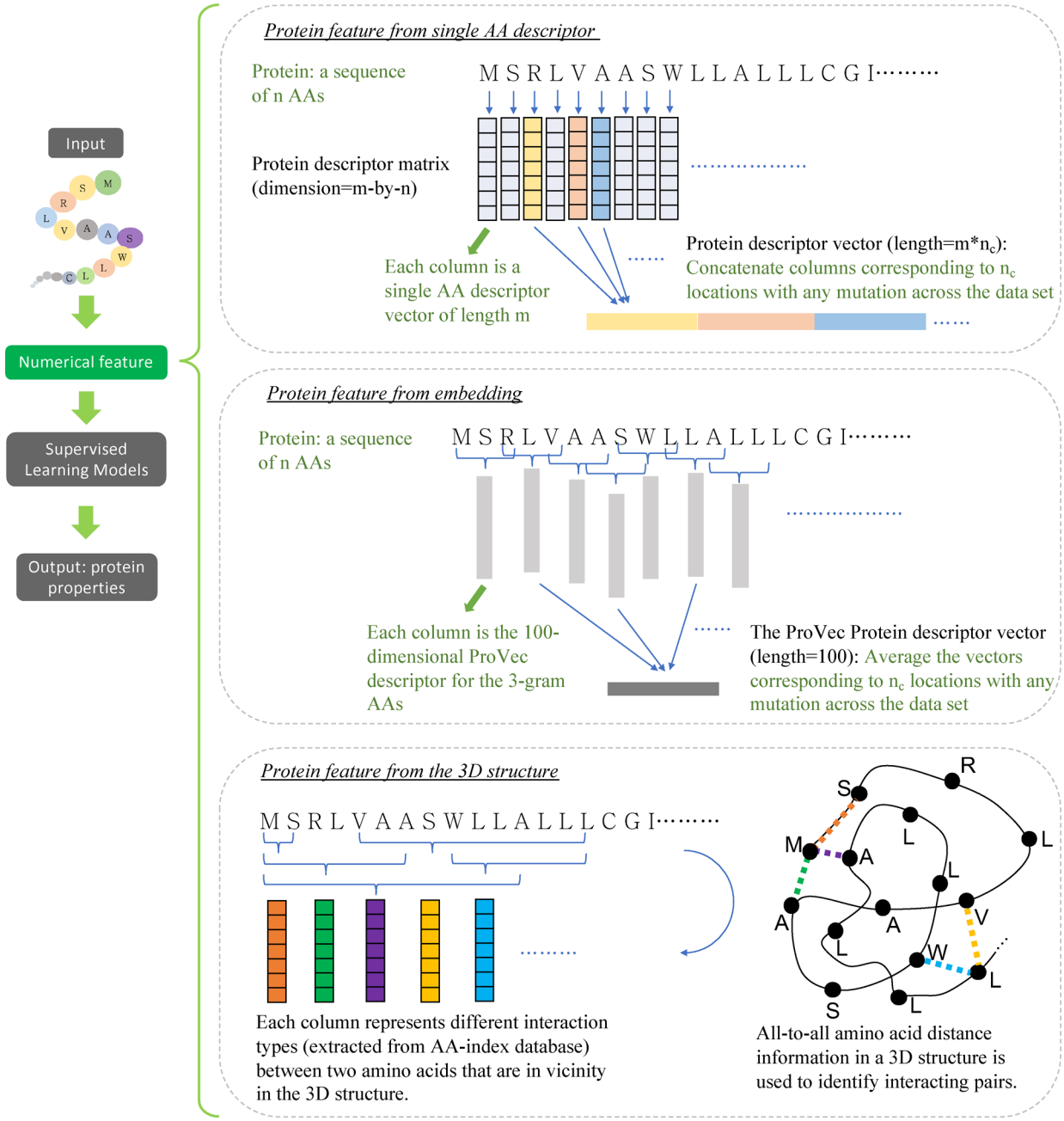


Figure 1. Illustration of different approaches for creating numerical descriptors from input protein sequence, which is the first step in building machine learning prediction models for proteins (left flowchart). Please note that the examples described here are for a single protein sequence, and the feature construction strategy can easily be expanded to represent a full data set of aligned sequences: (top) protein feature vector or matrix created from single amino acid properties and mutation locations; (middle) protein feature vector based on ProtVec, which is a widely used embedding feature of 3-gram; and (bottom) protein feature vector created from the 3D structure.

independent families of 18 hydrophobic properties, 17 steric properties, and 15 electronic properties, giving a total of 50 physicochemical variables of the 20 amino acids.

PCscores. AA-index is a database of published and curated numerical indices representing various physiochemical and biochemical properties of amino acids.[25] The first 11 principal components of the 20 × 533 matrix accounted for more than 90% of the variance in the data and thus were extracted as the feature vector.

sScales. The sScales feature construction is similar to the PCscores; however, the selection of the AA-index properties was based on the literature search instead of the data-driven dimension reduction approach. In addition, each AA-index descriptor was normalized, and the values were summed for each amino acid prior to constructing the sequence vector. This enabled the embedding of a diverse set of AA-index descriptors (representing physiochemical and biochemical properties) in a single value. The AA-index properties used herein are information value for accessibility:[26] average fraction 35%,[26] α helix propensity from T4 lysozyme,[27] normalized frequency of α helix,[28] volumes including the crystallographic waters using ProtOr,[29] amino acid composition of multispanning membrane proteins,[30] composition of amino acids in intracellular proteins (percentage),[31] conformational preference for beta-strands,[32] optimized relative partition energies - method C,[33] hydrophobicity index,[34] size,[35] and number of hydrogen bond donors.[36]

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790

2.1.2. Structure-Based Feature Vector. sPairs. In addition to the AA-index-based sequence descriptors, the AA-index amino acid pairwise contact potential, two-dimensional (2D) descriptors were employed. Each 2D index is represented by a 20 × 20 matrix, for example, the "statistical contact potential derived from 25 X-ray protein structures",[37] which can be accessed at https://www.genome.jp/dbget-bin/www_bget?aaindex:TANS760101. There are several ways to employ the 2D index values in the data set herein; however, constructing all possible pairs for each of the sequences within this data set to derive a numerical feature vector would result in a very large number of possible paired combinations, making model building slow and intractable. To construct a feature vector in a tractable way, it was necessary to include structural information for the protein of interest. The native structure from each of the data sets was used to construct a list of all possible paired positions that were within 8 Å in the structure, which is shown in Figure 1 (bottom). This filtered paired list allowed us to efficiently construct a structure-based feature vector from the 2D AA-index descriptors. It should be noted that only a single 3D structure was used to extract the paired position information. The same list of positions was carried forward to the full data set to construct a feature vector for every sequence. Any mutation of the amino acid at a given position would result in a change in all of the vector scores that position is interacting with—a change that could explain the differences in the measured experimental values by the machine learning model. The structure was either taken directly from the Protein Data Bank (www.pdb.org)[38] or constructed via homology modeling techniques. More details are provided in Section 3: Data Sets. Each structure was processed, cleaned, and minimized with restraints (heavy atoms converged to a root-mean-square deviation of 0.30 Å and an H-bond assignment using PROPKA[39] at pH 7.0) in Schrodinger (Maestro version 11.8.012)[40] using the Protein Preparation Wizard. Where necessary, the homology models were constructed using the Advanced Homology Toolbox with default settings in Schrodinger.

2.1.3. Embedding Representation. ProtVec. ProtVec is an unsupervised, data-driven representation where each protein sequence is represented by a dense vector created by embedding. Word embedding models in Natural Language Processing (NLP) have been used to extract features of protein sequences and have been successfully applied to predicting protein families or properties.[11,16,22,41] ProtVec[22] demonstrated that a novel NLP representation of protein sequences, like word2vec,[42,43] can be used to build a protein family classification model with much higher accuracy than previous approaches. Asgari et al.[22] used the Swiss-Prot database to train an unsupervised embedding model solely on sequences and provided the 100-dimensional vector representations for all 3-gram (consecutive triplets of amino acids). Figure 1(middle) illustrates how to compute the ProtVec protein feature vector from the embedding descriptors of 3-gram. Beginning with all overlapping 3-gram in a protein sequence, those related to positions without mutation across the data set have been removed, and the average of the embedding feature vectors of the remaining 3-gram was taken as the protein feature vector.

2.1.4. Mutation Indicator. MutInd. The MutInd descriptor is a binary vector with elements 0 or 1 indicating the presence of specific mutations in a sequence. This approach uses a list in the format "oldAA-position-newAA" to describe all possible mutations in a data set, where the "position" is an integer indicating the sequence location of a mutation when compared to its parent sequence, and the "oldAA" and "newAA" are the amino acid in parent sequence and variant protein sequence, respectively. The "MutInd" descriptor vector has the same length as the mutation list, with elements 0 or 1 indicating whether the corresponding mutation occurred in the variant sequence. For example, given a data set of three aligned proteins, in which the two variants have a single mutation from "A" to "D" or "G" at location 256 (see Table 1 below), the MutInd descriptor is a binary vector with two elements, indicating the presence of mutations "A256D" and "A256G".

Table 1. Example to Illustrate the MutInd Descriptor

	sequence (position 256 is shown in bold)	mutation A256D indicator	mutation A256G indicator	mutInd descriptor
parent	...VL**A**KP···	0	0	(0,0)
variant 1	...VL**D**KP···	1	0	(1,0)
variant 2	...VL**G**KP···	0	1	(0,1)

2.2. Performance Metrics. To compare different models, predictions on test sets have been evaluated using the following three metrics:

- RMSE: The root-mean-square error measuring the difference between predicted and observed values.
- R-squared: The squared Pearson correlation coefficient between predicted and observed activities is unitless and ranges from 0 to 1 for all data sets.
- AUROC: The Area Under the Receiver Operating Characteristic (ROC) curve is frequently used to describe the goodness of a binary classifier. It is equal to the probability that the predictor will rank a randomly chosen positive sample higher than a randomly chosen negative sample.[44] Usually it ranges between 0.5 and 1, where 0.5 means the classifier performs no better than random guesses, and 1 if the classifier perfectly separates positive samples from negative. Here, AUROC has been used in the analysis of results from proprietary data sets, where the positive or negative class will be defined given the experimental values.

2.3. Machine Learning Algorithms. Prediction models were constructed using labeled protein sequence data, employing the protein sequence feature vector or matrix as input and predicting the target protein property, a continuous variable. In this study, a wide selection of supervised learning algorithms to perform the regression task has been explored. Although many of the methods in comparison have significant precedent across the machine learning literature and have been widely applied to problems in the pharmaceutical industry, they have not yet been extensively used for protein engineering tasks.

To evaluate the performance of the different methods and descriptors for predicting protein properties, prediction models were trained for each data set in Section 3 using all possible combinations of 6 methods and 8 descriptors. Among the 6 machine learning algorithms, most of them take vectors as input thus can use all descriptors. However, CNN requires a 2D matrix input, so it can only use Identity, PCscores, VHSE, and zScales descriptors. Therefore, there are 44 combinations of method and descriptor in total. For each data set, all the prediction models were trained on the training set and evaluated on the test set, where the training and test splits are as described in Section 3.2.

Fivefold cross-validation was used for hyperparameter tuning of all machine learning algorithms. For each machine learning

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790

algorithm, a set of hyperparameters expected to have significant impact on model performance was specified, and grid-search was performed to find optimal values under each set of protein descriptors (see Table 2 for details). The optimal hyperparameter values for all choices of data set/ML algorithm/descriptor are provided in the Supporting Information.

Table 2. Hyperparameter Settings for Tuning Procedure

method	tuning parameters	fixed parameters
GLMNET	alpha (mixing percentage) = 0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0	
	lambda (regularization) = 2^{-4}, 2^{-3}, ..., 2^{2}	
SVM	sigma (inverse kernel width) = 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}	
	C (cost) = 2^{0}, 2^{2}, 2^{4}, ..., 2^{10}	
RF	n_tree = 200, 500, 1000	
	max_features = log2, sqrt, 0.33	
XGBoost	nrounds = 500, 1000, 1500	gamma = 0
	max_depth = 10, 15	subsample = 0.5
	eta = 0.01, 0.05	min_child_weight = 1
	colsample_bytree = 0.3, 0.667	
MLP	batch size = 10, 20, 40	optimizer = SGD
	learning rate = 0.001, 0.005, 0.01	momentum = 0.9
	initializer scale for uniform distribution = 0.01, 0.05	eval_metric = mse
	seven different model structures: number of hidden layers ranging from 1 to 4, with number of hidden nodes varies	maximum training epochs = 300
CNN	batch size = 20, 40, 60	optimizer = nesterov_momentum
	learning rate = 0.001, 0.005, 0.01	momentum = 0.9
	10 different model structures: number of convolution layers ranging from 1 to 4, number of dense layers = 1 or 2, with number of hidden nodes varies	objective = mse
		maximum training epochs = 500
		initialization = HeNormal

Once the optimal hyperparameter set was determined, the final model was trained on the whole training set using these hyperparameters. The performance metrics in Section 2.2 were then calculated on the test set predictions from the final models and experimentally observed protein properties. For models with a stochastic training process, such as Random Forests or neural nets, the final model was trained 10 times with different random seeds, and the median of evaluation metric across repeats was used for comparison.

Regularized Regression (GLMNET). GLMNET stands for the Elastic-Net Regularized Generalized Linear Models.[45] It is a regularized linear regression model where the penalty term for the loss function is a linear combination of L_1 (LASSO) and L_2 (ridge) penalty

$$\lambda^*[(1-\alpha)/2\|\beta\|_2^2 + \alpha\|\beta\|_1], \quad \lambda \geq 0,\ 0 \leq \alpha \leq 1$$

in which β is the vector of regression coefficients, the hyperparameter α is called the mixing parameter, and λ is the multiplication factor for penalty. The linear regression model is easy to interpret, yet the method performs well in many situations.

Support Vector Machine Regression (SVM). The SVM regression model is a nonlinear method[46] and has been successfully applied to many problems in chemistry and drug design.[47−49] The method implicitly transforms the input descriptors into a high dimensional space and fits the linear model in that space. The implicit transformation is achieved through a kernel function. In this work the RBF kernel with parameters σ and C was used, where σ controls the width of the kernel function.

Random Forest (RF). The RF regression model[50,51] is an ensemble of trees, in which each tree model is trained on a bootstrap sample of the training data, and at each node, a random subset of descriptors is tested as candidates for splitting. For prediction, the outputs from all trees are averaged. Two hyperparameters, fraction of input variables used at each split (max_features), and the number of trees (n_tree) were tuned in this work.

Extreme Gradient Boosting (XGBoost). Gradient tree boosting is another popular ensemble learning method with performance generally comparable to that of RF.[52,53] While RF builds independent trees, gradient boosting builds each tree sequentially, where each tree fits the residuals of predictions from all previous trees. XGBoost is an efficient implementation of gradient boosting.[54] Several hyperparameters were incorporated for tuning the models in this work: the number of shallow trees (nrounds), the maximum depth of a tree (max_depth), the boosting step size shrinkage (eta), and the subsample ratio of input variables for each tree (colsample_bytree).

Multilayer Perceptron (MLP). An MLP model is a fully connected feedforward neural network, with one or more hidden layers. When there are multiple hidden layers, it is called Deep Neural Network (DNN). An example small-scale MLP architecture is illustrated in Figure 2(a). In this work, the rectified linear unit (ReLU) was used as the activation function for all hidden layers. The training data from various data sets and protein descriptors may require different network architectures to achieve better performance. Thus, the network architecture was considered as one of the tuning parameters, and several choices of network architectures with different number of hidden layers and number of neurons in each layer were prespecified. The stochastic gradient descent (SGD) optimizer with momentum fixed at 0.9 was used in the training procedure. Several options of minibatch size (10/20/40) and learning rate (0.001/0.005/0.01) were explored, and the initial weights in the network were initialized by samples from the uniform distribution $[-u, u]$, where u is either 0.01 or 0.05.

Convolutional Neural Network (CNN). CNN is a special class of a neural network model commonly used in tasks with images as input.[55−59] In the CNN model architecture, each hidden node receives a signal from a small region of the previous layer. By using such local connections instead of fully connected layers, the convolutional layers can efficiently handle the spatial dependency in images and can extract meaningful high-level features. Usually a CNN model takes 2D images with multiple channels (e.g., the RGB color channels) as input, but 1D CNNs are possible for certain data types.[60−62]

In this work, a 1D CNN model has been proposed for protein sequence data, where the input protein feature is in a matrix format as described in Figure 1 (top), and the 1D convolution filter is operated along the amino acid sequence dimension (columns) in order to extract high-level features from the neighboring positions. The rows, or multiple amino acid properties at each position, are also called multiple input channels.

A typical 1D CNN architecture is illustrated in Figure 2(b). In this example, an input protein sequence consists of 300 amino acids, and the 11-dimensional PCscores descriptor is used as a single amino acid feature. The input feature for one protein is a 11×300 matrix represented by 11 vertical lines in the first block of Figure 2(b). In a convolutional layer, each node receives input from positions within a small neighborhood in the previous layer,[63] and the subsequent max-pooling layer performs downsampling along the sequence dimension to reduce the number of

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790

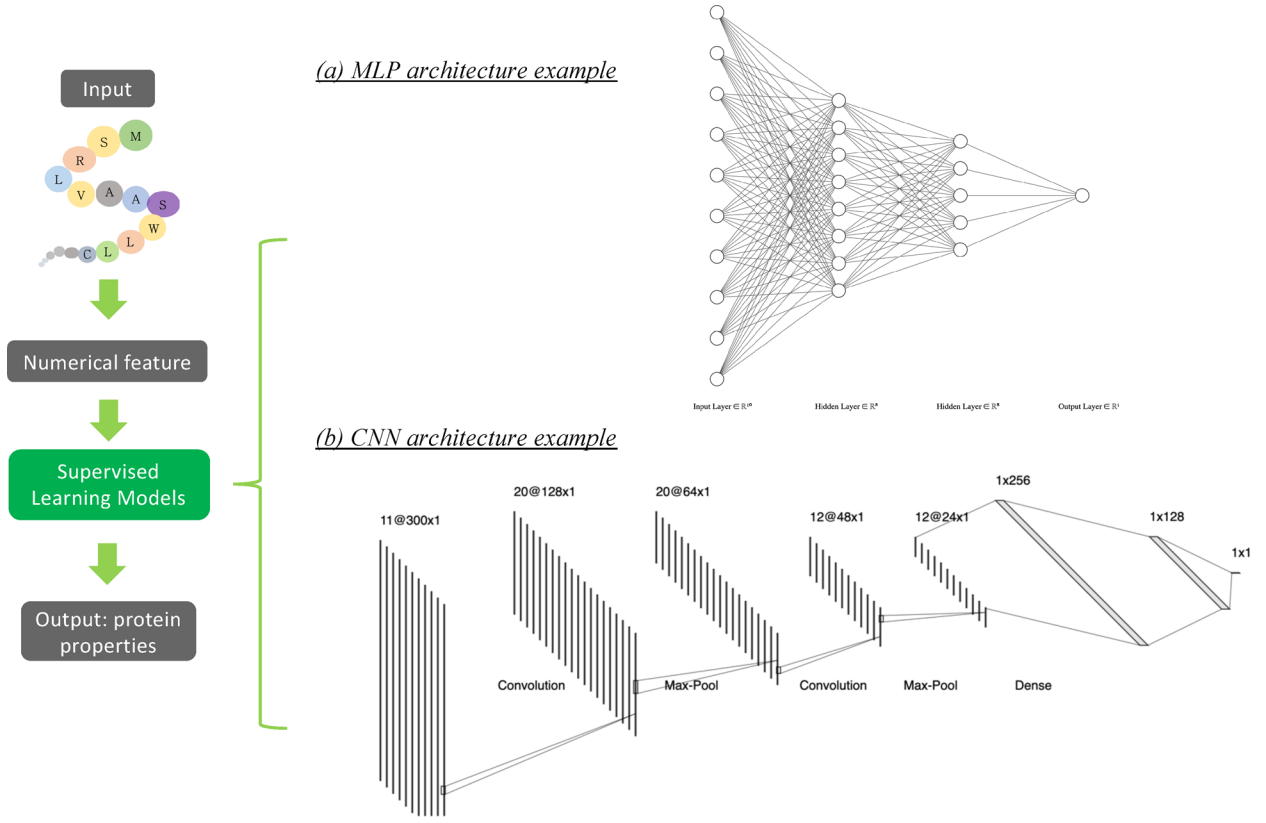


Figure 2. Example diagram of the two neural network models: (a) simple MLP architecture with two hidden layers and (b) simple CNN architecture with two convolution layers and two dense layers. Here we assume that the input protein sequence length is 300, and the 11-dimensional PCscores is used as a single amino acid descriptor, which results in a protein feature matrix with dimension 11 × 300.

extracted features. Each convolution layer contains multiple filters, where each 1D convolution filter functions as a sliding window, creating a new sequence of features from the multiple channels of features in the previous layer. For example, the first convolution layer in Figure 2(b) consists of 20 filters and creates a 20-channel output feature (the 20 vertical lines) for the next layer, and the first max-pooling layer takes the maximum value for each of the two elements along the sequence, thus reducing the feature sequence length by 2. Multiple convolution and max-pooling layers can be applied to the input feature matrix for feature extraction. Finally, the output from convolutional layers is flattened into a "fingerprint vector", and several fully connected layers are used to further process the high-level features and perform the regression task.

Like MLP, the layout of the CNN architecture needs to be prespecified. There are many choices to be made when it comes to the model architecture, such as the number of convolution and max-pooling layers, the number of filters in each convolution layer, the filter size and stride, the number of fully connected layers after flattening the CNN output features, etc. Here, several candidate model architectures were specified to choose from as part of the model tuning process. For other hyperparameters such as minibatch size and learning rate, a grid-search using values shown in 2 was performed.

2.4. Implementation. The CNN model structure was implemented in Python using the Lasagne[64] library, and Theano[65] was used for GPU computing. The MLP model was implemented using the MXNet[66] Python package in GPU mode. The RF model was implemented by the Scikit-learn[67] Python library, and the rest of the machine learning algorithms were implemented in R using glmnet,[45] kernlab,[68] xgboost,[69] and caret[70] packages. The hardware platform used in this study was Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40 GHz and one NVIDIA TITAN X (Pascal) GPU card.

3. DATA SETS

3.1. Public and Proprietary Data Sets. The methodologies described above have been applied to several publicly available and proprietary data sets. The publicly available sets span different classes of protein (membrane, globular) and different experimentally observed properties to be predicted. This is valuable, since it permits consideration of the generality of the principles proposed herein to a wide variety of cases. The proprietary data sets are exclusively enzymes, but owing to their proprietary nature they have been labeled Enzyme A through Enzyme D. The measured outcome for each of the proprietary enzymes is substrate conversion (i.e., the fraction of a reactant that is converted to product on the enzyme in a given time), and so while the chemistry is different in each case, the methodology is comparable between enzymes. The raw experimental outcome is normalized and transformed to facilitate quantitative modeling. Table 3 summarizes the data sets included in this analysis.

3.2. Training and Test Splits. A typical protein engineering workflow includes multiple rounds of experiments, wherein each round may introduce new mutations at positions both previously explored and not previously explored, gradually expanding the scope of the sequence space. In order to simulate the real-world application scenario, the training data and test data are split by time (i.e., data from earlier rounds are used as training data and those from later rounds are used as test data).[74]

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790

Table 3. Summary of Data Sets

	data set	source (ref)	observable	protein	sequence length
public	Public-A	Arnold Lab[71]	absorption	Bacteriorhodopsin	298
	Public-B	Reetz Lab[72]	enantioselectivity	epoxide hydrolase	398
	Public-C	Arnold Lab[5]	enantioselectivity	bacterial hemoglobin	145
	Public-D	Sarkisyan et al.[73]	fluorescence	GFP	238
proprietary	Proprietary-A	internal data	conversion	Enzyme A	476
	Proprietary-B	internal data	conversion	Enzyme B	229
	Proprietary-C	internal data	conversion	Enzyme C	762
	Proprietary-D	internal data	conversion	Enzyme D	762

Table 4. Number of Mutations in Each Training/Test Set

Data Set	number of mutations												
	0	1	2	3	4	5	6	7	8	9	10	11	12−15
Public-A													
training	1	52	6	3	0								
test	0	0	0	4	15								
Public-B													
training	1	3	9	9	14	32	23	40	5				
test	0	8	5	2	1	0	0	0	0				
Public-C													
training	1	24	57	139	34	0	0						
test	0	0	0	0	0	184	134						
Public-D													
training	1	40	459	0	0	0	0	0	0	0	0	0	0
test	0	1044	12318	12336	9387	6825	4298	2526	1364	627	299	118	73

Public Data Sets. For the Public Data sets A and B, the same time-splits used in Yang et al.[11] have been adopted. Since the experimental chronology is not available in Public Data set C and D, training sets were created by selecting sequences with fewer mutations, which can be thought of as mimicking the real-world application case in which earlier rounds of evolution have fewer mutations from the wild-type than later rounds. The training set of Public Data set C was created by selecting sequences with 5 or fewer mutations, while the test set contains sequences with more than 5 mutations. Public Data set D is a large data set containing 50,000+ proteins, while practically there are only a few hundred sequences measured and used as a training set. Thus, the training set of Public Data set D is a random sample of 500 sequences with 2 or fewer mutations.

Detailed tables of mutation counts are provided in Table 4, and the summary statistics of training/test sets in public data sets are shown in Table 5. Figure 3 demonstrates that such training/test splits create different distributions of mutation counts and properties.

Table 5. Summary Statistics for Public Data Sets

data set	n_{total}	$n_{training}$	$percent_{training}$	n_{test}	$percent_{test}$
Public-A	81	62	76.5%	19	23.5%
Public-B	152	136	89.5%	16	10.5%
Public-C	573	255	44.5%	318	55.5%
Public-D	51715	500	1.0%	51215	99.0%

Proprietary Data Sets. All of the proprietary data sets have multiple rounds of experimental data. The sequences from earlier rounds were used as the training set, and those from later rounds were used as the test set. The split was chosen such that approximately 75%−85% of data was used for training. Table 6 displays the summary statistics of training/test splits for each proprietary data set. Similarly, Figure 4 shows the difference in distribution between training/test sets in terms of mutation counts and measured protein properties. In these proprietary experiments, the goal is to optimize proteins to achieve higher conversion rate, so the test sets (measured in later rounds) show improved experimental outcome conversion as well as more mutations.

4. RESULTS

To evaluate the performance of the different methods and descriptors for predicting protein properties, prediction models were trained for each data set using all available combinations of descriptors. Since CNN requires 2D matrix input, there are 44 choices of method and descriptor combinations. For each data set, all the prediction models were trained using the training set and evaluated on the test set, where the training and test splits are described in Section 3.2.

Hyperparameter tuning and cross-validation of all machine learning algorithms were performed as described in Section 2.3. The final prediction models were trained on all data points in each training set. The performance metrics in Section 2.2 were calculated from the test set predictions and true protein properties to enable comparison.

4.1. Public Data Sets. Table 7 and Table 8 show the summary statistics of scaled-RMSE and R-Squared on a test set with various ML methods and descriptors. For each data set and evaluation metric, the performance of 44 final models (i.e., the 44 descriptor/method combinations) was ranked according to the median value in increasing order for RMSE and decreasing order for R-Squared. The medians of the performance metrics and their ranks are provided in the Supporting Information.

In order to compare the performance across data sets and performance metrics, the median of ranks by ML methods in Table 9 and by the combination of descriptor/method in Table 10

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790

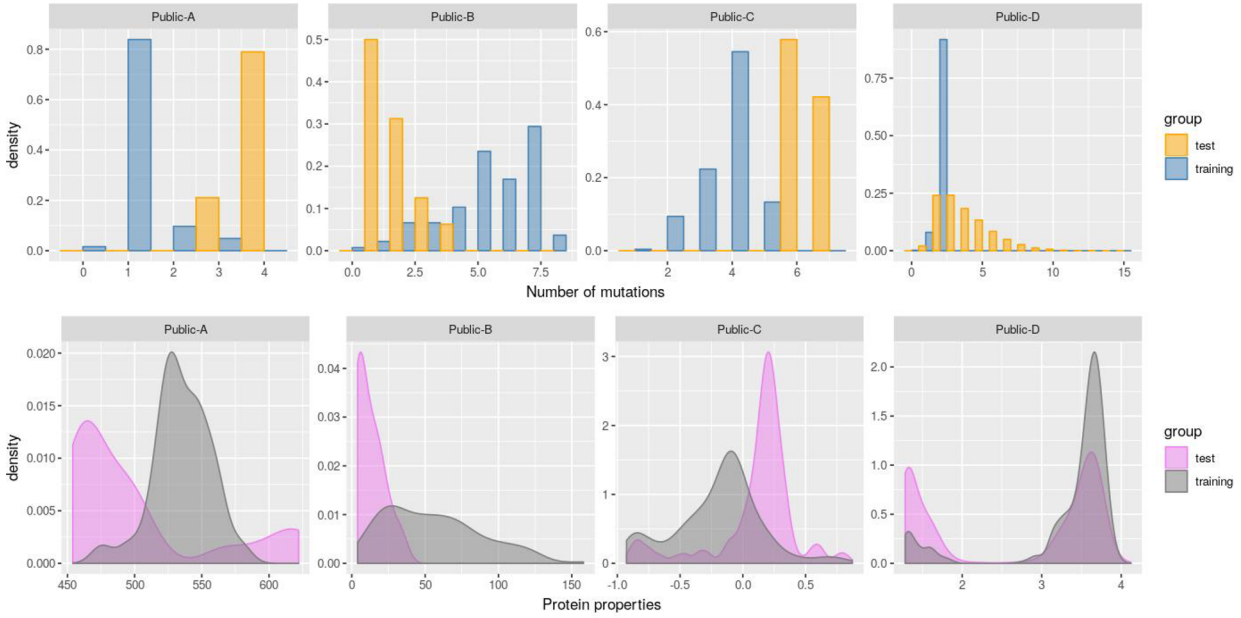


Figure 3. Comparison of training and test sets for each public data set: top row, density histogram of number of mutations, and bottom row, distribution of observed protein properties.

Table 6. Summary Statistics for Proprietary Data Sets

data set	n_{total}	$n_{training}$	$percent_{training}$	n_{test}	$percent_{test}$
Proprietary-A	3306	2628	79.5%	678	20.5%
Proprietary-B	2056	1640	79.8%	416	20.2%
Proprietary-C	4352	3657	84.0%	695	16.0%
Proprietary-D	4885	4127	84.5%	758	15.5%

was further calculated. The CNN model has the best median rank, 10.5, among all machine learning algorithms, and the CNN model with the PCscores protein descriptor achieves the best median rank of 3 among all 44 combinations of descriptor/method (thus, the upper and lower bounds for the ranks are 44 and 1).

Table 11 shows the average computational time in seconds across data sets of training the final model for each descriptor/method combination. Clearly there is a trade-off between accuracy and computational cost, where the top performing CNN models took more computational time during training. Although the two neural network models take longer than others, they have the unique advantage that a trained model with optimized weights can be easily updated with additional training data in the future, to accommodate subsequent rounds of experimental information as they become available.

4.2. Proprietary Data Sets. Table 12 and Table 13 show the summary statistics of scaled-RMSE and R-Squared on proprietary test sets with various ML methods and descriptors.

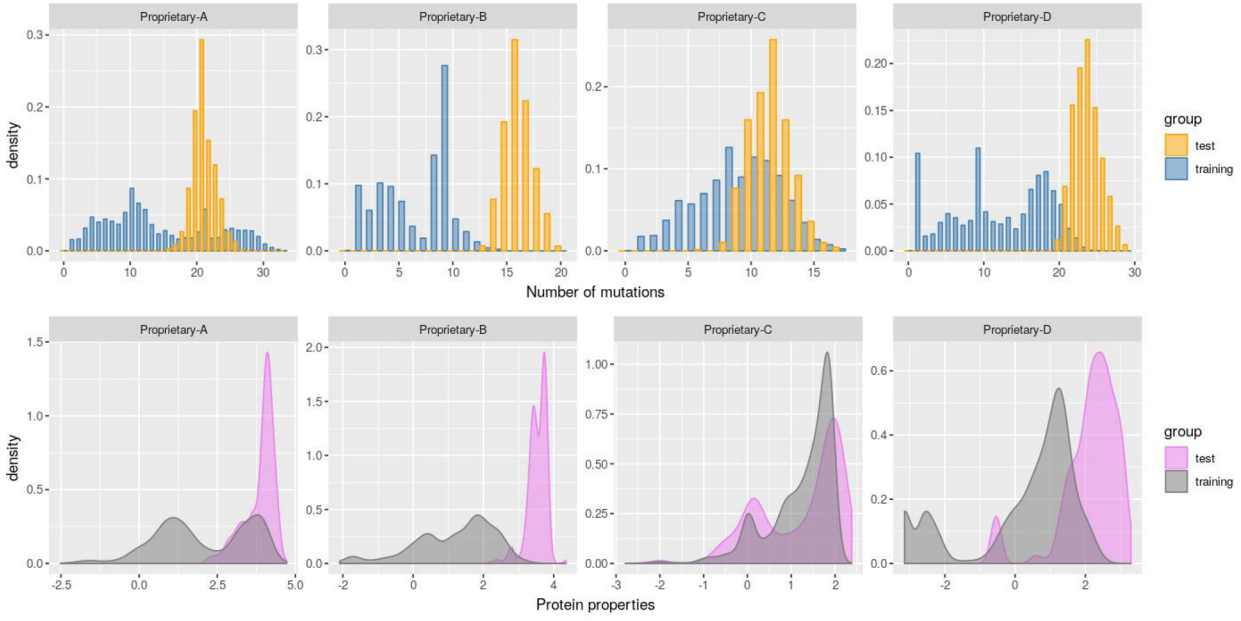


Figure 4. Comparison of training and test sets for each proprietary data set: top row, density histogram of number of mutations, and bottom row, distribution of observed protein properties.

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790

Table 7. Median Scaled-RMSE of Public Test Sets Predictions from Models Trained with Different ML Algorithm and Descriptor Combinations[a]

data set	method	Identity	mutInd	PCscores	protVec	sPairs	sScales	VHSE	zScales
Public-A	CNN	1.228(0.054)		1.067(0.168)				1.232(0.065)	1.135(0.064)
	GLMNET	1.116	1.136	1.212	1.175	1.533	1.98	1.266	1.199
	MLP	1.068(0.028)	1.116(0.068)	1.071(0.099)	1.525(0.087)	1.22(0.04)	1.674(0.19)	1(0.056)	1.22(0.113)
	RF	1.345(0.008)	1.404(0.015)	1.344(0.014)	2.383(0.037)	1.397(0.007)	1.321(0.01)	1.322(0.02)	1.339(0.024)
	SVM	1.127	1.216	1.132	1.574	1.208	1.864	1.192	1.592
	XGB	1.14(0.018)	1.108(0.012)	1.157(0.028)	2(0.04)	1.189(0.011)	1.148(0.016)	1.143(0.03)	1.155(0.008)
Public-B	CNN	1(0.259)		1.027(0.285)				1.364(0.182)	1.325(0.26)
	GLMNET	2.122	1.556	1.716	2.158	2.991	4.514	3.025	1.945
	MLP	1.864(0.735)	1.507(0.244)	2.339(0.906)	1.865(0.412)	3.272(0.491)	3.686(1.431)	1.832(0.495)	2.098(0.538)
	RF	1.669(0.074)	1.222(0.034)	1.945(0.069)	1.936(0.056)	1.582(0.065)	1.84(0.124)	2.189(0.079)	1.983(0.069)
	SVM	1.708	1.896	1.653	1.153	2.648	2.243	1.964	1.889
	XGB	1.315(0.11)	1.177(0.082)	1.334(0.061)	1.46(0.149)	1.451(0.067)	1.434(0.055)	1.506(0.058)	1.385(0.077)
Public-C	CNN	1.203(0.125)		1.169(0.138)				1.208(0.078)	1.262(0.088)
	GLMNET	1.054	1.132	1.126	1.316	1.239	1.016	1.271	1.445
	MLP	1.2(0.078)	1.436(0.116)	1.272(0.131)	1.397(0.091)	1.423(0.047)	1.069(0.16)	1.422(0.141)	1.34(0.056)
	RF	1.142(0.019)	1.33(0.025)	1.262(0.025)	1.285(0.025)	1.43(0.013)	1.075(0.028)	1.313(0.013)	1.232(0.019)
	SVM	1.002	1.135	1.015	1.159	1.324	1.377	1.136	1.279
	XGB	1.094(0.019)	1.276(0.013)	1(0.022)	1.091(0.016)	1.318(0.022)	1.067(0.022)	1.087(0.013)	1.045(0.059)
Public-D	CNN	1.198(0.03)		1.138(0.024)				1.146(0.037)	1.161(0.014)
	GLMNET	1.25	1.349	2.531	1.501	1.557	8.022	1.561	1.794
	MLP	1.166(0.016)	1.381(0.009)	1.271(0.026)	1.362(0.085)	1.215(0.051)	1.486(0.191)	1.076(0.041)	1.11(0.042)
	RF	1.236(0.002)	1.492(0.004)	1.216(0.001)	1.232(0.005)	1.139(0.001)	1.253(0.001)	1.21(0.001)	1.22(0.001)
	SVM	1.092	1.107	1	1.346	1.35	1.394	1.114	1.114
	XGB	1.182(0.005)	1.373(0.001)	1.202(0.001)	1.064(0.005)	1.084(0.002)	1.266(0.002)	1.202(0.002)	1.198(0.002)

[a]For algorithms with variability from random seeds, the standard deviation of scaled-RMSE from multiple runs of the final model is included in parentheses. The RMSE values are scaled by the minimum of median RMSE for each data set, i.e., the model with median scaled-RMSE equal to 1 is the best model for the corresponding data set.

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790

Table 8. Median R-Squared of Public Test Sets Predictions from Models Trained with Different ML Algorithm and Descriptor Combinations[a]

data set	method	Identity	mutInd	PCscores	protVec	sPairs	sScales	VHSE	zScales
Public-A	CNN	0.852(0.008)		0.89(0.009)				0.877(0.008)	0.898(0.017)
	GLMNET	0.871	0.868	0.863	0.86	0.843	0.825	0.876	0.884
	MLP	0.884(0.005)	0.888(0.01)	0.901(0.018)	0.797(0.018)	0.836(0.01)	0.867(0.027)	0.934(0.014)	0.897(0.01)
	RF	0.85(0.003)	0.848(0.003)	0.861(0.005)	0.667(0.02)	0.848(0.002)	0.892(0.003)	0.862(0.004)	0.872(0.006)
	SVM	0.867	0.854	0.876	0.763	0.846	0.817	0.883	0.797
	XGB	0.864(0.006)	0.871(0.004)	0.86(0.009)	0.689(0.021)	0.863(0.004)	0.872(0.004)	0.866(0.01)	0.868(0.003)
Public-B	CNN	0.709(0.081)		0.743(0.076)				0.579(0.095)	0.559(0.104)
	GLMNET	0.328	0.344	0.185	0.001	0.754	0.613	0.503	0.286
	MLP	0.366(0.134)	0.331(0.121)	0.261(0.174)	0.072(0.103)	0.38(0.09)	0.245(0.081)	0.358(0.13)	0.106(0.091)
	RF	0.454(0.042)	0.508(0.011)	0.257(0.039)	0.573(0.051)	0.637(0.01)	0.432(0.081)	0.195(0.029)	0.243(0.026)
	SVM	0.501	0.5	0.154	0.586	0.448	0.488	0.613	0.036
	XGB	0.651(0.065)	0.487(0.053)	0.494(0.054)	0.565(0.143)	0.646(0.043)	0.583(0.053)	0.451(0.032)	0.447(0.058)
Public-C	CNN	0.201(0.035)		0.209(0.045)				0.182(0.017)	0.159(0.027)
	GLMNET	0.279	0.223	0.176	0.075	0.138	0.129	0.15	0.074
	MLP	0.192(0.04)	0.216(0.005)	0.194(0.042)	0.112(0.01)	0.192(0.011)	0.171(0.053)	0.123(0.048)	0.127(0.01)
	RF	0.244(0.008)	0.234(0.007)	0.166(0.01)	0.155(0.01)	0.195(0.004)	0.285(0.014)	0.169(0.004)	0.185(0.005)
	SVM	0.249	0.241	0.236	0.21	0.272	0.098	0.191	0.168
	XGB	0.259(0.012)	0.245(0.005)	0.288(0.014)	0.21(0.01)	0.231(0.006)	0.295(0.01)	0.242(0.01)	0.262(0.037)
Public-D	CNN	0.412(0.015)		0.453(0.011)				0.448(0.014)	0.427(0.009)
	GLMNET	0.341	0.257	0	0.108	0.011	0.003	0.035	0.147
	MLP	0.376(0.006)	0.251(0.002)	0.328(0.015)	0.177(0.01)	0.301(0.005)	0.19(0.013)	0.404(0.008)	0.393(0.009)
	RF	0.406(0.002)	0.243(0.005)	0.411(0.002)	0.359(0.002)	0.474(0.002)	0.382(0.001)	0.414(0)	0.406(0.002)
	SVM	0.515	0.42	0.504	0.285	0.267	0.218	0.421	0.419
	XGB	0.397(0.004)	0.244(0.001)	0.381(0.001)	0.382(0.002)	0.461(0.002)	0.339(0.002)	0.376(0.002)	0.386(0.002)

[a]For algorithms with variability from random seeds, the standard deviation of R-squared from multiple runs of the final model is included in parentheses.

Table 9. Median Rank of Methods for Public Data Sets[a]

method	medianRank
CNN	11.5
XGB	15
SVM	20
RF	26
MLP	28
GLMNET	32

[a]The rank is calculated based on all descriptor/method combinations for each data set/metric.

Table 10. Median Rank of Methods and Descriptor for Public Data Sets[a]

descriptor	method	medianRank
PCscores	CNN	5.5
Identity	SVM	8
PCscores	SVM	10.5
VHSE	CNN	11.5
zScales	CNN	11.5
Identity	XGB	12
sScales	XGB	12.5
VHSE	SVM	13
sPairs	XGB	13.5
protVec	XGB	15.5

[a]Only the top ten are shown. The rank is calculated based on all descriptor/method combinations for each data set/metric.

As in the analysis for public data sets, for each data set and evaluation metric, the performance of 44 final models (i.e., descriptor/method combinations) was ranked according to the median value in increasing order for RMSE and decreasing order for R-Squared. The medians of performance metric and their ranks are provided in the Supporting Information.

The median rank by methods and by descriptor/method combination are shown in Table 14 and Table 15. Again, the CNN model achieved the best median rank (8.5) among methods, and the combination of the CNN model with the VHSE descriptor was the top performer with a median rank of 5.

Table 16 shows the average computational time in seconds of training the final model across proprietary data sets. For these large data sets, which are typical of the industrial scale, all model training processes were completed within 3 min, thereby demonstrating the efficiency of these methods for future application.

4.2.1. Classification Task and Evaluation. In most proprietary protein engineering tasks, the goal is to identify protein sequences that promote the highest substrate conversion. Therefore, machine learning models are required to distinguish between protein sequences that are likely to have high conversion and those that do not. It is desirable, therefore, to define a positive/negative label based on the true measured conversion and evaluate the predictive model performance in terms of a binary classification problem.

The test set predictions (the average across 10 repeated runs with different random seeds) from the best performing model in Table 15, the CNN model with the VHSE descriptor, were considered as an example. For each test set, proteins with outcome conversion higher than the 80^{th} percentile were defined as positive, while the rest were defined as negative. The top row in Figure 5 shows the scatter plot of prediction ($pred_{avg}$) vs true protein property (y_{test}) for each test set, with the positive samples highlighted in blue. A common phenomenon was noted across data sets: that test set proteins with higher true outcome properties were most likely to achieve more accurate predictions, while the spread of distribution of predictions for those with lower true values was wider. If a binary classifier was built based on the prediction, the false positive rate would be

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790

Table 11. Average Computational Time (in s) of Training One Final Model Across Public Data Sets and Repeated Runs (if Any)

	Identity	mutInd	PCscores	protVec	sPairs	sScales	VHSE	zScales
CNN	7.7		6.3				4.7	4.7
GLMNET	0.8	0.6	0.8	0.5	0.6	0.5	0.7	0.6
MLP	3.7	4.9	2	3.1	1.5	5.4	3.7	3.5
RF	2.4	0.8	4.4	0.5	2.2	3.0	7.9	2.2
SVM	0.9	0.6	0.9	0.5	0.6	0.5	0.7	0.6
XGB	2.5	2.0	2.5	2.4	2.0	1.3	2.5	1.9

expected to be high, while the false negative rate will be expected to be low. In practice, this means mutants with high conversions are very likely to be identified by the models, but the models will also identify many mutants that will not have high conversions. The bottom row in Figure 5 shows the corresponding ROC plots, which demonstrate the overall performance of a binary classifier built based on the predictions for each test set. Table 17 shows the AUROC when the cutoff defining the positive class was set at 80^{th}, 85^{th}, 90^{th}, and 95^{th} percentiles for all proprietary data sets. From the authors' experience analyzing noisy biological activity data sets, the range of AUROC achieved by the CNN+VHSE model is quite satisfactory.

4.3. Comparison of ML Algorithms. Figure 6 shows the boxplot comparison of methods with the results of rank aggregated across 8 data sets and 2 metrics; on the *x*-axis the methods are ordered by increasing median rank. Similarly, the two subplots in Figure 7 show the comparison of methods with rank calculated from each metric. Since the CNN model only takes four matrix format sequence-based descriptors (PCscores, zScales, VHSE, and Identity), in order to make a fair comparison, it was necessary to recalculate the ranks for the models using only these four descriptors (24 method/descriptor combinations in total). Ranking and width of boxplots in both figures demonstrate that CNN consistently achieved better results than all other methods. When comparing SVM and XGB, SVM has a wider variability across data sets and metrics, while the results from XGB are more consistent. Figure 8 shows the comparison of boxplots for all the ML methods and descriptors combinations, colored by different methods. Again, it is clear that both CNN and XGB achieve consistently better performance in combination with most descriptors, while the performance of SVM was strongly dependent on the descriptor choice.

4.4. Comparison of Protein Descriptors. Based on the numerical results shown in previous sections, it can be concluded that the choice of machine learning algorithm matters more than the choice of descriptors. However, the descriptor choice can provide (sometimes unexpected) nuance to the overall conclusions.

In some cases, simple descriptors, such as Identity or mutInd, can perform well. However, if the user is interested in predicting a new mutation that does not exist in the training set, the models built from the Identity or mutInd descriptor will ignore the new mutations during descriptor construction. In contrast, models using amino acid properties-based descriptors can extrapolate and thereby explore previously unseen mutants. For example, in terms of the amino acid identity, there is no connection between glutamate and aspartate, but using property descriptors glutamic acid and aspartic acid can be recognized as highly similar due to their anionic charge.

The sequence-based descriptors in Section 2.1.1 and the structure-based descriptor sPairs have another advantage over simple protein-level descriptors, in that, with a successful prediction model, the permutation-based variable importance metrics developed by Breiman for Random Forest[50] can be easily applied in order to identify the important mutation locations for providing data-driven insight for future experiments.

From Figure 8, it can be seen that a CNN model in combination with sequence-based descriptors constructed from amino acid properties (PCscores, zScales, and VHSE) performs slightly better than the Identity amino acid descriptor. With the numerical results and considering the advantages over the identity descriptors discussed above, the authors recommend using CNN models with sequence-based descriptors related to amino acid properties.

If there is difficulty in the implementation of CNN models (for example, requiring GPU) or to avoid the computational cost of CNN models, the secondary recommended model is the XGBoost model in combination with the sScales sequence-based descriptor, which also performs very well (see Figure 8). The novel sScales descriptor proposed herein is an amino acid property-based descriptor which permits extrapolation to previously unseen mutations and is a highly dense descriptor that uses a single number to summarize multiple properties for each amino acid. The small dimension of input vectors using the sScales descriptor significantly reduces the computational cost for the large proprietary data sets used here (Table 16).

5. DISCUSSION

This work has demonstrated that machine learning methods can be used as an effective tool to build prediction models for protein sequence/activity relationships. In addition, detailed procedures have been outlined for building machine learning models along with recommended methods and descriptors that serve as a practical reference for subsequent studies. For the machine learning methods, model architectures, hyperparameters, and protein sequence descriptors used herein, it was found that the CNN models with sequence-based amino acid property-related descriptors generally outperform other models across different protein types. However, the ranking of models depends on the given problem, data set, or model architecture. Although CNN models have a more complex model structure and many hyperparameters compared to other machine learning methods, the recommended model structures and hyperparameter sets that have been optimized for each data set (shown in the Supporting Information) may serve as a good starting point for scientists pursuing a machine learning approach to protein engineering.

The rationale for the success of the CNN models could be found in the fact that it encodes the ordering information on the entire protein sequence in the high-level features extracted, while other methods only make use of information on mutated positions.

That the embedding feature ProtVec performed unsatisfactorily in this study is notable, even though it had been shown to perform well in applications such as protein classification. However, protein redesign and protein classification are very different

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790

Table 12. Median Scaled-RMSE of Proprietary Test Sets Predictions from Models Trained with Different ML Algorithm and Descriptor Combinations[a]

data set	method	Identity	mutInd	PCscores	protVec	sPairs	sScales	VHSE	zScales
Proprietary-A	CNN	1.073(0.043)		1.081(0.072)				1.029(0.045)	1.147(0.058)
	GLMNET	1.219	1.229	1.154	1.482	1.312	4.387	1.242	1.211
	MLP	1.402(0.076)	1.068(0.065)	1.139(0.072)	1.69(0.327)	1.193(0.094)	1.764(0.343)	1.082(0.043)	1.296(0.058)
	RF	1.085(0.013)	1.074(0.013)	1.067(0.016)	1.728(0.011)	1.074(0.016)	1.083(0.018)	1.07(0.025)	1.074(0.022)
	SVM	1.13	1.409	1.01	1.243	1.434	1.354	1.111	1
	XGB	1.142(0.027)	1.059(0.02)	1.424(0.007)	1.992(0.038)	1.416(0.016)	1.096(0.011)	1.401(0.013)	1.327(0.027)
Proprietary-B	CNN	1.451(0.412)		1.548(0.598)				1.184(0.258)	1.318(0.458)
	GLMNET	1.669	1.457	1.678	1.336	2.019	2.39	1.933	1.4
	MLP	1(0.068)	2.754(0.311)	1.546(0.294)	2.992(0.576)	2.471(0.433)	2.462(0.365)	1.009(0.093)	1.35(0.265)
	RF	1.785(0.032)	1.885(0.036)	1.636(0.043)	3.798(0.054)	1.862(0.032)	1.677(0.039)	1.659(0.014)	1.692(0.018)
	SVM	1.302	3.31	4.362	3.912	7.565	8.828	2.031	7.799
	XGB	1.233(0.036)	1.056(0.025)	1.541(0.047)	4.085(0.075)	1.062(0.032)	1.212(0.029)	2.117(0.118)	1.617(0.036)
Proprietary-C	CNN	1.334(0.035)		1.28(0.081)				1.322(0.053)	1.255(0.129)
	GLMNET	1.336	1.567	1.344	1.801	1.575	1.543	1.619	1.347
	MLP	1.576(0.118)	1.822(0.077)	1.254(0.032)	1.239(0.037)	1.438(0.052)	1.704(0.387)	1.375(0.044)	1.335(0.05)
	RF	1.526(0.005)	1.684(0.011)	1.431(0.011)	1.457(0.011)	1.344(0.01)	1.537(0.006)	1.47(0.005)	1.431(0.018)
	SVM	1.108	1.11	1.015	1.175	1.321	1.56	1.125	1
	XGB	1.41(0.027)	1.486(0.013)	1.488(0.027)	1.476(0.016)	1.298(0.013)	1.368(0.011)	1.363(0.016)	1.293(0.01)
Proprietary-D	CNN	1(0.052)		1.055(0.042)				1.01(0.04)	1.016(0.03)
	GLMNET	1.094	1.268	1.08	1.338	1.265	4.254	1.086	1.155
	MLP	1.17(0.056)	1.238(0.075)	1.212(0.129)	1.351(0.097)	1.153(0.032)	1.62(0.209)	1.026(0.029)	1.081(0.04)
	RF	1.052(0.007)	1.072(0.005)	1.045(0.004)	1.171(0.01)	1.029(0.002)	1.061(0.005)	1.046(0.002)	1.048(0.004)
	SVM	1.415	2.102	1.593	1.917	2.569	1.069	1.567	1.971
	XGB	1.036(0.01)	1.057(0.005)	1.047(0.014)	1.202(0.012)	1.086(0.009)	1.037(0.005)	1.058(0.009)	1.071(0.016)

[a]For algorithms with variability from random seeds, the standard deviation of scaled-RMSE from multiple runs of the final model is included in parentheses. The RMSE values are scaled by the minimum of median RMSE for each data set, i.e., the model with median scaled-RMSE equal to 1 is the best model for the corresponding data set.

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790

Table 13. Median R-Squared of Proprietary Test Sets Predictions from Models Trained with Different ML Algorithm and Descriptor Combinations[a]

data set	method	Identity	mutInd	PCscores	protVec	sPairs	sScales	VHSE	zScales
Proprietary-A	CNN	0.119(0.045)		0.122(0.04)				0.163(0.027)	0.095(0.035)
	GLMNET	0.187	0.114	0.183	0.055	0.138	0	0.16	0.129
	MLP	0.069(0.033)	0.09(0.057)	0.188(0.046)	0.25(0.05)	0.109(0.032)	0.01(0.018)	0.088(0.041)	0.15(0.035)
	RF	0.071(0.011)	0.058(0.012)	0.08(0.02)	0.075(0.003)	0.095(0.014)	0.049(0.023)	0.075(0.023)	0.077(0.022)
	SVM	0.099	0.284	0.34	0.177	0.196	0.009	0.26	0.201
	XGB	0.092(0.005)	0.112(0.016)	0.048(0.003)	0.11(0.008)	0.057(0.004)	0.084(0.008)	0.052(0.003)	0.061(0.003)
Proprietary-B	CNN	0.251(0.04)		0.171(0.032)				0.175(0.038)	0.132(0.026)
	GLMNET	0.093	0.172	0.01	0.04	0.008	0.018	0	0.023
	MLP	0.092(0.046)	0.027(0.032)	0.018(0.041)	0.009(0.004)	0.001(0.002)	0.007(0.009)	0.051(0.04)	0.006(0.016)
	RF	0.005(0.002)	0.023(0.004)	0.003(0.002)	0(0)	0.011(0.002)	0.003(0.001)	0(0)	0(0)
	SVM	0.182	0.035	0.008	0.006	0.002	0	0.047	0.001
	XGB	0.136(0.025)	0.127(0.011)	0.074(0.01)	0(0)	0.077(0.019)	0.044(0.011)	0.071(0.017)	0.095(0.014)
Proprietary-C	CNN	0.35(0.028)		0.41(0.05)				0.368(0.035)	0.425(0.081)
	GLMNET	0.328	0.4	0.338	0.135	0.013	0.049	0.013	0.273
	MLP	0.317(0.042)	0.381(0.015)	0.479(0.034)	0.346(0.041)	0.362(0.018)	0.208(0.061)	0.534(0.034)	0.536(0.027)
	RF	0.358(0.007)	0.304(0.014)	0.347(0.013)	0.167(0.018)	0.333(0.014)	0.394(0.007)	0.407(0.006)	0.339(0.018)
	SVM	0.506	0.51	0.579	0.424	0.359	0.22	0.523	0.612
	XGB	0.383(0.012)	0.347(0.007)	0.428(0.015)	0.153(0.016)	0.286(0.014)	0.397(0.011)	0.423(0.007)	0.386(0.009)
Proprietary-D	CNN	0.205(0.077)		0.055(0.021)				0.088(0.058)	0.089(0.033)
	GLMNET	0.037	0	0.07	0.005	0.001	0.011	0.03	0.027
	MLP	0.135(0.034)	0.08(0.034)	0.025(0.016)	0.106(0.028)	0.032(0.02)	0.038(0.012)	0.048(0.038)	0.06(0.024)
	RF	0.001(0.004)	0(0)	0.005(0.002)	0.028(0.003)	0.037(0.002)	0.002(0.002)	0.004(0.002)	0.003(0.002)
	SVM	0.181	0.153	0.028	0.076	0.044	0.045	0.033	0.034
	XGB	0.076(0.011)	0.01(0.003)	0.054(0.015)	0.048(0.006)	0.069(0.01)	0.059(0.009)	0.026(0.009)	0.033(0.011)

[a]For algorithms with variability from random seeds, the standard deviation of R-squared from multiple runs of the final model is included in parentheses.

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790

Table 14. Median Rank of Methods for Proprietary Data Sets[a]

method	medianRank
CNN	10
XGB	18.5
SVM	19.5
MLP	23
RF	27.5
GLMNET	27.5

[a]The rank is calculated based on all descriptor/method combinations for each data set/metric.

Table 15. Median Rank of Methods and Descriptor for Proprietary Data Sets[a]

descriptor	method	medianRank
VHSE	CNN	6
Identity	SVM	7.5
zScales	CNN	9
Identity	CNN	10.5
mutInd	SVM	12.5
PCscores	CNN	13
Identity	XGB	14
VHSE	MLP	14
zScales	MLP	14.5
zScales	SVM	15.5

[a]Only the top ten are shown. The rank is calculated based on all descriptor/method combinations for each data set/metric.

Table 17. AUROC of the CNN+VHSE Model for Proprietary Data Sets

	quantile cutoff			
data set	0.8	0.85	0.9	0.95
Proprietary-A	0.665	0.698	0.689	0.680
Proprietary-B	0.726	0.705	0.665	0.638
Proprietary-C	0.797	0.770	0.742	0.668
Proprietary-D	0.779	0.820	0.848	0.884

tasks. In a protein family classification task, the proteins have a very diverse sequence pattern and can be of different lengths. However, in a protein engineering task, small differences between highly similar variants derived from the same parent are compared. The highly dense 100-dimensional ProtVec feature vector may not be able to provide a sufficient signal to differentiate these mutants. In addition, it does not contain positional information, which may have contributed to the poor performance compared to other descriptors.

When engineering proteins, it is desirable to identify variants with optimal properties containing as few mutations as possible and in as few evolution steps as possible. Usually, the first few rounds of an evolution experiment consist of variants that are very close to the starting (often wild-type) sequence, that is, the sequence design exploration space is highly empirical and limited. Such a limited data set makes ML model construction

Table 16. Average Computational Time (in s) of Training One Final Model Across Proprietary Data Sets and Repeated Runs (if Any)

	Identity	mutInd	PCscores	protVec	sPairs	sScales	VHSE	zScales
CNN	163.8		99.2				122.4	146.9
GLMNET	1.7	2.0	9.1	0.5	5.9	0.5	9.7	5.8
MLP	60.9	30.4	49.5	26.6	34.7	32.3	67.4	32.2
RF	69.6	4.0	19.7	16.7	16.2	7.0	53.8	8.4
SVM	11.1	3.2	36.1	2.6	16.7	5.1	26.9	17.3
XGB	60.5	10.9	23.3	9.1	9.5	6.1	20.1	12.9

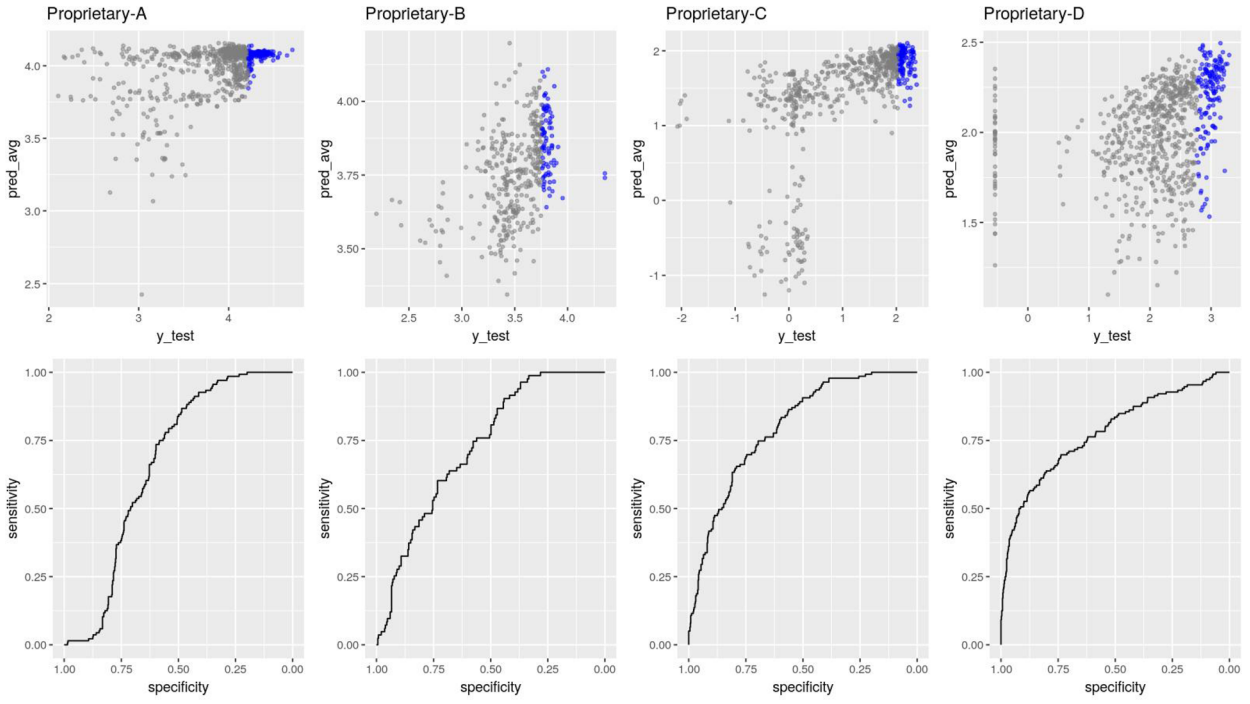


Figure 5. Scatter plot (top row) of test set prediction from CNN+VHSE and ROC curve (bottom row) with positive samples threshold at 0.8. The blue points in scatter plots represent positive test set data.

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790

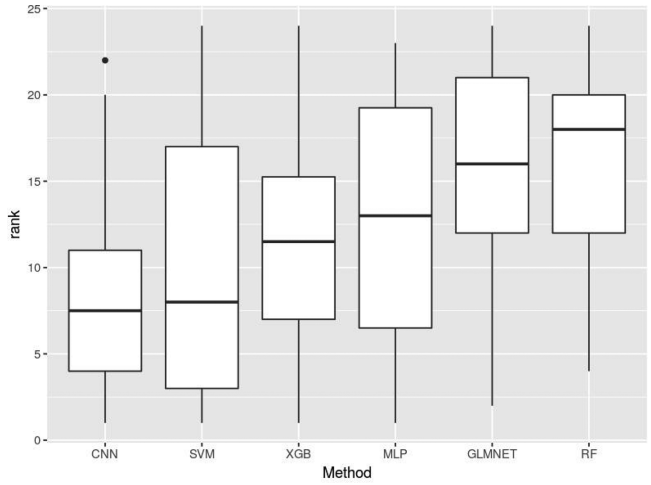


Figure 6. Boxplot comparison of rank by ML method. For each data set and evaluation metric, the rankings are calculated for six ML methods and four descriptors that are used by all ML algorithms (PCscores, zScales, VHSE, and Identity).

more challenging. The models are improved as the evolution progresses to include more rounds of data (that again consists of empirically identified mutations). This process risks being trapped in local optima. To speed up a workflow or explore a larger design space, it is recommended to generate a large number of protein sequences in silico, considering as many combinations as possible. For example, if one is interested in exploring all 20 amino acids at N locations in the parent sequence, the total number of possible mutants would be 20^N. As N increases, the feasibility of experimentally testing all of these variants diminishes. Thus, in silico methods can aid in this process, since the designed sequences can be explored, ranked, and selected computationally for further experimental evaluation. In order to enable machine learning as a routine part of a protein engineering workflow, it may be preferable to experimentally target large numbers of mutations in the earliest rounds (for example, using site saturation mutagenesis across all sequence positions) to generate the largest possible data set for model training. For example, in contrast to previous studies with fewer than ten mutations in the training set, the proprietary data sets in this work include mutants with more sequence variability (Figures 3 and 4). This has permitted construction of more information rich models, which in turn has enabled exploration of a larger in silico design space and thus larger jumps in the evolution. This can drive the directed evolution process in a faster and more informed way.

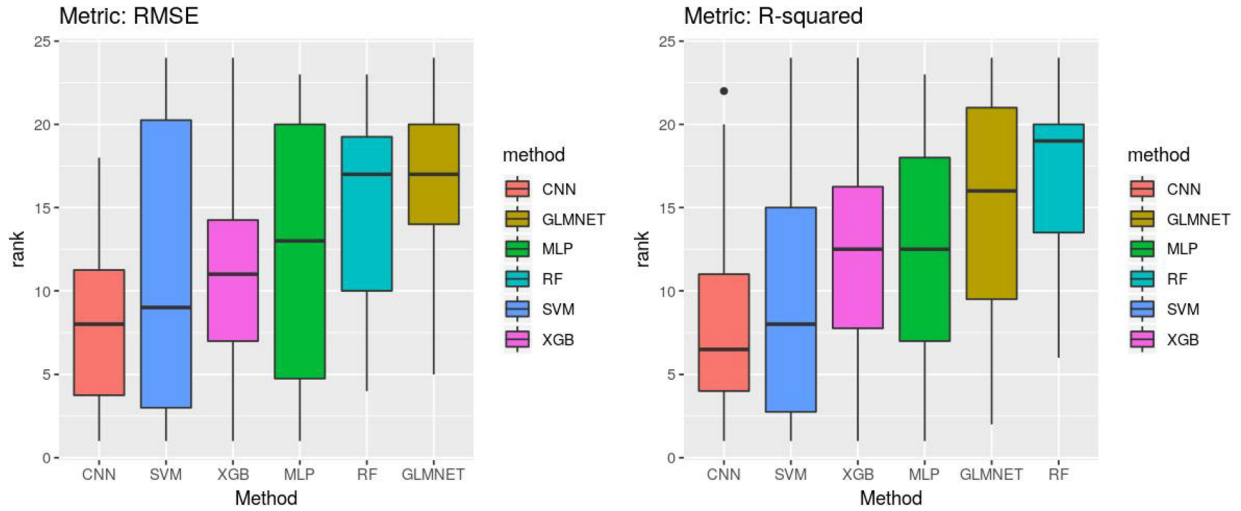


Figure 7. Left: Boxplot comparison of rank by RMSE metric. Right: Boxplot comparison of rank calculate by R-squared metric. For each data set and evaluation metric, the rankings are calculated for six ML methods and four descriptors that are used by all ML algorithms (PCscores, zScales, VHSE, and Identity).

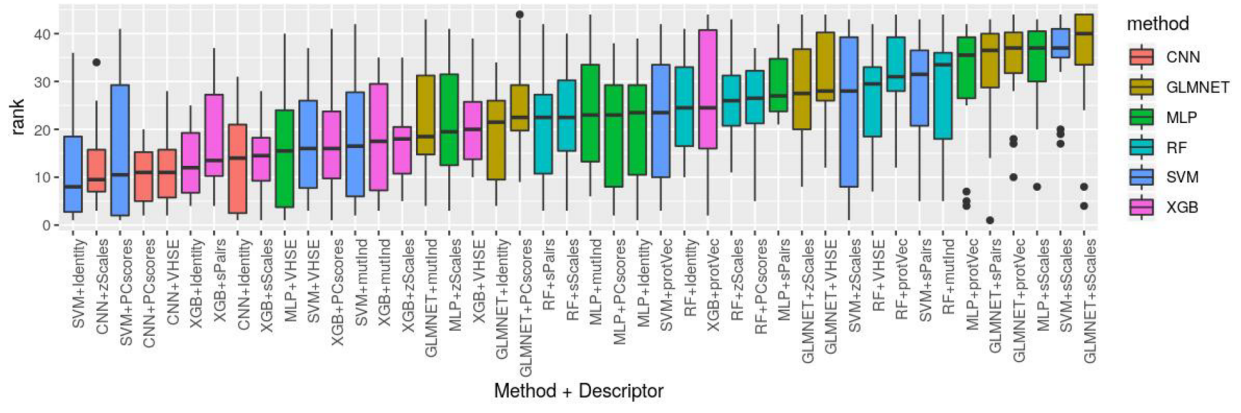


Figure 8. Boxplot comparison of rank by model (ML method + descriptor combination). For each data set and evaluation metric, the rankings are calculated for all 44 descriptor/method combinations.

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790

One potential extension of this work would be investigating whether the high-level descriptors extracted from the last layer of trained CNN models could be helpful for constructing protein descriptors for other tasks (known as transfer learning[75]). Another extension may be to combine data sets of similar proteins with different measurable variables to determine whether multitask machine learning models could improve the overall predictive performance. Although a wide selection of protein sequence descriptors in combination with state-of-the-art machine learning methods have been explored, there are still many options which may further improve the model performance for future studies. For example, the "1cycle learning rate policy" is a novel neural network training strategy that aims to reduce training time and improve performance.[76] Various optimization algorithms, regularization techniques, and model structures that have succeeded in other deep learning applications are also worth further investigation. Additional protein descriptors may also be explored, including using entity embeddings of categorical variables,[77] to create descriptors for amino acids.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jcim.0c00073.

Table S1, optimal hyperparameter values for all choices of data set/ML algorithm/descriptor; Table S2, medians of all performance metrics and rank for public data set; and Table S3, medians of all performance metrics and rank for proprietary data set (PDF)

■ AUTHOR INFORMATION

Corresponding Authors

Jennifer M. Johnston − *Computational and Structural Chemistry, Merck & Co., Inc., Kenilworth, New Jersey 07033, United States;* orcid.org/0000-0003-3582-9567; Email: jennifer.johnston@merck.com

Yuting Xu − *Biometrics Research, Merck & Co., Inc., Rahway, New Jersey 07065, United States;* orcid.org/0000-0003-2091-3854; Email: yuting.xu@merck.com

Authors

Deeptak Verma − *Computational and Structural Chemistry, Merck & Co., Inc., Kenilworth, New Jersey 07033, United States*

Robert P. Sheridan − *Computational and Structural Chemistry, Merck & Co., Inc., Kenilworth, New Jersey 07033, United States;* orcid.org/0000-0002-6549-1635

Andy Liaw − *Biometrics Research, Merck & Co., Inc., Rahway, New Jersey 07065, United States*

Junshui Ma − *Early Oncology Statistics, Merck & Co., Inc., Rahway, New Jersey 07065, United States*

Nicholas M. Marshall − *Invenra, Inc., Madison, Wisconsin 53719, United States*

John McIntosh − *Process Research & Development, Merck & Co., Inc., Rahway, New Jersey 07065, United States*

Edward C. Sherer − *Computational and Structural Chemistry, Merck & Co., Inc., Kenilworth, New Jersey 07033, United States;* orcid.org/0000-0001-8178-9186

Vladimir Svetnik − *Biometrics Research, Merck & Co., Inc., Rahway, New Jersey 07065, United States*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.0c00073

Author Contributions

Y.X. and D.V. contributed equally to this work. Y.X. contributed on machine learning methodology and implementation; D.V. contributed on proposing novel descriptors and computational protein design.

Notes

The authors declare the following competing financial interest(s): At the time this work was completed, all authors were employed by Merck Sharp & Dohme Corp., a subsidiary of Merck & Co., Inc., Kenilworth, NJ, USA.

Supporting research data regarding web-page reference of example 2D AA-index (statistical contact potential derived from 25 X-ray protein structures) for this article may be accessed at https://www.genome.jp/dbget-bin/www_bget?aaindex:TANS760101.

■ ACKNOWLEDGMENTS

The authors wish to thank the Protein Engineering department of MRL for support and experimental effort in this work and particularly Andrew Roberts from the High Performance Computing team for support in database management.

■ ABBREVIATIONS

CNN, Convolutional Neural Networks; GLMNET, Lasso and Elastic-Net Regularized Generalized Linear Models; MLP, multilayer perceptron; RF, Random Forests; SVM, Support Vector Machine; XGB, Extreme Gradient Boosting

■ REFERENCES

(1) Cobb, R. E.; Chao, R.; Zhao, H. Directed Evolution: Past, Present, and Future. *AIChE J.* **2013**, *59*, 1432−1440.

(2) Jäckel, C.; Kast, P.; Hilvert, D. Protein Design by Directed Evolution. *Annu. Rev. Biophys.* **2008**, *37*, 153−173.

(3) Wang, J.; Cao, H.; Zhang, J. Z.; Qi, Y. Computational Protein Design with Deep Learning Neural Networks. *Sci. Rep.* **2018**, *8*, 6349.

(4) Yang, K. K.; Wu, Z.; Arnold, F. H. Machine Learning in Protein Engineering. 2018,*arXiv preprint arXiv:1811.10775*.

(5) Wu, Z.; Kan, S. 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*, 8852−8858.

(6) Ismail, H. D.; Saigo, H.; KC, D. B. RF-NR: Random Forest based Approach for Improved Classification of Nuclear Receptors. *IEEE/ACM Trans. Comput. Biol. Bioinf.* **2018**, *15*, 1844−1852.

(7) Hou, J.; Adhikari, B.; Cheng, J. DeepSF: Deep Convolutional Neural Network for Mapping Protein Sequences to Folds. *Bioinformatics* **2018**, *34*, 1295−1303.

(8) Zacharaki, E. I. Prediction of Protein Function using a Deep Convolutional Neural Network Ensemble. *PeerJ. Computer Science* **2017**, *3*, No. e124.

(9) White, C.; Ismail, H. D.; Saigo, H. CNN-BLPred: A Convolutional Neural Network based Predictor for β-lactamases (BL) and their Classes. *BMC Bioinf.* **2017**, *18*, 577.

(10) Ragoza, M.; Hochuli, J.; Idrobo, E.; Sunseri, J.; Koes, D. R. Protein−Ligand Scoring with Convolutional Neural Networks. *J. Chem. Inf. Model.* **2017**, *57*, 942−957.

(11) Yang, K. K.; Wu, Z.; Bedbrook, C. N.; Arnold, F. H. Learned Protein Embeddings for Machine Learning. *Bioinformatics* **2018**, *34*, 2642−2648.

(12) Romero, P. A.; Krause, A.; Arnold, F. H. Navigating the Protein Fitness Landscape with Gaussian Processes. *Proc. Natl. Acad. Sci. U. S. A.* **2013**, *110*, E193−E201.

(13) Yang, K. K.; Wu, Z.; Arnold, F. H. Machine-Learning-Guided Directed Evolution for Protein Engineering. *Nat. Methods* **2019**, *16*, 687.

(14) Mason, D. M.; Friedensohn, S.; Weber, C. R.; Jordi, C.; Wagner, B.; Meng, S.; Gainza, P.; Correia, B.; Reddy, S. T. Deep Learning Enables Therapeutic Antibody Optimization in Mammalian Cells by

Deciphering High-dimensional Protein Sequence Space. *bioRxiv* **2019**, 617860.
(15) Han, X.; Ning, W.; Ma, X.; Wang, X.; Zhou, K. Improve Protein Solubility and Activity based on Machine Learning Models. *bioRxiv* **2019**, 817890.
(16) Kimothi, D.; Soni, A.; Biyani, P.; Hogan, J. M. Distributed Representations for Biological Sequence Analysis. 2016, *arXiv preprint arXiv:1608.05949*, https://arxiv.org/abs/1608.05949 (accessed 2020-04-10).
(17) Le, Q.; Mikolov, T. Distributed Representations of Sentences and Documents. *International conference on machine learning* **2014**, 1188−1196.
(18) Xiao, N.; Cao, D. S.; Zhu, M. F.; Xu, Q. S. Protr/ProtrWeb: R package and Web Server for Generating Various Numerical Representation Schemes of Protein Sequences. *Bioinformatics* **2015**, *31*, 1857−1859.
(19) Kawashima, S.; Kanehisa, M. AAindex: Amino Acid Index Database. *Nucleic acids research* **2000**, *28*, 374.
(20) Osorio, D.; Rondón-Villarrea, P.; Torres, R. Peptides: A Package for Data Mining of Antimicrobial Peptides. *R Journal* **2015**, *7*, 4.
(21) 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*, D202−D205.
(22) Asgari, E.; Mofrad, M. R. Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics. *PLoS One* **2015**, *10*, No. e0141287.
(23) Sandberg, M.; Eriksson, L.; Jonsson, J.; Sjöström, M.; Wold, S. New Chemical Descriptors Relevant for the Design of Biologically Active Peptides. A Multivariate Characterization of 87 Amino Acids. *J. Med. Chem.* **1998**, *41*, 2481−2491.
(24) Mei, H.; Liao, Z. H.; Zhou, Y.; Li, S. Z. A New Set of Amino Acid Descriptors and its Application in Peptide QSARs. *Biopolymers* **2005**, *80*, 775−786.
(25) Kolinski, A.; Pokarowska, M.; Kanehisa, M.; Pokarowski, P.; Katayama, T.; Kawashima, S. AAindex: Amino Acid Index Database, Progress Report 2008. *Nucleic Acids Res.* **2007**, *36*, D202−D205.
(26) Biou, V.; Gibrat, J.; Levin, J.; Robson, B.; Garnier, J. Secondary Structure Prediction: Combination of three different Methods. *Protein Eng., Des. Sel.* **1988**, *2*, 185−191.
(27) Blaber, M.; Zhang, X. J.; Matthews, B. W. Structural Basis of Amino Acid Alpha Helix Propensity. *Science* **1993**, *260*, 1637−1640.
(28) Nagano, K. Logical Analysis of the Mechanism of Protein Folding: I. Predictions of Helices, Loops and β-Structures from Primary Structure. *J. Mol. Biol.* **1973**, *75*, 401−420.
(29) Tsai, J.; Gerstein, M. Calculations of Protein Volumes: Sensitivity Analysis and Parameter Database. *Bioinformatics* **2002**, *18*, 985−995.
(30) Nakashima, H.; Nishikawa, K. The Amino Acid Composition is Different between the Cytoplasmic and Extracellular Sides in Membrane Proteins. *FEBS Lett.* **1992**, *303*, 141−146.
(31) Cedano, J.; Aloy, P.; Perez-Pons, J. A.; Querol, E. Relation between Amino Acid Composition and Cellular Location of Proteins. *J. Mol. Biol.* **1997**, *266*, 594−600.
(32) Lifson, S.; Sander, C. Antiparallel and Parallel β-Strands Differ in Amino Acid Residue Preferences. *Nature* **1979**, *282*, 109.
(33) Miyazawa, S.; Jernigan, R. L. Self-consistent Estimation of Inter-residue Protein Contact Energies based on an Equilibrium Mixture Approximation of Residues. *Proteins: Struct., Funct., Genet.* **1999**, *34*, 49−68.
(34) Argos, P.; Rao, J. M.; Hargrave, P. A. Structural Prediction of Membrane-Bound Proteins. *Eur. J. Biochem.* **1982**, *128*, 565−575.
(35) Smith, R.; Dawson, J. R.; Tanford, C. The Size and Number of Polypeptide Chains in Human Serum Low Density Lipoprotein. *J. Biol. Chem.* **1972**, *247*, 3376−3381.
(36) FAUCHÈRE, J.-L.; Charton, M.; Kier, L. B.; Verloop, A.; Pliska, V. Amino Acid Side Chain Parameters for Correlation Studies in Biology and Pharmacology. *Int. J. Pept. Protein Res.* **1988**, *32*, 269−278.
(37) Tanaka, S.; Scheraga, H. A. Medium-and Long-range Interaction Parameters between Amino Acids for Predicting Three-Dimensional Structures of Proteins. *Macromolecules* **1976**, *9*, 945−950.
(38) Berman, H.; Henrick, K.; Nakamura, H. Announcing the Worldwide Protein Data Bank. *Nat. Struct. Mol. Biol.* **2003**, *10*, 980.
(39) Olsson, M. H.; Søndergaard, C. R.; Rostkowski, M.; Jensen, J. H. PROPKA3: Consistent Treatment of Internal and Surface Residues in Empirical p K a Predictions. *J. Chem. Theory Comput.* **2011**, *7*, 525−537.
(40) Release, S. *Schrödinger Release 2019-4, Maestro*; Schrödinger, LLC: New York, NY, 2019.
(41) Lee, T. K.; Nguyen, T. *Protein Family Classification with Neural Networks*; 2016.
(42) Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. 2013, *arXiv preprint arXiv:1301.3781*.https://arxiv.org/abs/1301.3781 (accessed 2020-04-10).
(43) Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; Dean, J. Distributed Representations of Words and Phrases and their Compositionality. *Advances in neural information processing systems* **2013**, 3111−3119.
(44) Fawcett, T. An Introduction to ROC Analysis. *Pattern recognition letters* **2006**, *27*, 861−874.
(45) Friedman, J.; Hastie, T.; Tibshirani, R. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software* **2010**, *33*, 1−22.
(46) Drucker, H.; Burges, C. J.; Kaufman, L.; Smola, A. J.; Vapnik, V. Support Vector Regression Machines. *Advances in neural information processing systems* **1997**, 155−161.
(47) Doucet, J.-P.; Barbault, F.; Xia, H.; Panaye, A.; Fan, B. Nonlinear SVM Approaches to QSPR/QSAR Studies and Drug Design. *Curr. Comput.-Aided Drug Des.* **2007**, *3*, 263−289.
(48) Liang, Y.; Xu, Q.-S.; Li, H.-D.; Cao, D.-S. *Support Vector Machines and their Application in Chemistry and Biotechnology*; CRC Press: 2016.
(49) Yao, X.; Panaye, A.; Doucet, J.-P.; Zhang, R.; Chen, H.; Liu, M.; Hu, Z.; Fan, B. T. Comparative study of QSAR/QSPR Correlations using Support Vector Machines, Radial Basis Function Neural Networks, and Multiple Linear Regression. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 1257−1266.
(50) Breiman, L. Random Forests. *Machine learning* **2001**, *45*, 5−32.
(51) Svetnik, V.; Liaw, A.; Tong, C.; Culberson, J. C.; Sheridan, R. P.; Feuston, B. P. Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 1947−1958.
(52) Friedman, J. H. Stochastic Gradient Boosting. *Computational statistics & data analysis* **2002**, *38*, 367−378.
(53) Svetnik, V.; Wang, T.; Tong, C.; Liaw, A.; Sheridan, R. P.; Song, Q. Boosting: An Ensemble Learning Tool for Compound Classification and QSAR Modeling. *J. Chem. Inf. Model.* **2005**, *45*, 786−799.
(54) Chen, T.; Guestrin, C. Xgboost: A Scalable Tree Boosting System. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*; 2016; pp 785−794, DOI: 10.1145/2939672.2939785.
(55) Rawat, W.; Wang, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural computation* **2017**, *29*, 2352−2449.
(56) Alom, M. Z.; Taha, T. M.; Yakopcic, C.; Westberg, S.; Sidike, P.; Nasrin, M. S.; Van Esesn, B. C.; Awwal, A. A. S.; Asari, V. K. The history began from Alexnet: A Comprehensive Survey on Deep Learning Approaches. 2018, *arXiv preprint arXiv:1803.01164*. https://arxiv.org/ftp/arxiv/papers/1803/1803.01164.pdf (accessed 2020-04-10).
(57) LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278−2324.
(58) Krizhevsky, A.; Sutskever, I.; Hinton, G. E. Imagenet Classification with Deep Convolutional Neural Networks. *Advances in neural information processing systems* **2012**, 1097−1105.
(59) Cireşan, D.; Meier, U.; Schmidhuber, J. Multi-column Deep Neural Networks for Image Classification. 2012, *arXiv preprint arXiv:1202.2745*. https://arxiv.org/abs/1202.2745 (accessed 2020-04-10).
(60) Bai, S.; Kolter, J. Z.; Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence

Modeling. 2018, *arXiv preprint arXiv:1803.01271*. https://arxiv.org/abs/1803.01271 (accessed 2020-04-10).
(61) Kalchbrenner, N.; Grefenstette, E.; Blunsom, P. A Convolutional Neural Network for Modelling Sentences. 2014, *arXiv preprint arXiv:1404.2188*. https://arxiv.org/abs/1404.2188 (accessed 2020-04-10).
(62) Zeng, H.; Edwards, M. D.; Liu, G.; Gifford, D. K. Convolutional Neural Network Architectures for Predicting DNA−Protein Binding. *Bioinformatics* **2016**, *32*, i121−i127.
(63) LeCun, Y.; Bengio, Y. Convolutional Networks for Images, Speech, and Time Series. *handbook of brain theory and neural networks* **1995**, *3361*, 1995.
(64) Dieleman, S.; Schlter, J.; Raffel, C.; Olson, E.; Snderby, S. K.; Nouri, D.; Maturana, D.; Thoma, M.; Battenberg, E.; Kelly, J. *Lasagne: First Release*; 2015; http://dx.doi.org/10.5281/zenodo.27878 (accessed 2020-04-10).
(65) Theano Development Team, *Theano: A Python Framework for Fast Computation of Mathematical Expressions*. 2016, *arXiv e-prints abs/1605.02688*. https://arxiv.org/abs/1605.02688 (accessed 2020-04-10).
(66) Chen, T.; Li, M.; Li, Y.; Lin, M.; Wang, N.; Wang, M.; Xiao, T.; Xu, B.; Zhang, C.; Zhang, Z. Mxnet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. 2015, *arXiv preprint arXiv:1512.01274*. https://arxiv.org/abs/1512.01274 (accessed 2020-04-10).
(67) Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **2011**, *12*, 2825−2830.
(68) Karatzoglou, A.; Smola, A.; Hornik, K.; Zeileis, A. kernlab − An S4 Package for Kernel Methods in R. *Journal of Statistical Software* **2004**, *11*, 1−20.
(69) Chen, T.; He, T.; Benesty, M.; Khotilovich, V.; Tang, Y.; Cho, H.; Chen, K.; Mitchell, R.; Cano, I.; Zhou, T.; Li, M.; Xie, J.; Lin, M.; Geng, Y.; Li, Y. *xgboost: Extreme Gradient Boosting*, R package version 0.82.1; 2019.
(70) Kuhn, M. *caret: Classification and Regression Training*, R package version 6.0-84; 2019.
(71) Engqvist, M. K.; McIsaac, R. S.; Dollinger, P.; Flytzanis, N. C.; Abrams, M.; Schor, S.; Arnold, F. H. Directed Evolution of Gloeobacter Violaceus Rhodopsin Spectral Properties. *J. Mol. Biol.* **2015**, *427*, 205−220.
(72) 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*, 1060−1066.
(73) Sarkisyan, K. S.; Bolotin, D. A.; Meer, M. V.; Usmanova, D. R.; Mishin, A. S.; Sharonov, G. V.; Ivankov, D. N.; Bozhanova, N. G.; Baranov, M. S.; Soylemez, O.; Bogatyreva, N. S.; Vlasov, P. K.; Egorov, E. S.; Logacheva, M. D.; Kondrashov, A. S.; Chudakov, D. M.; Putintseva, E. V.; Mamedov, I. Z.; Tawfik, D. S.; Lukyanov, K. A.; Kondrashov, F. A. Local Fitness Landscape of the Green Fluorescent Protein. *Nature* **2016**, *533*, 397.
(74) Sheridan, R. P. Time-Split Cross-Validation as a Method for Estimating the Goodness of Prospective Prediction. *J. Chem. Inf. Model.* **2013**, *53*, 783−790.
(75) Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How Transferable are Features in Deep Neural Networks? *Advances in neural information processing systems*; 2014; pp 3320−3328.
(76) Smith, L. N. A Disciplined Approach to Neural Network Hyper-parameters: Part 1−Learning Rate, Batch Size, Momentum, and Weight Decay. 2018, *arXiv preprint arXiv:1803.09820*. https://arxiv.org/abs/1803.09820 (accessed 2020-04-10).
(77) Guo, C.; Berkhahn, F. Entity Embeddings of Categorical Variables. 2016, *arXiv preprint arXiv:1604.06737*. https://arxiv.org/abs/1604.06737 (accessed 2020-04-10).

https://dx.doi.org/10.1021/acs.jcim.0c00073
J. Chem. Inf. Model. 2020, 60, 2773−2790