The NK Landscape as a Versatile Benchmark for Machine Learning Driven Protein Engineering

Adam C. Mater[1‡], Mahakaran Sandhu[1‡], and Colin Jackson[1*]

[1]Research School of Chemistry, The Australian National University, Canberra, Australia

[*]Corresponding author: colin.jackson@anu.edu.au

Abstract

Machine learning (ML) has the potential to revolutionize protein engineering. However, the field currently lacks standardized and rigorous evaluation benchmarks for sequence-fitness prediction, which makes accurate evaluation of the performance of different architectures difficult. Here we propose a unifying framework for ML-driven sequence-fitness prediction. Using simulated (the NK model) and empirical sequence landscapes, we define four key performance metrics: interpolation within the training domain, extrapolation outside the training domain, robustness to sparse training data, and ability to cope with epistasis/ruggedness. We show that architectural differences between algorithms consistently affect performance against these metrics across both experimental and theoretical landscapes. Moreover, landscape ruggedness is revealed to be the greatest determinant of the accuracy of sequence-fitness prediction. We hope that this benchmarking method and the code that accompanies it will enable robust evaluation and comparison of novel architectures in this emerging field and assist in the adoption of ML for protein engineering.

‡ These authors contributed equally.

Introduction

Machine learning has transformed many domains of science and is poised to revolutionize protein science and engineering. The fundamental problem in protein engineering is the accurate prediction of protein activity (i.e. fitness) from sequence information. Successfully addressing this core problem promises to transform the field, leading to better proteins for industry and medicine at a fraction of the cost. A number of ML methods have been implemented to address this[1], including Gaussian process regression[2–5], unsupervised statistical analyses[6], deep neural networks and sequence models[7–11]. However, a uniformly used set of objectives and benchmarks against which each architecture can be evaluated is currently unavailable.

As cutting-edge ML models are adapted to protein sequence-fitness prediction in the coming years, it is critical that they be evaluated against a standard framework. Developing this standard framework requires careful consideration of the fundamental structure on which sequence-fitness relationships exist: the *fitness landscape*. The fitness landscape is a well-known idea in biology, first introduced in the seminal work of Sewall Wright[12], and applied to proteins by John Maynard Smith[13]. The protein fitness landscape rests on the notion of a *combinatorial sequence space*, an enormous space in which all possible sequences of a given length N exist. The neighbors of some sequence X in this space are all sequences one mutation away (i.e. Hamming distance[14] of 1), and form what we call the first *mutational regime*. Likewise, sequences that are 2 mutations away from X form the second mutational regime, and so on (Figure 1 A). Connecting all sequences to their 1-mutation neighbors produces a sequence graph (Figure 1 B). A fitness function maps each sequence X in the combinatorial sequence space to a numerical fitness value, F, (which is typically measured as some biophysical property such as thermostability, fold enrichment, or fluorescence) forming a fitness graph (Figure 1 C). The fitness graph can also be construed as the *protein fitness landscape* by embedding it in a Euclidean space (Figure 1 D).

A key characteristic of fitness landscapes is their *ruggedness*. When the fitnesses of adjacent

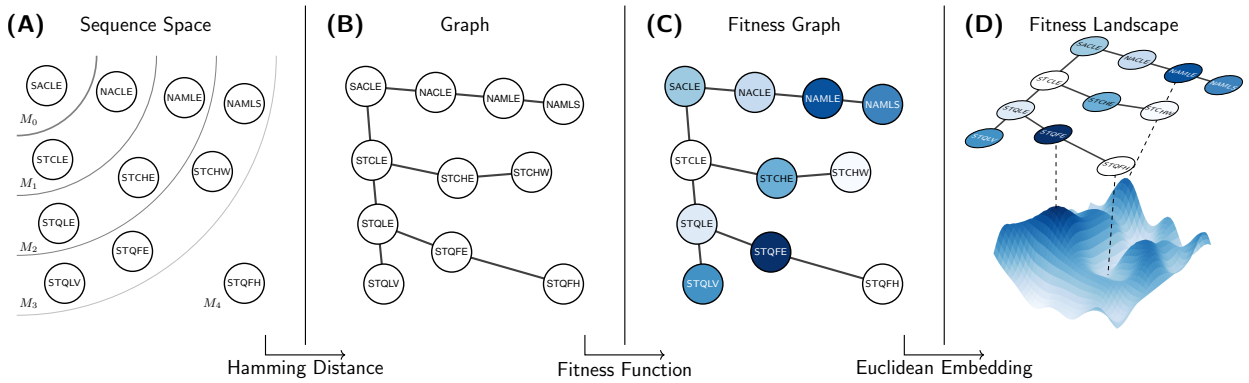

Figure 1: **(A)** A section of combinatorial sequence space (for a length 5 protein), with mutational regimes highlighted relative to the *SACLE* seed sequence (M_0). **(B)** A graph-theoretic representation of protein sequence space, where edges connect sequences (i.e. nodes) one mutation away from each other (i.e. Hamming distance of 1). **(C)** A *fitness function* maps each sequence to a fitness, producing a *fitness graph*. Here, the darker blues indicate a higher fitness against some arbitrary fitness metric. **(D)** The fitness graph can also be construed as the more well-known concept of the *fitness landscape* by embedding it within Euclidean space.

sequences are similar, there are smooth changes in fitness as the sequence space is traversed, resulting in a smooth or *correlated* landscape. At the other extreme, in a rugged or *uncorrelated* landscape, adjacent sequences can have sharp changes in fitness (akin to sharp crags and deep chasms in geographical landscapes), making reliable prediction extremely challenging. Highly rugged landscapes generally result from epistasis i.e where the change in fitness is dependent on the genetic context or sequence background in which it is introduced[15,16]

The goal of ML sequence-fitness predictors is to approximate the fitness function for a particular functional subspace (i.e. a specific protein fold) of sequence space, ideally from sparse experimental sampling of the fitness landscape. Typically, the experimental sampling is constrained within a certain mutational regime M_h, where h is the Hamming distance from the central, or wild-type, sequence, which could be arbitrarily chosen[17,18]. Once the fitness function is learnt by the ML algorithm, the ML algorithm can interpolate, i.e. predict the fitnesses of sequences within the mutational regimes of the training set and extrapolate, i.e. predict the fitnesses of sequences outside the mutational regimes of the training set.

We posit 4 metrics against which to rigorously assess ML algorithm performance in the con-

text of the fitness landscape: (1) ability to interpolate within mutational regimes present in the training set; (2) ability to extrapolate beyond the mutational regimes present in the training set; (3) ability to cope with increasing landscape ruggedness; and (4) sensitivity to sparse experimental sampling of the fitness landscape in the training data. Formally assessing all of these metrics requires a single training dataset that exhaustively spans multiple mutational regimes and is of increasing, and *a priori* known, ruggedness. The former is prohibitive experimentally, and the latter is not viable in reality because the same region of sequence space (i.e. protein) cannot have multiple ruggedness values against the same fitness criterion.

The shortcomings of incomplete and untuneable datasets can be overcome through the use of simulated landscapes. The NK landscape[19] of Kauffman and colleagues provides a precise and mathematically rigorous simulated fitness landscape for a length N protein, and has previously been used in a narrow sense to evaluate ML models[20]. In the NK landscape, fitnesses are distributed according to the K parameter, which controls the degree of epistasis, and hence ruggedness of the landscape, with higher K values produce higher degrees of epistasis and hence ruggedness. In other words, the NK mathematical model allows complete landscapes to be generated and for the ruggedness of these landscapes to be tuned.

Here we develop a unifying framework for the field that provides a standard against which ML algorithms can be rigorously evaluated, as well as a useful conceptual framework for thinking about ML problems in the context of protein sequence-function relationships. By (1) simulating NK landscapes of increasing ruggedness, (2) stratifying the sequences in each into mutational regimes (M_h), and (3) training on training sets of varying sampling density, it becomes possible to evaluate any ML model against the four metrics posited. This approach provides a benchmark against which the capacity of an ML model to learn sequence-fitness relationships can be assessed and compared. We analyze a variety of models, including decision-tree based architectures (random forest (RF)[21] and gradient boosted (GB) decision tree regression[22]); deep learning architectures (multilayer perceptron (MLP) and recurrent neural network (RNN)[23]); and linear regression methods (multivariate linear regression (Linear) and support vector regression (SVR)). Tests with the NK landscape,

alongside well-characterized experimental datasets show that the NK landscape is a faithful substitute for real fitness landscapes and a valid benchmark landscape for ML-driven sequence-fitness prediction. Finally, we provide a lightweight, open-source Python package that simplifies the integration of protein datasets with sci-kit learn[24] and provides the tools to perform the evaluations presented in this paper.

Results

Interpolation and extrapolation performance. On any dataset that spans multiple mutational regimes, interpolation and extrapolation can be assessed concurrently by stratifying the data into mutation regimes M_h from some seed sequence M_0 (which could be the wild-type), followed by expanding the training data to include an increasing number of mutational regimes (Figure 2).

This test was performed for all models on NK landscapes of increasing ruggedness ($K = 0, 1, 2, 3, 4$ for $N = 5$). The seed sequence was arbitrarily chosen. To avoid sampling artefacts, five instances of each landscape were randomly generated, and the results on each were averaged as a variant on cross validation. Each of the training mutational regimes were split into 80%/20% training and testing data respectively, while the mutational regimes beyond the training data were used in their entirety to test the models' performance (Figure 2). Their performance on each test set is evaluated as the coefficient of determination (R^2) between the ground-truth value and the ML predicted value.

To validate the NK model as a benchmark, experimental datasets must be used. It is not experimentally tractable for any real protein dataset to exhaustively search three, let alone five, mutational regimes. As a compromise, we used G domain B1 dataset (GB1-4) of Wu *et al.*[18], a combinatorial fitness landscape of all 20 amino acids at 4 unique amino acid sites thus spanning the sequence space (M_1 - M_4) at the 4 positions that are mutated, permitting extrapolation to be tested on an experimental dataset. Results of these tests for all model architectures are shown in 3.

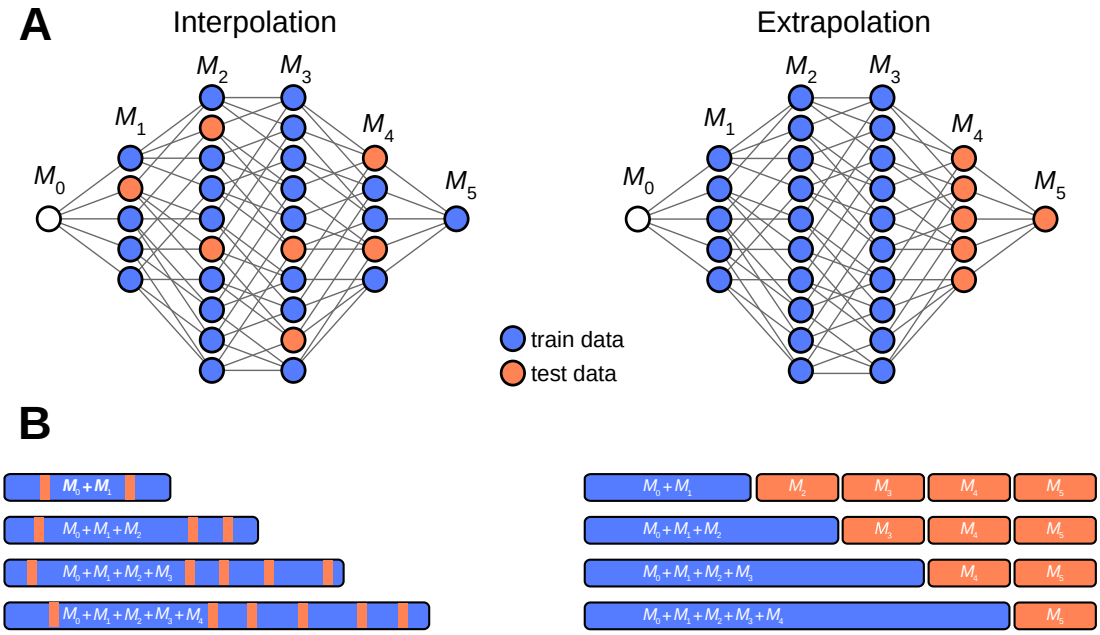

Figure 2: **(A)** Small model combinatorial sequence spaces, stratified into mutational regimes $M_1 - M_5$ from a seed sequence M_0 (black node). In **interpolation**, the same mutational regimes (in this case, M_0 to M_3) are represented in the train and test sets, typically with a random 80%:20% train/test split; in **extrapolation**, different mutational regimes are represented in the train set (here, M_0 to M_3) and the test set (here M_4 to M_5). In extrapolation as performed in this work, there is 100 % representation from the train mutational regimes and the testing mutational regimes in the train and test sets, respectively. Note that in later sections where sensitivity to sparse data is tested, the 80% train set is further ablated to different sampling densities. **(B)** Paradigm for assessing interpolation and extrapolation performance. For interpolation, performance is evaluated on an expanding number of mutational regimes, with the test set being drawn randomly from the data. For extrapolation, the model is trained on an expanding number of mutational regimes, but performance is evaluated on mutational regimes *not included* in the train set.

As expected, all models perform worse on interpolation as ruggedness increases. At the limit of the completely uncorrelated landscapes ($K = 4$ for $N = 5$) all models catastrophically fail. The decision-tree methods outperformed the other categories, with RF leading in terms of capacity to cope with ruggedness, and GB producing the best interpolation and extrapolation in lower mutational regimes. Both deep learning architectures (i.e. multilayer perceptron (MLP) and recurrent neural network (RNN)) show strong performance, with the RNN competing with the GB at extrapolation and, to a lesser extend, interpolation on $K = 0$; however this performance diminishes sharply with increasing ruggedness. The linear methods showed the worst performance, with SVR outperforming the linear model.

The trends observed across the NK landscape are clearly replicated in the experimental dataset, with the decision tree based models performing the best and demonstrating minor extrapolative ability. The RNN is the best of the remaining models; however, its predictions only become meaningful when large volumes of data are available; i.e., when more mutational regimes are used in the training dataset. Once again, the linear and SVR models perform very poorly on this task.

The decrease in performance as a result of ruggedness is seen across all models. Increasing ruggedness implies that the landscape is becoming less correlated and therefore random. For any statistical inference method, including the ML models used here, as the complexity/randomness of a function increases, the accuracy of approximation decreases if the number of samples is kept constant. A corollary of this is that as the complexity/randomness of a function increases, it is necessary for an inference method to be informed with greater number of samples. In the limit of a completely random function, the number of samples required is equal to the number of points in the function. This dependence of machine learning performance on landscape ruggedness highlights the need for a clear understanding of ruggedness in experimental landscapes.

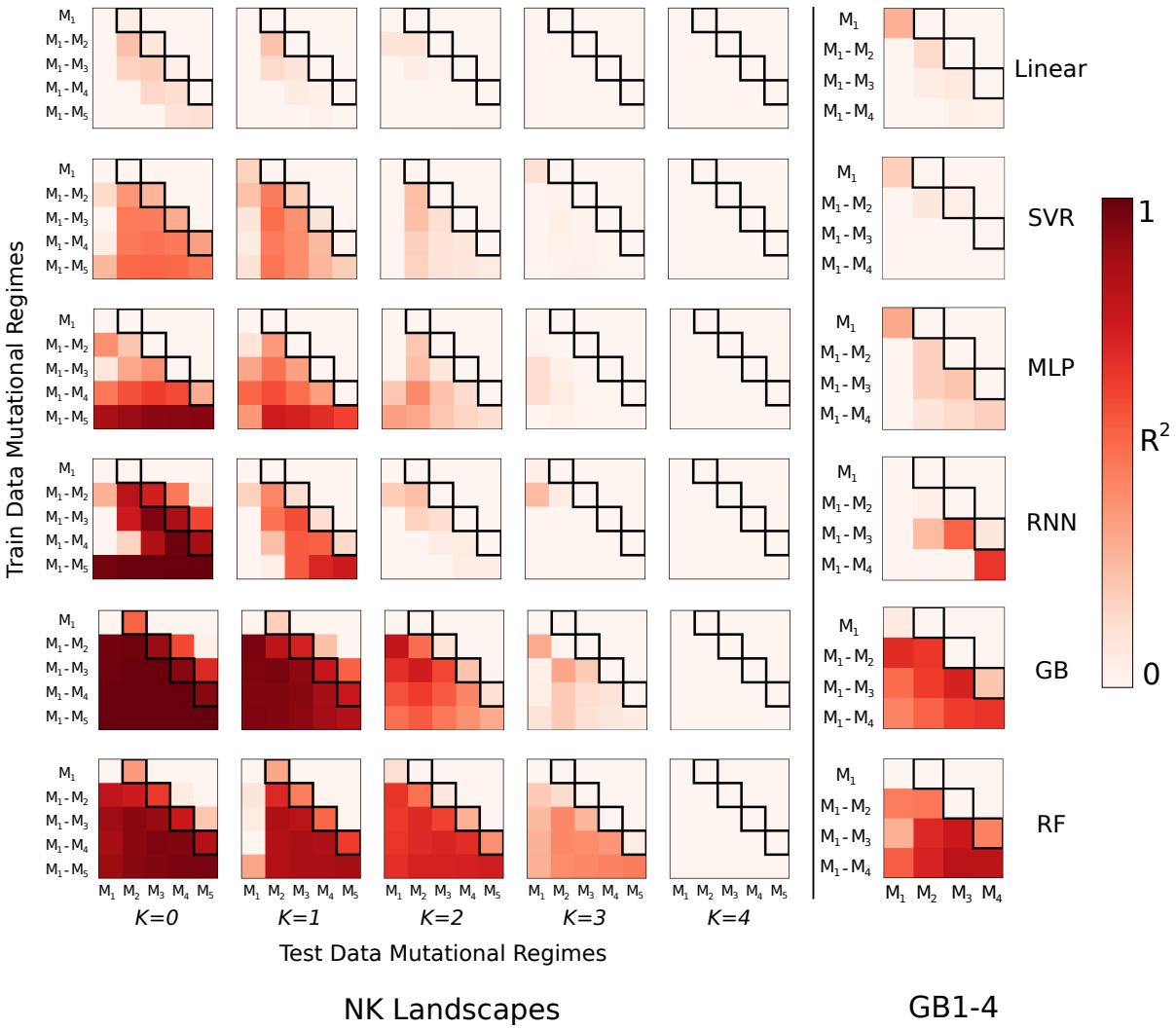

Figure 3: Interpolation and extrapolation performance for all models on both NK landscapes and GB1-4. Five different NK landscapes with ruggedness (K) between 0-4 are shown, with results being averaged across five instances of each. Experimental results used 5 fold cross validation to ensure robustness. Each heatmap shows how the correlation between ground truth and predicted values changes as a larger number of mutational regimes are used for training. Extrapolation into one mutational regimes beyond the training data is highlighted in cells with a black border.

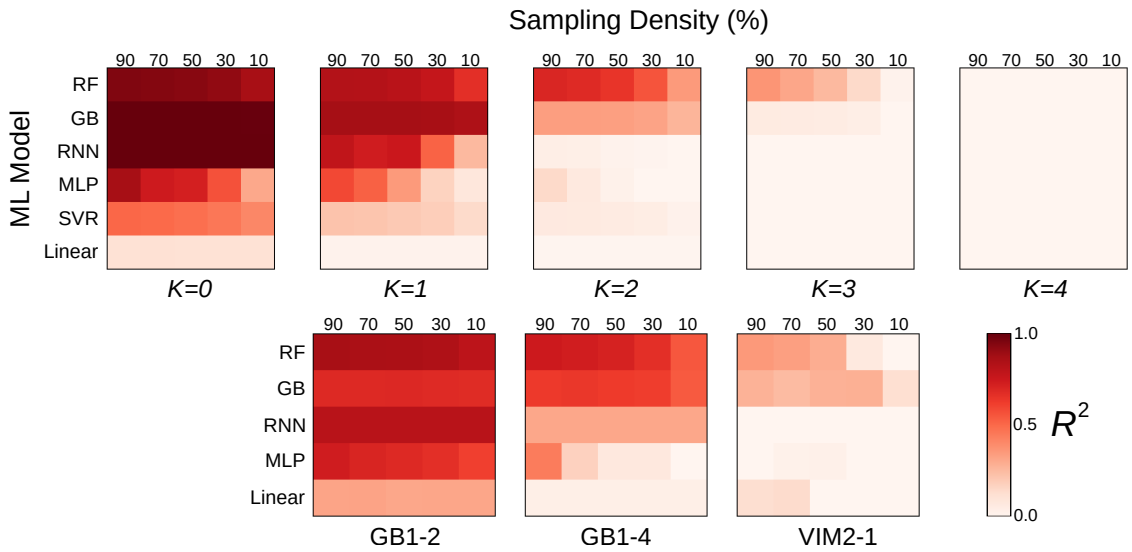

Figure 4: Performance on ablated train data for RF, GB, MLP, RNN, SVR and Linear models. Train data was randomly ablated to 90, 70, 50, 30 and 10% of original. Heatmap shows model performance (R^2) on test data at each train data sampling density, for NK landscapes of increasing ruggedness ($K = 0, .., 4$), and for the 3 experimental datasets.

Robustness to sparse training data. Learning from sparse training data is resource-efficient, saving time, effort and money. The ideal ML sequence-function algorithm interpolates and extrapolates accurately after training on sparse training data. Recognising (1) that extrapolation performance varies directly with interpolation performance as landscape ruggedness increases, and (2) running extrapolation tests for each model is computationally very expensive, we recommend testing robustness to sparse training data in the context of interpolation. Note that, even when sparse, training data should sample diverse regions of the relevant fitness landscape in order to maximally inform the ML algorithm. To test robustness to training data sparsity, the test set is kept at a constant size (20% of the total dataset) while the train set is randomly ablated to the desired sampling density (10%, 30%, 50%, 70%, and 90%). The dataset is shuffled and fresh train/test splits are initialised for each sampling density before train set ablation in order to avoid sampling artefacts.

We tested robustness to training data ablation for our chosen ML models (Figure 4). As seen in interpolation and extrapolation, the performance of all models declines as landscape ruggedness increases. Additionally, higher ruggedness values also cause a quicker drop in

performance as less training data is sampled; the magnitude of this decline varies between models. At $K = 0$, GB and RNN models have the best performance, but this declines rapidly for the RNN. Overall, RF is the most robust to data ablation. Deep learning models such as the MLP and RNN are notoriously data-dependent and performed poorly at lower sampling densities. This illustrates a severe limitation not only in protein sequence-fitness prediction but in ML more broadly; the complexity and large number of parameters of deep learning models often means that large amounts of data are required to train them appropriately. In the setting of protein sequence-fitness prediction, acquiring such large training datasets experimentally can be prohibitively resource intensive. Benchmarking candidate models for robustness to low training data volume permits judicious choice of ML model, thereby reducing the resource burdens of experimental data collection.

To confirm whether the results from the simulated dataset were consistent with empirical datasets, data ablation studies were performed. We again used experimental datasets to validate the NK model as a benchmark. In this case, we selected the GB1-4 dataset, the G domain B1 dataset (GB1-2) dataset of Olson *et al.*[17], a combinatorial sequence space of all 20 amino acids at all sequence positions except the first, and the VIM2 (VIM2-1) dataset of Tokuriki *et al.*[25], in which deep mutational scanning was used to characterize the functional behaviour of 5600 single amino acid variants of the β-lactam degrading antibiotic resistance enzyme VIM-2. The SVR did not train in a reasonable amount of time (>750 CPU hours) and was thus omitted. The results are shown in Figure 4. Again, the experimental results conform to the NK landscape results Again, the experimental results conform to the NK landscape results, with decision-tree based models exhibiting the strongest predictive performance and resistance to ablation. The MLP produces strong results but demonstrates susceptibility to ablation. This sensitivity is particularly severe for GB1-4, where the performance rapidly declines. It seems that experimental landscapes have substantial variance in terms of their ruggedness, with GB1-2 ostensibly being quite smooth, leading to commensurately high-quality predictions. Based on model performance, the VIM2-1 dataset seems to exhibit a much higher degree of ruggedness. Nevertheless, there may be other factors that lead to decreased performance such as the very different sequence lengths. All models

struggle to interpolate when trained only on the first mutational regime (M_1) (Figure 3). Initially this appears to be a problem of data scarcity. However, there is a more subtle effect that we believe is responsible for this behaviour: the demanding nature of what we term *positional extrapolation*.

Positional extrapolation. We have thus far considered extrapolation in the context of concurrent amino acid mutations, e.g. changing two positions simultaneously produces a sequence at a Hamming distance of 2 from the seed sequence, and thus in M_2. An additional type of extrapolation that is rarely considered involves tasking the model with predicting the influence of a mutation at a position has not been altered in the training set. To distinguish these two forms, we refer to the previously introduced form (extrapolating to higher mutational regimes) as regime extrapolation and this new form as positional extrapolation. In positional extrapolation, the model is tasked with predicting the effect of mutations at sequence positions that are never modified in the training data. This type of extrapolation is particularly pronounced when assessing interpolation on M_1, as any splitting of this regime is likely to remove all training examples of some positions in the sequence, thus demanding that the model perform positional extrapolation. To highlight the challenge presented by position extrapolation we benchmark all models on five NK landscapes, but the stratification in this case is not performed by distance, but rather by position.

The results of these tests are shown in Figure 5. The heatmap appears featureless; however, what is shown is a complete collapse of all models in terms of their performance on this task. This highlights the sheer difficulty of this task; the model cannot learn from any examples of the positions being modified. This requirement is the cause of the "not in either" segment of the data, which contains samples in which, of the two positions modified, one falls into the train category and one into the test category. Providing the model with these types of sequences would give it information about its test regimes, and thus compromise the rigor of the challenge, hence producing large numbers of samples which must be omitted. While seemingly contrived, this represents a very common expectation of these models: given

Figure 5: Results for positional extrapolation across all ML architectures except SVR (due to its inability to train in the allotted time). The left hand side demonstrates the separation of the sequence based on mutatted positions, with the bar chart showing how much data is in each of the three possible datasets with "not in either" representing species that have one mutation in each of the two categories.

modifications to a portion of the protein sequence, predict what happens at unmodified positions. These results cast serious doubt as to the ability of simple predictive models, (e.g. the ones tested here) to perform such a task. We posit that only models with explicit structural understanding of the protein, or a large amount of transfer learning (in which large unlabelled datasets are used for unsupervised training before fine-tuning this system with labelled data) would be capable of meaningfully performing positional extrapolation. Indeed, recent work that produces meaningful predictions from low data sampling relies on this precise idea[10,11]. When analysing the performance of future models to understand where predictions do, and do not, make sense, an understanding of the nuances of extrapolation is clearly required so as to avoid relying on simple scalar metrics like mean absolute deviation, which can obfuscate areas in which the model is catastrophically failing.

Estimating landscape ruggedness. Our results have highlighted the critical importance of landscape ruggedness in ML sequence-fitness prediction performance. In simulated landscapes, ruggedness is known *a priori*; in empirical landscapes, it must be estimated. There is no clear consensus regarding the best methods to estimate landscape ruggedness[26]; therefore, we chose two metrics that performed similarly and showed capacity to scale to large empirical datasets: (1) extrema ruggedness estimation (an extension of maxima estimation[26,27]) which counts the number of extrema in the landscape and normalises to the number of data points; and (2) r/s slope estimation[28], which compares the slope, s, of a linear fit to the root mean squared error r to produce an estimate of ruggedness/non-linearity that a linear model cannot account for.

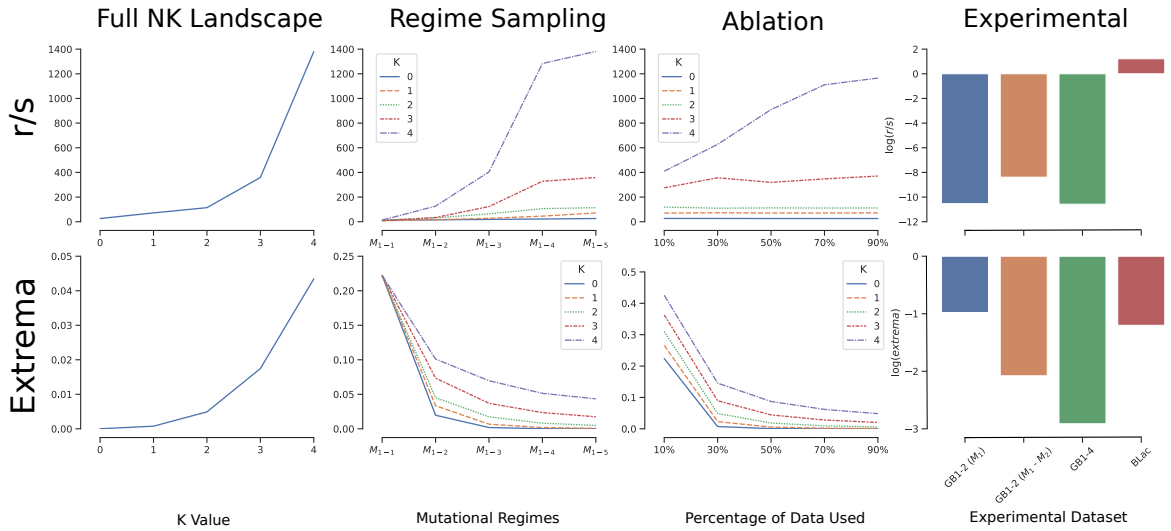

Figure 6: **Ruggedness Estimator Comparison:** Top row shows r/s estimation, while bottom shows extrema estimator. **Full** NK **landscape** shows the estimation for the entire NK landscape as a function of K. **Extrapolation** shows how the estimation changes when limited mutational regimes are used. **Ablation** shows how the estimates change as fractions of the data are used. **Experimental** shows how ruggedness across the three datasets from the methods, with GB1-2 having been split into its two mutational regimes, with the respective regimes identified using M_h notation.

Figure 6 demonstrates that both ruggedness estimators produce sensible results for the NK landscapes, showing a steep rise in both values as K increases. The NK landscape, with its *a priori* known ruggedness, provides an ideal control for testing ruggedness estimators. We hypothesised that estimated ruggedness would show dependence on the mutational regimes considered in the calculation: as more mutational regimes are included, a richer view of the landscape is obtained (e.g. higher orders of epistasis), providing a better estimate of its features. In order to test this hypothesis, we again stratified NK landscapes of increasing *a priori* known ruggedness ($K = 0$ to $K = 4$) into mutational regimes ($M_1 - M_5$), and then assessed ruggedness on expanding subsets of these (Figure 6). From the results, it is evident that both r/s and extrema metrics improve as more mutational regimes are considered in the calculations, with all estimated ruggedness values being equal (and therefore meaningless) for M_1. The reason for this is that if fewer neighbors an individual node are considered,

then the likelihood of the node being assessed as an extrema increases. This produces the diminshing ruggedness value as more points are considered as seen in both regime sampling and ablation. This is concerning, as it corresponds to the majority of datasets obtained from deep sequencing, which typically only sample M_1, making it difficult to accurately approximate ruggedness. Nevertheless, it is a logical consequence of the data structure: ruggedness is largely derived from higher-order epistasis, i.e. the non-additivity of mutations, which cannot be captured by examining a Hamming distance of 1 from the seed sequence.

An additional factor that can influence ruggedness estimation is the sampling density of the landscape. To evaluate the effect of data sparsity on ruggedness calculation, we ablated the NK landscape data randomly and assessed the performance of the estimators using diminishing fractions of the data. Both r/s and extrema estimation function well even with limited data (Figure 6); however, the r/s metric shows greater dependence on data than does the extrema metric. These two tests (regime sampling and ablation) roughly describe the two paradigms for data acquisition in protein science: (1) deep scans and (2) shallow scanning of many mutational regimes, commonly seen in directed evoliution experiments.

Having performed these control experiments on NK landscapes, we then tested the ruggedness across experimental datasets (Figure 6). We stratified GB1-2 into the mutational regimes M_1 and $M_1 - M_2$; GB1-4 contains 4 mutational regimes ($M_1 - M_4$); and the VIM2-1 β-lactamase dataset contains only 1 mutational regime (M_1). In accordance with our results for NK landscapes, we infer that the extrema ruggedness values obtained for GB1-2 M_1 and β-lactamase are effectively meaningless, being almost indistinguishable. However, estimates based on r/s ruggedness vary enormously. This is because this method is capable of estimating *magnitude of ruggedness*, which extrema estimation cannot[26]. The implication that the β-lactamase dataset is far more rugged conforms with the difficulties which all models have demonstrated in generating meaningful predictions on it.

Interestingly, the ruggedness of any landscape can be broadly inferred in a reverse manner by assessing the performance of the ML models studied here, and comparing the results to

that of the NK landscape. While not rigorous, this provides a useful heuristic to develop an expectation for the performance of models on novel datasets.

Sensitivity to sequence length. As a final test, the sensitivity of each model to sequence length was assessed. To test this, a random sequence of a particular length N was generated and the NK data was injected at randomly chosen indices to produce a new resultant sequence (Figure 7 A). These new datasets are representative of the commonly performed experiment of mutating key residues within a protein; for example, as in GB1-4.

We tested sensitivity to sequence length for our chosen models against NK landscapes of increasing ruggedness ($K = 0, 1, 2, 3, 4$ for $N = 5$). Sequence lengths varying from 10-100 in increments of 10 were tested for all models (Figure 7 B).

Strikingly, most models show no length dependence (e.g. RF and GB), with only the deep learning architectures demonstrating sensitivity, although non-linearly. This reflects the fundamental architectural differences in these models: RF and GB models create decision trees based on input features, identifying changing sequence positions as important and relegating the remainder of the sequence as inconsequential. The deep learning models show variation in performance with sequence length, reflecting variability in their training, with longer sequences making this variability more noticeable.

Discussion

This work clearly demonstrates the capacity of the NK landscape framework to serve as a valid benchmark for a diverse array of ML architectures, with the predictive performance on simulated NK landscape data matching the observed performance on experimental data. In addition to this, four key metrics for predictions in protein space were articulated and combined into a holistic benchmark set that we hope will aid in standardisation and interpretation of results within this new field.

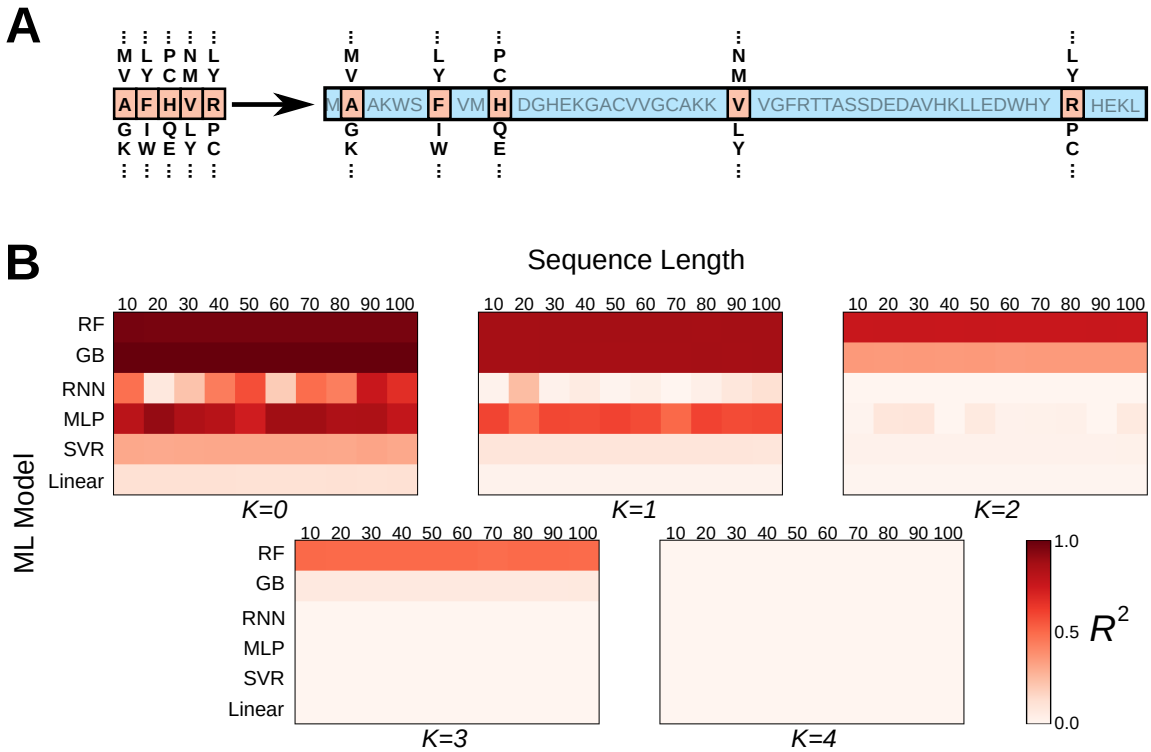

Figure 7: **(A)** Length transformation of NK landscape sequences. Arbitrary sequences can be injected between positions (columns) of sequences generated from NK landscapes (left) to create a sequence S of arbitrary length (right). The number of positions mutated in sequence S (right) is $n = N$, where N is the length parameter of the NK landscape. Note that while injected sequences are arbitrary, they must be consistent (i.e. the same) within each landscape. **(B)** Sensitivity to sequence length for RF, GB, MLP, RNN, SVR and Linear models. NK data was injected into sequence lengths of length 10 - 100 in instances of 10. Heatmap shows model performance (R^2) on test data for landscapes of increasing ruggedness ($K = 0, .., 4$).

The purpose of this work was not to provide a ranking of ML models for the sequence-fitness problem; nevertheless, a variety of insights into the advantages and limitations of various models were obtained. These were backed by both experimental and theoretical results. To summarise briefly, RF models are the best out-of-the-box performer, capable of predicting on highly rugged landscapes with strong extrapolation and interpolation. Additionally, it has strong resistance to ablation and shows no length dependency. Its key limitation is that its predictions are meaningless outside of the training input domain. As such, the problem must be carefully considered prior to their application. Deep learning methods showed the most capacity to improve, but their need for vast volumes of data was apparent in both the theoretical and experimental results. Additionally they show the greatest variability with training, presenting challenges if automating the process of model selection is desired. Finally, linear based models are best avoided if possible due to their poor predictive performance, and, in the case of SVR, enormous training time.

Benchmarking with the NK landscapes has two limitations: sequence length must remain constant, which is a minor issue as length is typically held constant in protein engineering. More importantly, benchmarking using the NK landscape (or any theoretical landscape) lacks biophysical grounding; it is better viewed as a faithful reproduction of the fundamental problem of predicting sequence-fitness relationships on protein landscapes. This means that NK-benchmarking should not be used with transfer learning models or with models that incorporate structural information. Because the NK model lacks physical grounding, transferred knowledge, when projected onto the non-physical NK-benchmark, will lead to meaningless results. However, due to the immense computational cost in applying transfer learning, this benchmark may serve as an ideal companion, enabling cheap and rapid benchmarking of potential predictors to couple with transfer learning.

To our knowledge, this study is the first time that extrapolation has been explicitly broken down into sub-problems, i.e. regime and positional extrapolation. The enormous challenge presented by positional extrapolation highlights the importance of transfer learning and structural information, and should draw focus onto recent work which has utilized transfer

learning to perform predictions with very small amounts of data[10,11].

Critically, this work emphasizes the dependence of ML models on the underlying landscape, with all of the four metrics showing strong dependency on landscape ruggedness. This dependency demands that the underlying landscapes are considered when ML is applied, and directs the field towards better understanding and quantification of the properties, particularly ruggedness, of experimental landscapes. We recommend that researched looking to apply ML to protein engineering carefully consider the nature of the experimental system and perform ruggedness estimations (which is fully supported in the provided codebase - see Code Availability), as well as consider the extent to which positional extrapolation is required with the given dataset. In estimating ruggedness, we recommend the r/s method due to its sensitivity to magnitude and rapid runtime. Soberingly, fundamental limits on statistical inference for rugged landscapes may mean that there are some proteins for which ML is not suitable; the frequency and nature of such protein fitness landscapes is an open and fundamental question, one which has important implications for the role of ML in the future of protein science and engineering.

Methods

The NK landscape. The NK landscape Kauffman and colleagues[19] is a theoretical model of combinatorial fitness landscapes that captures the key characteristic of these landscapes: *ruggedness*, which results from *epistasis*. N gives the number of positions in the sequence (i.e. length of the sequence); K gives the number of positions that interact with a given position i; thus, K gives the order of epistasis in the landscape. For any position i, the set of positions that interact with it are $v_i = \{v_{i,1}, v_{i,1}, ..., v_{i,K}\}$. These are typically assigned uniformly and independently at random[26]. The fitness of some sequence X of length N is given by:

$$f(X) = \sum_{i=1}^{N} f_i(\{x_j\}_{j \in v_i}) \tag{1}$$

Here $\{x_j\}_{j \in v_i}$ is the set of positions position i interacts with (i.e. the *interaction set*). If any of the positions in the interaction set for some position i are mutated, a new fitness f_i is randomly assigned from a continuous uniform distribution on the interval $[0, 1)$; else, f_i stays the same. It then follows that as the interaction set of i increases (with increasing K), the probability that a mutation anywhere in the sequence will affect the fitness contribution f_i of position i also increases (i.e. the likelihood of f_i being re-assigned increases). This means that a single mutation affects the sequence fitness $f(X)$ more drastically, thereby increasing the ruggedness of the fitness landscape.

All NK landscapes were generated using custom code that was adapted from Obolski and colleagues[29]. For each ruggedness value, five landscapes were generated from randomized initial conditions. Each of these landscapes contained all sequences of length five ($N = 5$), using the first 10 canonical amino acids of the single letter amino acid alphabet. Limiting the amino acid pool was necessary to restrict the population size to a number that is tractable for many ML architectures, while also maintaining a sequence length that enables a wide variety of ruggedness values

Experimental datasets. In order to validate the NK landscape as a sound benchmark for ML sequence-fitness prediction performance, we compared ML model performance qualitative rankings against a number of published empirical datasets from deep sequencing. These included 2 datasets of the B1 domain of immunoglobin-G (IgG) binding protein G (hereafter referred to as protein GB1), and one VIM-2 lactamase dataset.

Protein GB1 double-mutation scan dataset. Protein GB1 is a well-characterised prokaryotic protein extensively used as a model system in protein science, along with use in biotechnology applications like antibody purification. We used the deep mutational scan dataset from Olson and colleagues[17] that exhaustively characterised all possible pairwise mutations in the domain. The dataset consists of 536962 variants and both their input and selection counts which were transformed into fitness values using equation 2, with input and selection count

are used as defined by[17]. We refer to this dataset as protein GB1-2 (the number after the hyphen denotes the mutational regimes spanned by the dataset). Within our framework, this dataset spans the first 2 mutational regimes ($M_1 - M_2$) in the sequence space of protein GB1.

$$\text{fitness}(X) = \log\left(\frac{\text{selection count}(X)}{\text{input count}(X) + 1}\right) \tag{2}$$

Protein GB1 quadruple-mutation dataset. We used a second deep sequencing dataset of protein GB1 of Wu and colleagues[18]. This dataset exhaustively samples the sequence space of of 4 positions (positions V39, D40, G41 and V54) near the C-terminus of the domain, consisting of 149361 variants. Fitness was calculated according to equation 2. In our framework, this dataset spans the first 4 mutational regimes ($M_1 - M_4$) in the sequence space of these 4 positions (i.e. it spans the entire sequence space of these 4 positions). We refer to this dataset as protein GB1-4. Fitnesses on this dataset were calculated in the same way as GB1-2 (Equation 2).

Beta-lactamase dataset. We used a recently published deep mutational scan of the VIM-2 lactamase from Chen and colleagues[25]. This dataset iteratively mutates the entire 267 position sequence of the VIM-2 lactamase, mutating to each position to each of the 20 canonical amino acids. In our framework, this dataset spans the first mutational regime M_1 in the sequence space of the VIM-2 lactamase. We refer to this dataset as VIM2-1. Fitnesses in this dataset are the $log_2(EC_{50})$ values provided in the dataset.

ML architectures. To demonstrate broad validity of this approach, we tested the benchmark against a wide variety of established ML architectures, including all outlined in Yang, Wu, and Arnold's review[1] (with the exception of Gaussian Process Regression due its inability to scale to large datasets). The models fell broadly into three categories: (1) the decision-tree based architectures random forest[21] (RF) and gradient boosted[22] (GB) decision tree regression); (2) deep learning architectures including a multilayer perceptron (MLP) and recurrent neural network[23] (RNN); and (3) linear regression methods multivariate linear regression, referred to simply as Linear and support vector regression[30] (SVR). We classified

SVR as a linear method because while it uses a non-linear kernel to project data into higher dimensional space, it is a linear model model that is then fitted. All models were implemented in scikit-learn[24] except the RNN, which was implemented in PyTorch[31] and trained in scikit-learn by wrapping it within the skorch package[32].

The purpose of this work is not to demonstrate the best possible performance, and in each architecture, but instead to benchmark the performance of each model on the NK landscape with the metrics posited, and then show that this corresponds to their performance on experimental datasets. Therefore, careful hyperparameter optimisation was not performed, and models were used largely out-of-the-box with some simple parameter modification to increase performance. Better performance could be extracted from these models with more intensive hyperparameter tuning. Performance of each model was determined as the R^2 correlation between the predicted values and the ground truths. Because negative correlations can be arbitrarily large with our chosen package, negative values were set to zero for ease of visualisation.

Benchmarking package. Our key goal is to stimulate progress in the field with clear objectives and benchmarks. Therefore, we seek not only to demonstrate the validity of our benchmarks, but also to provide robust and user-friendly software to the community. Our open-source benchmarking package (Python) is available for install on Github (`https://github.com/acmater/NK_Benchmark`), and includes all code used to generate landscapes, models, and perform the tests. Importantly, we contribute a protein landscape class that can be used to handle both simulated and empirical protein landscapes, including methods for running ruggedness tests, segregating sequences into mutational regimes, and more complex indexing operations, enabling rapid stratification of the data and therefore more robust benchmarking of models on novel landscapes.

Code Availability

Our open-source benchmarking package (Python 3.7) is available for install on Github (`https://github.com/acmater/NK_Benchmarking`), and includes all code and Jupyter Notebooks required to reproduce the NK landscapes, models, perform the tests and reproduce the figures.

Acknowledgements

A.C.M. thanks the Australian National University and the Westpac Scholars Trust for PhD scholarships. This project was funded by ARC Centre of Excellence in Peptide and Protein Science. The funders had no role in study design, data collection and interpretation, or the decision to submit the work for publication.

Author Contributions

A.C.M., M.S. and C.J.J. conceived the study. A.C.M. and M.S. conceived the experiments and methods, wrote the software, analysed results, prepared the figures and wrote and edited the manuscript. A.C.M unified and vectorized the software database, curated data, acquired and applied computational resources and performed the simulation experiments. C.J.J. supervised and administered the project, acquired funding, reviewed and edited the manuscript.

References

[1] Kevin K Yang, Zachary Wu, and Frances H Arnold. "Machine-learning-guided directed evolution for protein engineering". *Nature Methods* 16.8 (2019), pp. 687–694.

[2] Claire N Bedbrook et al. "Machine learning-guided channelrhodopsin engineering enables minimally invasive optogenetics". *Nature Methods* 16.11 (2019), pp. 1176–1184.

[3] Philip A. Romero, Andreas Krause, and Frances H. Arnold. "Navigating the protein fitness landscape with Gaussian processes". *Proceedings of the National Academy of Sciences* 110.3 (2013), E193–E201.

[4] Emmi Jokinen, Markus Heinonen, and Harri Lähdesmäki. "mGPfusion: predicting protein stability changes with Gaussian process kernel learning and data fusion". *Bioinformatics* 34.13 (June 2018), pp. i274–i283.

[5] Douglas E. V. Pires, David B. Ascher, and Tom L. Blundell. "mCSM: predicting the effects of mutations in proteins using graph-based signatures". *Bioinformatics* 30.3 (Nov. 2013), pp. 335–342.

[6] Jorge Fernandez-de-Cossio-Diaz, Guido Uguzzoni, and Andrea Pagnani. "Unsupervised inference of protein fitness landscape from deep mutational scan". *Molecular Biology and Evolution* (2020).

[7] Babak Alipanahi et al. "Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning". *Nature Biotechnology* 33.8 (July 2015), pp. 831–838.

[8] Haoyang Zeng et al. "Convolutional neural network architectures for predicting DNA–protein binding". *Bioinformatics* 32.12 (June 2016), pp. i121–i127.

[9] Jianjun Hu and Zhonghao Liu. "DeepMHC: Deep Convolutional Neural Networks for High-performance peptide-MHC Binding Affinity Prediction" (Dec. 2017).

[10] Ethan C. Alley et al. "Unified rational protein engineering with sequence-based deep representation learning". *Nature Methods* 16.12 (Oct. 2019), pp. 1315–1322.

[11] Surojit Biswas et al. "Low-N protein engineering with data-efficient deep learning" (Jan. 2020).

[12] Sewall Wright. "The roles of mutation, inbreeding, crossbreeding, and selection in evolution". *Proceedings of the Sixth International Congress on Genetics* 1.8 (1932), pp. 355–366.

[13] John Maynard Smith. "Natural Selection and the Concept of a Protein Space". *Nature* 225.5232 (1970), pp. 563–564.

[14] R. W. Hamming. "Error Detecting and Error Correcting Codes". *Bell System Technical Journal* 29.2 (Apr. 1950), pp. 147–160.

[15] Tyler N. Starr and Joseph W. Thornton. "Epistasis in protein evolution". *Protein Science* 25.7 (Feb. 2016), pp. 1204–1218.

[16] Daniel M Weinreich, Richard A Watson, and Lin Chao. "Sign epistasis and genetic constraint on evolutionary trajectories". *Evolution; international journal of organic evolution* 59.6 (June 2005), pp. 1165–1174.

[17] C. Anders Olson, Nicholas C. Wu, and Ren Sun. "A Comprehensive Biophysical Description of Pairwise Epistasis throughout an Entire Protein Domain". *Current Biology* 24.22 (2014), pp. 2643–2651.

[18] Nicholas C Wu et al. "Adaptation in protein fitness landscapes is facilitated by indirect paths". *eLife* 5 (July 2016). Ed. by Richard A Neher, e16965.

[19] Stuart A Kauffman and Edward D Weinberger. "The NK model of rugged fitness landscapes and its application to maturation of the immune response". *Journal of Theoretical Biology* 141.2 (1989), pp. 211–245.

[20] Richard Fox. "Directed molecular evolution by machine learning and the influence of nonlinear interactions". *Journal of Theoretical Biology* 234.2 (2005), pp. 187–199.

[21] Leo Breiman. "Random Forests". *Machine Learning* 45.1 (2001), pp. 5–32.

[22] J Friedman. "Greedy function approximation: A gradient boosting machine." 2001.

[23] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". *Nature* 323 (1986), p. 533.

[24] F Pedregosa et al. "Scikit-learn: Machine Learning in {P}ython". *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[25] John Z Chen, Douglas M Fowler, and Nobuhiko Tokuriki. "Comprehensive exploration of the translocation, stability and substrate recognition requirements in VIM-2 lactamase". *eLife* 9 (June 2020).

[26] Ivan G. Szendro et al. "Quantitative analyses of empirical fitness landscapes". *Journal of Statistical Mechanics: Theory and Experiment* 2013.1 (2013).

[27] Frank J. Poelwijk et al. "Reciprocal sign epistasis is a necessary condition for multipeaked fitness landscapes". *Journal of Theoretical Biology* 272.1 (2011), pp. 141–144.

[28] Takuyo Aita, Masahiro Iwakura, and Yuzuru Husimi. “A cross-section of the fitness landscape of dihydrofolate reductase”. *Protein Engineering, Design and Selection* 14.9 (Sept. 2001), pp. 633–638.

[29] Uri Obolski, Yoav Ram, and Lilach Hadany. “Key issues review: evolution on rugged adaptive landscapes”. *Reports on Progress in Physics* 81.1 (Dec. 2017), p. 012602.

[30] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. *Machine Learning* 20.3 (1995), pp. 273–297.

[31] Adam Paszke et al. “Automatic differentiation in PyTorch” (2017).

[32] Marian Tietz et al. *skorch: A scikit-learn compatible neural network library that wraps PyTorch*. July 2017.