Evaluating *eUniRep* and other protein feature representations for *in silico* directed evolution

Andrew Favor

Ivan Jayapurna

August 7, 2020

Abstract

This study analyzes and adds to the Low-N protein engineering with data-efficient deep learning work done by Biswas et al. We provide a complete, open-source, end-to-end re-implementation of the *in silico* protein engineering pipeline with improved computational efficiency, more detailed documentation, cleaner API and additional features to lower the barrier to entry for use of this pipeline as an engineering tool. We additionally perform a more thorough evaluation of the success and necessity of each step in the pipeline for *in silico* directed evolution, by re-implementing select portions of the study of TEM-1 β-lactamase, as well as applying the full *in silico* pipeline to two novel protein engineering tasks - increasing the melting temperature of plastic degrading enzyme IsPETase and improving the thermostability the MS2 bacteriophage's capsid protein. By comparing the performance of various UniRep-based feature representations we provide proof that linear kernels can be equivalent to additive fitness landscapes and outperform more complex models on small or simple mutation prediction tasks. This is assumed in many previous works but never explicitly shown. We believe it helps to elucidate the main strength of the eUniRep representation: its ability to overcome epistatic effects in proposing extensively mutated candidate sequences with optimized functionality.-

Introduction

The foundation of protein engineering lies in introducing new mutations for the purpose of modifying protein function. The first attempts to rationally design new mutants with improved function relied on leveraging chemical and biological domain knowledge to correlate human-interpretable patterns and motifs in secondary and tertiary folded protein structures with possible functionality improvements [1]. These traditional predictive pipelines require accurate characterization or prediction of the folded 3D structures of proteins, which is a difficult task [2]. Thus, one of the grand challenges of modern computational protein engineering is in skipping structure determination entirely, and instead developing accurate models to predict direct amino acid sequence-to-function relationships. Directed evolution is a hypothesis-free, non-structure dependent approach that has been efficiently utilized in order to optimize the functionality of biomolecular systems through the improvement of selected traits [3,4,5]. However, the original directed evolution pipeline is both slow and resource intensive - requiring multiple high-throughput iterations of sequencing and functional characterization. Moreover, desired protein functions are often difficult to test experimentally in high-throughput manner; thus, a correlated proxy fitness function is instead used as the predictive label. The search space (fitness landscape) of possible amino acid substitutions is both immense in size and sparse in protein structures that actually fold [6,7], let alone that yield improved fitness. Even when restricted to considering the number of variant primary sequences that can result from just two co-expressed mutations, it is virtually impossible to test the fitness of all possible combinations through physical synthesis or even computational modeling [3,8,9]. This necessitates the development of more sophisticated predictive models. Machine learning

techniques can be used to computationally guide the directed evolution process, reducing the number of experimental cycles required [10].

A remaining key bottleneck preventing further reductions in experimental cost is the high number of initial mutants (N) needed to develop any predictive model of protein fitness. In a recent manuscript, researchers in the lab of George Church propose a solution to this issue, using data-efficient deep learning to achieve low-N protein engineering [11]. The approach described by Biswas et al holds promise as a powerful tool in protein engineering: given only a small library of synthesized and characterized protein variants, and a single directed evolution cycle, one could engineer extensively modified proteins with optimized functionality. Several existing protein engineering procedures utilize a common Markov-chain Monte Carlo (MCMC) simulation regime with a Metropolis-Hastings acceptance criteria defined by a Boltzmann probability distribution, with a temperature parameter governing the rate at which new mutations are adopted over current states throughout the mutagenesis trajectory. The relative favorability of proposed mutations is approximated by statistical mechanics based energy calculations [12,13]. However, these approaches fall short in their ability to accurately predict the fitness of a given mutation, as their applied calculations are largely restricted to a protein's native folded state, and cannot predict whether a mutation allows the protein to assemble through a folding pathway to reach that state in the first place. The *in silico* directed evolution protocol in Biswas et al makes 2 major improvements to existing schemes. The first, is in exploiting the often ignored dark proteome by doing unsupervised pre-training on millions of uncharacterized protein sequences to learn a semantically rich, universal representation (UniRep) of protein sequences [14]. This representation is further fine-tuned on a smaller, curated subset of evolutionarily relevant sequences to get a final "evotuned" representation (eUniRep). The second key insight is in using a non-physical Boltzmann distribution that is a function of eUniRep-dependent fitness predictions in their MCMC *in silico* directed evolution. In this non-physical Boltzmann distribution, the exponent numerator argument is the difference between a proposed mutant's fitness and the current sequence's fitness value. A significant benefit is expected to be found through the use of this scheme, as the eUniRep representations are trained to encode features containing information on the natural laws that govern favorable folding and function capabilities for proteins. Optimization of this pipeline will be critical in the future of protein engineering for the development of various technologies ranging from nanomaterials and drug delivery [15] to enzymatic plastic degradation [16,17]. In this work we provide a complete, open-source, end-to-end re-implementation of the *in silico* UniRep protein engineering pipeline with improved computational efficiency, more detailed documentation, cleaner API and additional features to lower the barrier to entry for use of this pipeline as an engineering tool.

The cornerstone to the success of the low-N pipeline is the eUniRep representation. Better features leading to improved model performance is a well established tenet of machine learning, and it is becoming a more ubiquitously exploited way to improve model performance in protein bioinformatics as well [18,19,20,21]. To further explore the success and necessity of the eUniRep feature space, we use our modified pipeline to compare the performance of various protein feature representations, with various supervised models, for 3 vastly different proteins, optimizing for 3 different functions:

1. Increasing catalytic rate of TEM-1 β-lactamase (PDB: 1ZG4), using wild type and mutant fitness scores reported by Firnberg et al [22], as a reproducibility study to try replicate and add to the results and analysis presented in Biswas et al.
2. Increasing the melting temperature of *Ideonella sakaiensis* PETase (IsPETase, PDB: 6QGC), a recently discovered enzyme that can degrade polyethylene terephthalate (PET) and other aromatic polyesters. Plastic waste is an ever growing problem and PETase is revolutionary in its ability to degrade the ubiquitously used PET under mild conditions. However, its low melting temperature and inherent instability greatly limit its industrial scale use - thus presenting a need to engineer more thermally stable mutants. Moreover, as there are a multitude of research groups working on PETase engineering, it allows us to directly compare the eUniRep *in silico* directed evolution protein engineering approach to other recently published computational protein engineering pipelines, such as the GRAPE method [23]. Cui et al have used greedy algorithms and k-means clustering to design an IsPETase mutant with a melting temperature elevated by 31 degrees from the wild type, using a similarly low-N number of

starting mutants.

3. Finally, we chose the icosahedral coat-protein of the MS2 bacteriophage (PDB: 2MS2), due to the availability of extensive fitness data for mutated variants of this protein allowing for in depth analysis of epistatic effects [15,24]. Additionally, MS2's virus-like capsid particles are able to provide a stable scaffolding for the development of nanomaterials, with a versatile range of new functionalities and a remarkable tolerance to extensive synthetic modification [25,26,27].

Method

Existing Pipeline

We initially re-implemented the 5 *in-silico* steps of the low-N protein engineering pipeline as outlined in Biswas et al [11] for a target protein. This *in-silico* method assumes prior wet-lab functional characterization of a small number (low-N) of random mutants of the wild-type target protein:

1. Learn a global, 1900-dimensional protein feature representation by training a mLSTM to do next character prediction on $>$ 20 million raw (unlabeled) amino acid sequences [14]. This is the "UniRep" representation.
2. Curate a smaller (order of 10,000) dataset of characterized protein sequences that are known to be evolutionarily related to the target protein.
3. Learn features local to the target protein by fine-tuning the weights from step 1 on the dataset from step 2. This is the "evotuned UniRep" or "eUniRep" representation.
4. Train a "simple" supervised top model (enseble ridge regression with sparse refit) on the 1900-dimensional feature space representation of a small number ($<$100) of characterized (labeled) wild-type mutants of the target protein (obtained by passing the protein sequences through the mLSTM trained in step 3). This top model provides an end-to-end sequence-to-function model for proteins local to the target protein.
5. Markov Chain Monte Carlo (MCMC) based *in silico* directed evolution on the target protein, outputting mutated sequences that are predicted by the top model in step 4 to have the most improved function relative to wild-type ($>$WT). Although the in-silico pipeline is complete, the top mutant sequences must then be functionally characterized to complete the full directed evolution cycle and confirm functional improvement in the engineered protein sequences.

As the training in step 1 was reported to take 3.5 weeks of wall-clock time on an AWS instance, we opted to use the weights provided by Alley et al [14] as our starting point. Due to the minimal open-source code available, we largely re-implemented the remainder of the pipeline from scratch, following provided method descriptions in Biswas et al. We identified 5 key areas for improvement with our re-implementation:

1. Curation of evolutionarily similar sequences for evotuning

The EBI JackHMMer web application [28] used to curate the training set of sequences for evotuning is unable to reliably handle a high volume of output hits for similar protein sequences. The alternative to install and run HMMer locally requires downloading the entire ReferenceProteomes database, which was unrealistic given our local storage constraints. A potential workaround could be to script around a local install of HMMer to search the ReferenceProteomes database from an online source. In our modified pipeline, we draft a formalized procedure for the curation of the local dataset using the Pfam and InterPro online databases and APIs (**Supplementary 1**).

2. mLSTM training and representation fetching speed and memory improvements

The TensorFlow implementation of the mLSTM model used by Biswas et al was unable to process sequences >275 amino acids in length due to memory constraints, even with up to 16GB of RAM. This poses a major issue for many proteins with sequence lengths greater than this, including PETase, with a sequence length of approximately 300. We collaborated to complete an open-source JAX re-implementation of the mLSTM to get UniRep representations of sequences, as well as evotuning to get eUniRep sequence representations [29]. JAX was chosen for its ability to achieve 100x speedup in passing new protein sequences through the trained mLSTM to get the 1900-dimensional (e)UniRep representations. Technical implementation details and performance comparisons can be read in Kummer et al, and a specific comparison of the performance of variable and fixed batch size training is presented in **Supplementary 2**. Our JAX implementation crucially requires less memory to run, affording the ability to evotune on longer protein sequences.

Evolutionary fine-tuning of the UniRep representations were performed on an Amazon Web Services EC2 instance (g4dn.2xlarge: 8 vCPUs, 32 GiB RAM, and one NVIDIA T4 Tensor Core GPU). Each evotuning job was run for 25 epochs. Additionally, a local-evotuning procedure was performed, wherein training weights were initialized with random values, rather than the pre-trained Global UniRep weights. Evotuning was performed with learning rates of 10^{-5}. All evotuning procedures were performed with learning rates of 10^{-5}, which yielded loss-convergence after the first few epochs (**Fig. 1**).

3. Rigorous top model selection and fine tuning

Biswas et al only report performance comparisons for 4 top model types: Lasso-Lars, Ridge, Ridge with sparse refit, and ensemble ridge with sparse refit - all with minimal information presented on what hyperparameters were used and how the well the top models fit. Moreover, *in silico* top model performance comparisons are only given in the form of retrospective experiments for avGFP (Green Fluorescent protein Aequorea victoria), based off which it is both unconvincing that (A) pre-training on 20 million sequences is required, as Local eUniRep appears to show comparable performance, and (B) that ensembled ridge regression with sparse refit is required, as it only demonstrates minor improvement over basic ridge regression in fold improvement over random sampling, while requiring a greater training time and a larger number of hyperparameters to tune. Motivated by the lack of reported results comparing and evaluating various top model performances, we implemented a thorough, general top model evaluation script that can be run on each new target protein to determine the best top model and hyperparameters for a given dataset.

4. New scoring function to evaluate top model performance

To quantify the performance of a predictive model within this specific context of application, it is important consider its ultimate use case: the Metropolis-Hastings acceptance criteria. In the MCMC simulations that emulate directed evolution, the probability of a proposed mutant sequence being accepted as the new sequence at each iteration is dependent on whether the proposed sequence has a greater fitness score than the current sequence. Thus, an appropriate top model would not necessarily prioritize predicting fitness scores that optimize the average closeness to the experimental fitness scores, but would rather prioritize accurately predicting the relative magnitude of each fitness score with respect to the values of other mutant sequences being considered. In the methods described by Biswas et al, the only purely *in silico* top model characterization is done with the following scoring metric: use a given top model to do a ranked sorting of a holdout set of mutant sequences. The given score is based on counting the number of mutants in the top 10% of the sorted set that have fitness greater than the wild type, as a ratio over averaged, random 10% samplings of the dataset . The shortcoming of this scoring scheme is its lack of generalizability. The choice to look at the top 10% is an arbitrary one, and it requires as input the wild type fitness value, specific to

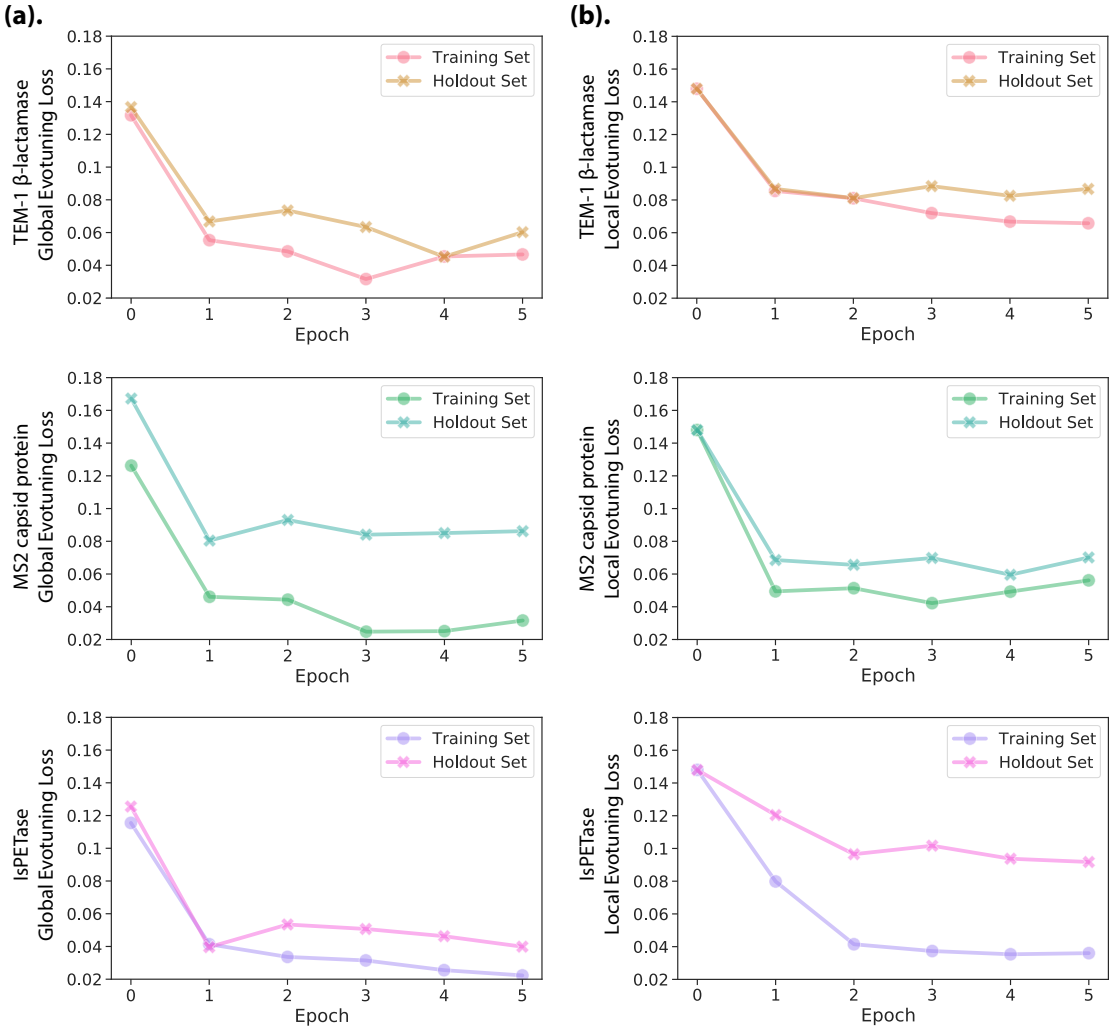

Figure 1: Loss-trajectories for the first five epochs of evotuning procedure for TEM-1 β-lactamase, MS2 capsid protein, and IsPETase. **(a).** Global evotuning trajectories: parameters are initiated as the globally trained UniRep weights; learning-rate = 10^{-5}. **(b).** Local evotuning: parameters are initiated as random values; learning-rate = 10^{-5}.

each protein and function. Furthermore, scores will likely not be comparable across different datasets, even within the same protein and function, as the scores are highly dataset distribution dependent (**Fig. 2**). To solve these issues and optimize parameters for the task of fitness score ranking, we developed a new scoring function (**Supplementary 3**).

5. Quality of life improvements

Finally and perhaps most importantly, to improve end-user usability and lower the barrier of entry for non-machine learning experts to utilize this pipeline in their research, we have made our entire pipeline well documented, open-source and easily customizable.

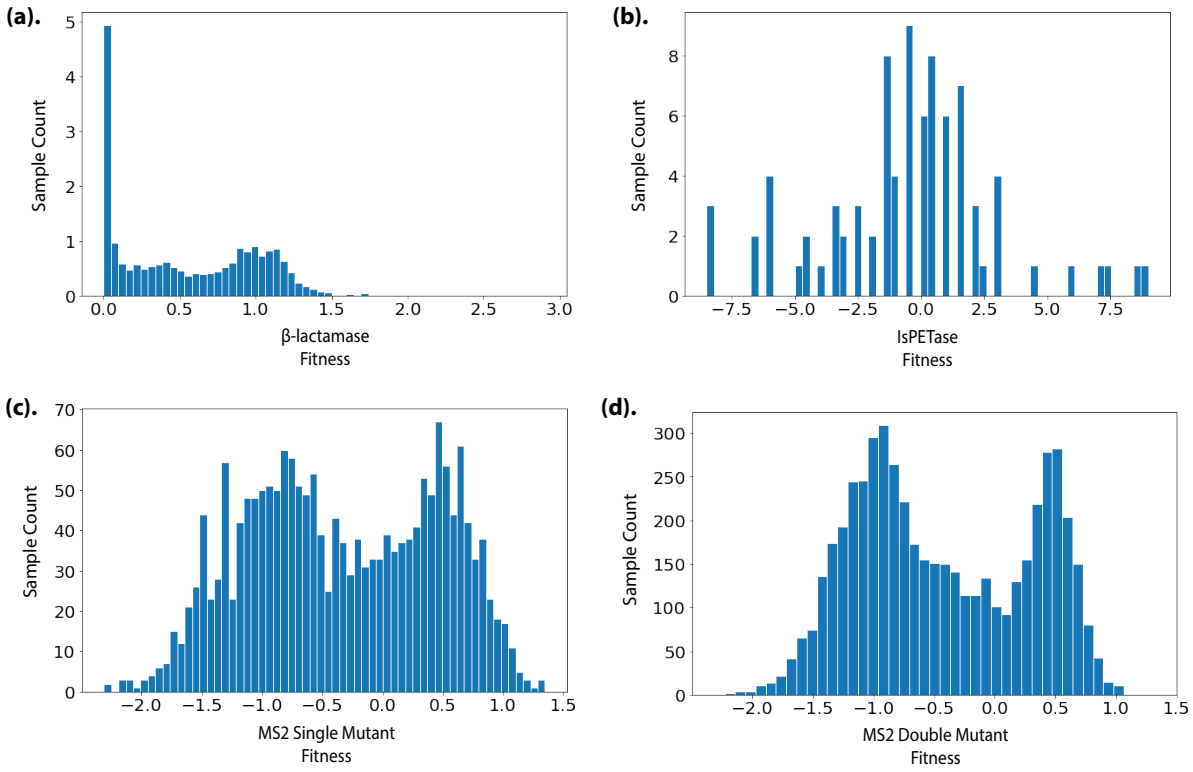

Figure 2: Histograms of the fitness score distributions of protein mutants from four datasets **(a).** TEM-1 β-lactamase single mutants, **(b).** IsPETase single mutants, **(c).** MS2 single mutants, and **(d).** MS2 double mutants. For MS2 single and double mutants, the wild-type fitness is approximately 0.5. For PETase, fitness scores represent change in T_m from wild-type, so the wild-type fitness is 0. The wild-type fitness for the TEM-1 β-lactamase dataset is approximately 1.

Feature Analysis

With the re-implemented pipeline, we replicate the analysis done in Biswas et al, comparing various protein feature representations with some additional comparisons: UniRep, several different evotuned weights (eUniReps), locally evotuned weights (Local Unirep), one-hot encodings of the full amino acid sequence, and reduced dimension representations by PCA of all of the above. This served to evaluate whether eUniRep is functional and necessary, across the different proteins and functions that we sought to optimize.

Results & Discussion

The success and necessity of feature representations varies across 3 different tasks

Our first objective was to confirm that our pipeline can reproduce the results presented for TEM-1 β-lactamase by Biswas et al. Using our fitness-ranking error function, we calculated the percent reduction in ranking-error relative to random sampling. Calculations were performed for four different mutant datasets, using several different embedded sequence representations, with performance compared across several different training-batch sizes **(Fig. 3)**.

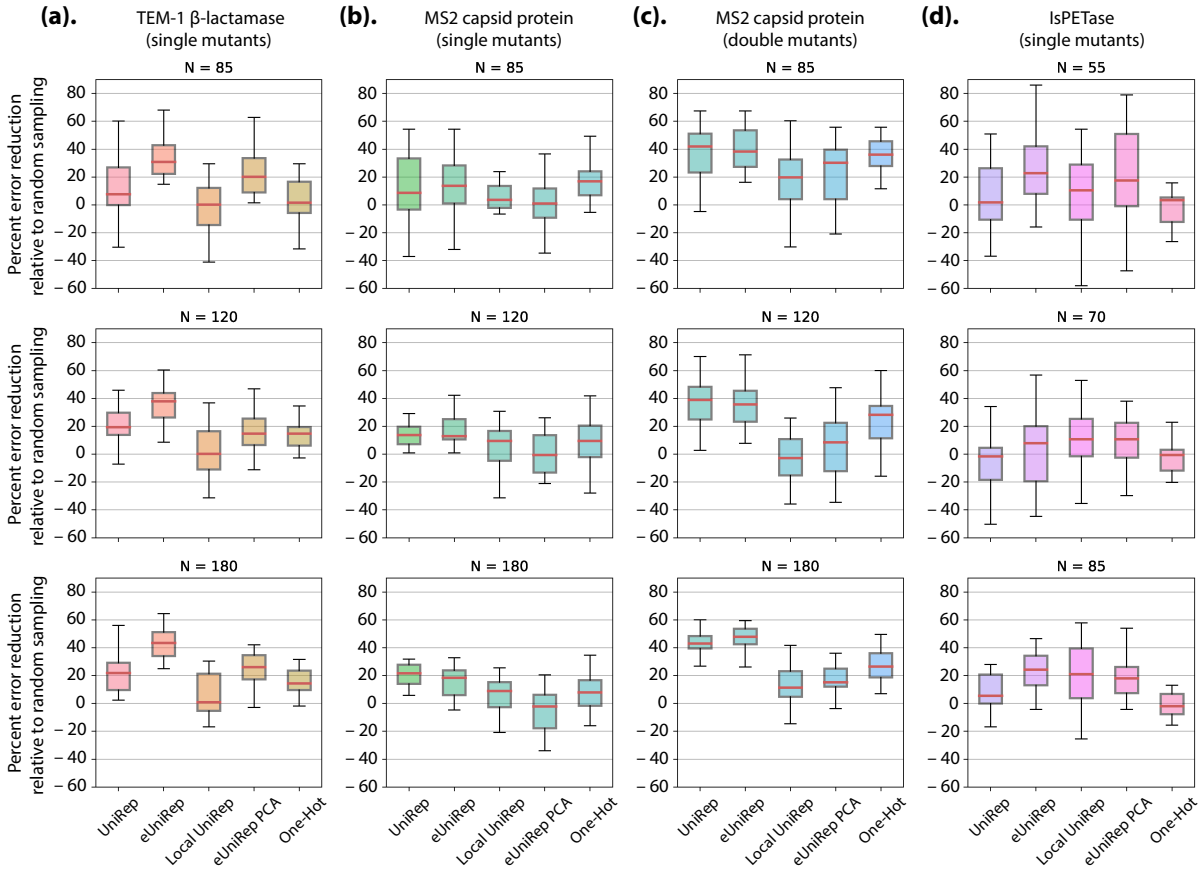

Figure 3: Percent error reduction in predicting the relative fitness rankings of test-set mutants, relative to random sampling, compared across three different batch sizes, and 20 randomized trials; datasets include **(a).** TEM-1 β-lactamase, **(b).** MS2 single-mutants, **(c).** MS2 double-mutants, and **(d).** IsPETase single-mutants. For each batch size (N), training and test sets were selected using a 80:20 percent batch size split.

For the TEM-1 β-lactamase dataset, percent error reduction was calculated for training-batch sizes of N

= 85, 120, and 180; the best reduction in error, averaged across all batch sizes, was achieved using a representation that was evotuned from UniRep initiation (eUniRep: average error reduction = 37.4%), followed by the Global UniRep representation itself (18.2%). A locally evotuned representation, trained from random parameter initiation, gave a 2.26 % error reduction. These results were compared against a one-hot encoding, which had an 11.2% average reduction in ranking-error. We additionally compared the performance of our evotuned TEM-1 weights with the TEM-1 weights provided by the Church Lab and used in Biswas et al (**Supplementary Fig. 2**). We show that the same top model performance achieved by the original TensorFlow model in 25 training epochs, can be achieved in a single epoch of training using our pipeline (JAX re-implementation) - a significant decrease in time and computation required. It should be further noted that this evotuning performance could be further fine tuned as training for further epochs has a negligible or detrimental impact (likely due to over-fitting) on top model performance (**Supplementary Fig. 3**).

For the single-mutant MS2 dataset, the Global UniRep, eUniRep and Local UniRep representations yielded 16.0, 15.8, and 5.66% average error reductions over random sampling, respectively; the one-hot encodings had intermediate performance, with 10.7% reduction in error. These analyses were repeated using a second dataset of MS2 mutants, containing fitness scores for all possible combinations of double amino acid substitutions within the regions of residues 71 to 76. Here we found a significant improvement in our top-model's ability to predict the relative fitness rankings of test-batch mutants, with percent error reductions of 38.7, 40.7, 9.88, and 28.1% for Global UniRep, eUniRep, Local UniRep, and one-hot encodings, respectively (averaged over training batch sizes of N = 85, 120, and 180). The same analysis described above was performed on our 85-sample IsPETase dataset, but with smaller training and test batch sizes, due to less data availability. Averaged over training-batch sizes of N = 55, 70 and 85, our predictive models produced ranking-error reductions of 4.01, 16.2, and 11.8% for the Global UniRep, eUniRep, and Local UniRep, respectively. The one-hot representations actually yielded a slight *increase* in error over random sampling, of 0.987%.

For all representations of all proteins, principal component analysis (PCA) was done (**Supplementary Fig. 4**). Training a top model on PCA reduced dimensions of any number of principal components (from N=3 to 1000) resulted in a worse top model performance when compared to globally trained eUniRep weights, in all cases. As shown in **Fig 3**, using a 3-component PCA representation of the evotuned representations (global initiation) yielded a decrease in ranking error of 20.9, 16.3, and 18.6% for TEM-1, MS2 double-mutants, and IsPETase, respectively, while the MS2 single-mutant dataset PCA representation gave an *increase* in ranking error of 1.35% relative to random sampling. This indicates that all 1900 features are necessary and serve to improve predictive capability. It is important to note that these analyses are limited in scope, as single mutants do not thoroughly represent evolutionary fitness landscapes. Multiple consecutive mutations need to be considered in order to represent epistatic effects, which may not be captured by more simple top models.

Epistasis detection: predicting the fitness of consecutive mutations from single mutant data

We define epistasis herein as the difference in the effect that multiple mutations have when expressed together, from the additive sum of the constituent individual mutation effects [30,31]. Due to the cooperative nature of interactions between amino acids within a protein, it is challenging to predict the effect that a single amino acid mutation has on the protein's overall ability to fold and retain a stable and functional state. The introduction of multiple mutations increases the complexity of this problem exponentially, and can result in novel interactions between amino acids, not observed in the wild type. To characterize the ability of the different embedded sequence representations to describe the effects of consecutive mutations, we trained our top model on MS2 single-mutant data and compared its predictions of double-mutant fitnesses to experimentally determined fitnesses of double mutants (**Fig. 4**).

The best predictive accuracy was achieved by use of one-hot sequence encodings with a mean square error

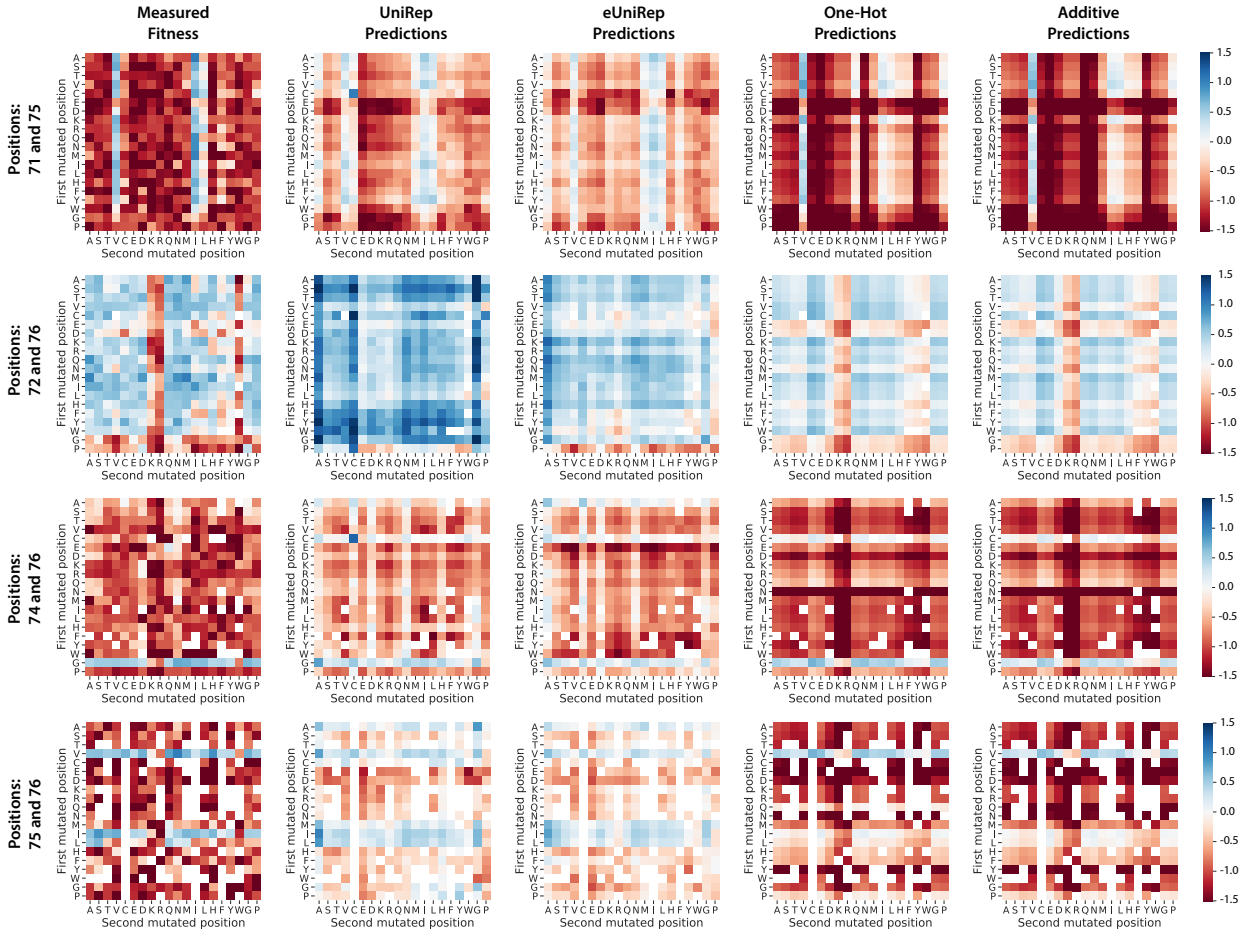

Figure 4: Comparison of top-model predictions of MS2 double-mutant fitness when trained on MS2 single-mutant dataset. Top models are trained using global UniRep, eUniRep, and one-hot sequence representations. Predicted fitness values are compared to experimentally measured fitness, and additive fitness-change predictions based on single-mutant differences from wild-type fitness. All models were trained using the full single-mutant fitness landscape dataset.

of 0.736, followed closely by Global UniRep and evotuned UniRep (global parameter initiation), which had mean square errors of 1.03 and 0.970, respectively. As an additional comparative metric to the one-hot representations, we calculated the additive predictions of fitness for double mutants (**Equation 2**), using the single-mutant fitness data; these predictions had a mean square error of 0.841, performing slightly worse than the one-hot representations. In the case of double-mutants, our analysis employs the following mathematical definition of the epistasis associated with a double-mutation, ij, with constituent single-mutations i and j:

$$\epsilon = \Delta y_{ij} - (\Delta y_i + \Delta y_j) \tag{1}$$

where $\Delta y_x = y_x - y_{wt}$, describes the change in fitness induced by mutation x, relative to the wild-type. In fitness-landscape regions where epistatic effects are minimal, the compounding effects of two mutations are expected to be "additive", such that the double-mutant fitness of mutations i and j can be predicted by:

$$\widehat{y}_{i,j} = y_{wt} + \Delta y_i + \Delta y_j \tag{2}$$

In previous works, it has been assumed that the effects of consecutive mutations trend with the additive changes in fitness predicted by the summed effects of their constituent single amino acid mutations [10]. In this work, we prove this mathematically (**Supplementary 4)** and by illustrative example (**Fig. 4**), where the one-hot trained model predictions are nearly identical to the calculated additive fitness-change predictions. This trend holds across all mutation positions in our MS2 double mutant dataset as can be seen in **Supplementary Figs. 5-7** for each mutation position. Although we find that in this case, UniRep-based representations are on average out-performed by one-hot representations, this does not necessarily suggest that learned sequence representations, like UniRep, offer negligible benefits over simple sequence representations, such as one-hot encodings. Linear models (such as full sequence one-hots) offer adequate performance on fitness landscapes where epistatic effects are limited, however, they will fail in contexts where a large number of consecutive mutants are introduced. In these cases the benefit of deep-learned representations is likely to become more pronounced. In our double mutants, the large degree of similarity between one-hot predictions and actual experimental values suggest that the double-mutant data was sampled from a region of considerably low epistatic effects. However, in directed evolution, epistatic complexity grows exponentially with each consecutive round of mutagenesis, as each additional mutation brings with it a range of new potential interactions between neighboring residues. We can thus expect superior UniRep performance over one-hot in our *in silico* directed evolution, as it has been trained to identify patterns of favorable or unfavorable local amino acid arrangements. On top of UniRep's ability to embed general information regarding amino acid arrangement patterns, it is likely that an evotuned representation would provide further value by embedding taxonomically-specific information, such as where specific residue arrangements should be located within a full amino acid sequence, for a given family of protein, such that the desired tertiary structures will assemble.

Top model selection & tuning for MCMC *in silico* directed evolution

In our analysis so far we have assumed the use of an optimized ridge regression top model. For the purpose of finding a model that could accurately predict the effects of a new mutation for a given sequence , we tested the predictive accuracy of several linear regressions, with a range of hyperparameters. While various regression models were tested inclusive of Lasso, Huber, RANSAC, and ensemble methods, we found minimal performance improvement with increased model complexity over simple linear ridge regression. Ridge regressions have a single regularization hyperparameter that can be optimized by cross-validation. In Biswas et al, a sparse refit is used with the effect of increasing regularization strength, to prevent overfitting to the starting mutant dataset, which is likely not representative of the actual fitness landscape. In our pipeline the scoring function for cross-validation is set as our custom ranking-error scoring function, so there is no need for any further hyperparameter refitting. Hyperparameters and batch sizes are selected before simulating directed evolution runs for each protein separately (**Fig. 5**), using the ranking-error function. For consistency, a selected alpha value of 10^{-2} and batch sizes of 85 were used for TEM-1, MS2, and PETase directed evolution simulations. Our evolution trajectories ran for 25 iterations of random mutagenesis coupled with selection steps based on the predicted fitness of proposed mutations (**Fig. 6**). Although we were able to effectively incorporate this component into our pipeline, and are therefore able to produce a large number of candidate mutants for the evaluated proteins, experimental validation is still required in order to determine whether the proposed variants do in fact possess the desired trait enhancements.

Open Questions: The importance of training mutant variance and fitness

A key result we found is the importance of the span or variance of the low-N input data on which the top model is trained. As an example, randomly sampling from the full TEM-1 dataset achieves far better top model performance in comparison to randomly sampling from large subsets (i.e. the first half) of the dataset. These findings corroborate what is hypothesized in Biswas et al. We believe this to be the primary

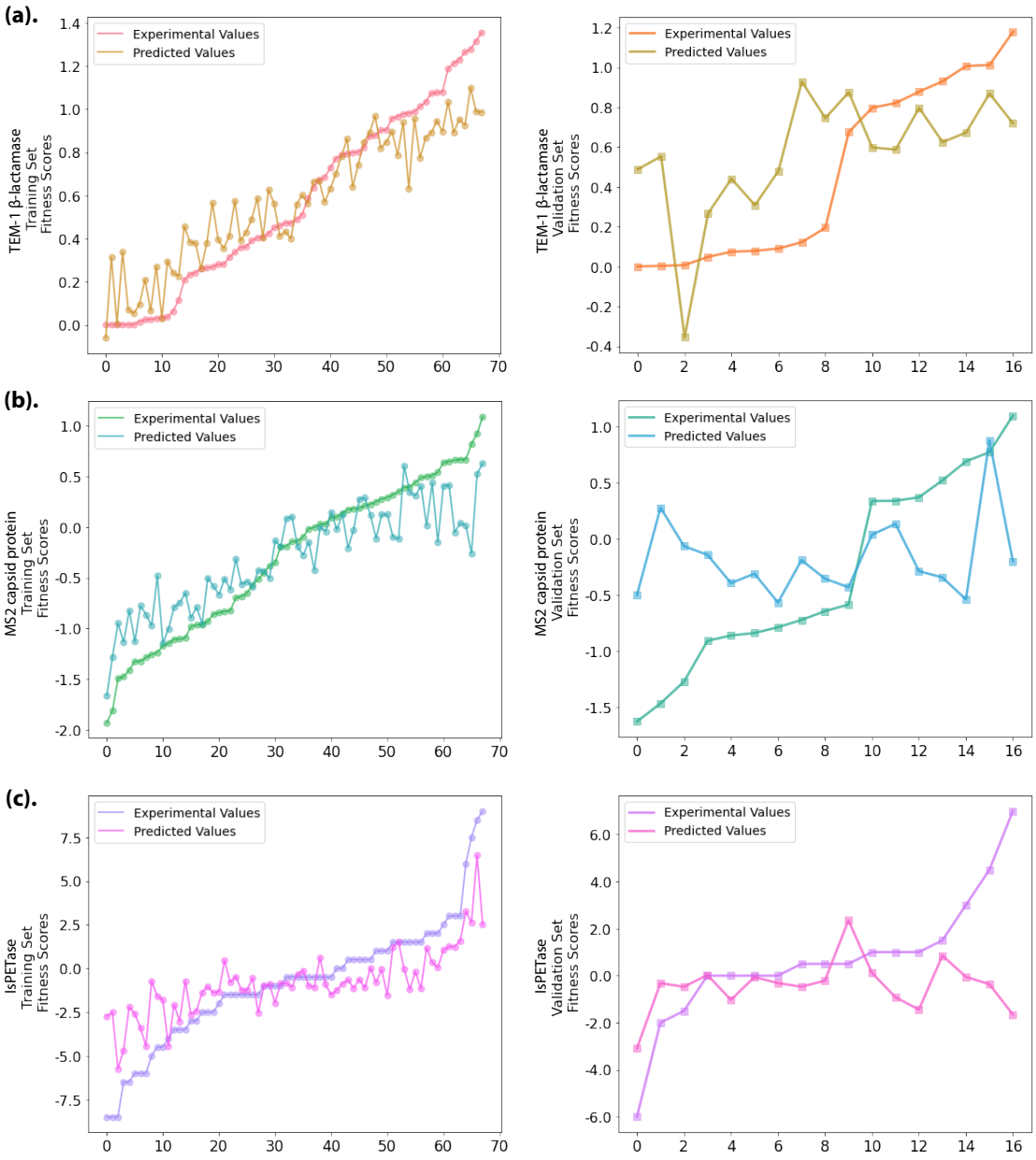

Figure 5: Predictive performance of linear top-models used in directed evolution simulations. Ridge regressions ($\alpha = 10^{-2}$) were trained and tested on single-mutant datasets of **(a).** TEM-1 β-lactamase, **(b).** MS2 capsid protein, and **(c).** IsPETase. Data for both training and test sets are shown sorted along the x-axis with respect to experimentally determined fitness scores.

explanation for our inability to achieve good top model performance for improving PETase fitness. Our small dataset of only 85 mutant samples, many of which representing mutations that occur in the middle of the amino acid sequence, likely does not adequately span the configuration space of possible mutations that we would be predicting for. However, this is an open question that still requires further analysis, as despite our eUniRep results matching those obtained by using Church Lab provided TEM-1 evotuned weights, it

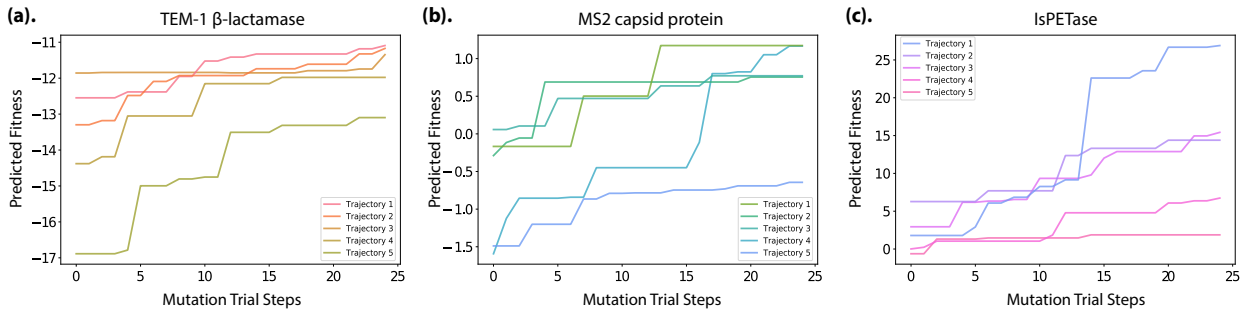

Figure 6: Five simulated 25-step trajectories of *in silico* directed evolution for variants of **(a).** TEM-1 β-lactamase, **(b).** MS2 capsid protein, and **(c).** IsPETase. In mutagenesis trial steps, fitness scores for candidate mutants are predicted by the linear models that were fit previously (Fig. 5)

is possible that our evotuning is still non-optimal. This sentiment is supported by the inferior performance of eUniRep MS2 compared to just global UniRep. The alternative hypothesis to explaining the lack of significant improvement from evotuning for PETase and MS2, is the inherent nature of the fitnesses being optimized. In the case of TEM-1 the fitness is catalytic activity, a trait specific to enzymes. Whereas T_m (for PETase) and stability (for MS2) are both universal to all proteins, thus any evolutionary local pre-training will not have added effect - it may even have a deleterious effect due to over-fitting.

Future Work

Beyond answering the open questions highlighted in the previous section, there are several additional avenues for further exploration and pipeline improvement. Due to the computational and time constraints involved in training the mLSTM, in this report we primarily explore top model performance and directed evolution. As a result, there is a lot of further work that can be done with the pre-training model. One curious finding from our few evotuning runs (**Fig. 1**) is that evotuning loss does not correlate with top model performance, per our scoring function (**Supplementary Fig. 3**). In order to further probe this relationship and determine the optimal stopping point for mLSTM training, directed evolution performance should be validated for various combinations of different learning rates and numbers of epochs. Finally, as is happening in various fields in bioinformatics and natural language processing, mLSTMs are being replaced by transformers [32], which could provide an interesting alternative implementation. Ultimately, the most important future work that needs to be done is experimental characterization of our directed evolution outputs.

We believe the biggest contribution of this work to be our complete, open-source, generalized, end-to-end re-implemented pipeline, that we hope researchers and engineers will find it helpful for use in their work. Perhaps someone reading this will have the lab setup to help do the experimental characterization, either on our 3 protein case studies, or a novel one of their own!

Software Repository

Full pipeline re-implementation: https://github.com/ivanjayapurna/low-n-protein-engineering

JAX-UniRep Implementation: https://github.com/ElArkk/jax-unirep

Acknowledgements

We would like to thank the Church Lab authors of both the UniRep and Low-N protein engineering papers for their fantastic work which inspired us to work on this project and develop our interest in computational protein engineering, as well as for sharing their trained weights with us; Eric Ma and Arkadij Kummer for the brilliant JAX-UniRep reimplementation and for teaching us a lot about best practices in working on open-source collaborative projects and designing clean APIs; and Prof. Jennifer Listgarten for inspiring us with the possibilities and potential of machine learning in biology and chemistry.

Supplementary Material

Supplementary discussions and figures can be found here

References

1.Lee, D., Redfern, O. & Orengo, C. Predicting protein function from sequence and structure.. *Nat Rev Mol Cell Biol* **8**, 995–1005 (2007).

2.Dill, K. A. & MacCallum, J. L. The protein-folding problem, 50 years on.. *Science* **338**, 1042–6 (2012).

3.Romero, P. A. & Arnold, F. H. Exploring protein fitness landscapes by directed evolution.. *Nat Rev Mol Cell Biol* **10**, 866–76 (2009).

4.Wintrode, P. L. & Arnold, F. H. Temperature adaptation of enzymes: lessons from laboratory evolution.. *Adv Protein Chem* **55**, 161–225 (2000).

5.Maheshri, N., Koerber, J. T., Kaspar, B. K. & Schaffer, D. V. Directed evolution of adeno-associated virus yields enhanced gene delivery vectors.. *Nat Biotechnol* **24**, 198–204 (2006).

6.Smith, J. M. Natural selection and the concept of a protein space.. *Nature* **225**, 563–4 (1970).

7.Ogbunugafor, C. B. A Reflection on 50 Years of John Maynard Smith's Protein Space.. *Genetics* **214**, 749–754 (2020).

8.Arnold, F. H. Combinatorial and computational challenges for biocatalyst design.. *Nature* **409**, 253–7 (2001).

9.Hopf, T. A. *et al.*. Mutation effects predicted from sequence co-variation.. *Nat Biotechnol* **35**, 128–135 (2017).

10.Yang, K. K., Wu, Z. & Arnold, F. H. Machine-learning-guided directed evolution for protein engineering.. *Nat Methods* **16**, 687–694 (2019).

11.Biswas, S., Khimulya, G., Alley, E. C., Esvelt, K. M. & Church, G. M. Low-N protein engineering with data-efficient deep learning. *bioRxiv* 2020.01.23.917682 (2020) doi:10.1101/2020.01.23.917682.

12.Beard, H., Cholleti, A., Pearlman, D., Sherman, W. & Loving, K. A. Applying physics-based scoring to calculate free energies of binding for single amino acid mutations in protein-protein complexes.. *PLoS One* **8**, e82849 (2013).

13.Release, S. 3: MacroModel, Schrödinger, LLC, New York, NY, 2014. *No Title* (2014).

14.Alley, E. C., Khimulya, G., Biswas, S., AlQuraishi, M. & Church, G. M. Unified rational protein engineering with sequence-based deep representation learning.. *Nat Methods* **16**, 1315–1322 (2019).

15.Hartman, E. C. *et al.*. Quantitative characterization of all single amino acid variants of a viral capsid-based drug delivery vehicle.. *Nat Commun* **9**, 1385 (2018).

16.Tournier, V. *et al.*. An engineered PET depolymerase to break down and recycle plastic bottles.. *Nature* **580**, 216–219 (2020).

17.Austin, H. P. *et al.*. Characterization and engineering of a plastic-degrading aromatic polyesterase.. *Proc Natl Acad Sci U S A* **115**, E4350–E4357 (2018).

18.Yang, K. K., Wu, Z., Bedbrook, C. N. & Arnold, F. H. Learned protein embeddings for machine learning.. *Bioinformatics* **34**, 4138 (2018).

19.Heinzinger, M. *et al.*. Modeling aspects of the language of life through transfer-learning protein sequences.. *BMC Bioinformatics* **20**, 723 (2019).

20.Raimondi, D., Orlando, G., Vranken, W. F. & Moreau, Y. Exploring the limitations of biophysical propensity scales coupled with machine learning for protein sequence analysis.. *Sci Rep* **9**, 16932 (2019).

21.Buchan, D. W. A. & Jones, D. T. Learning a functional grammar of protein domains using natural language word embedding techniques.. *Proteins* **88**, 616–624 (2020).

22.Firnberg, E., Labonte, J. W., Gray, J. J. & Ostermeier, M. A comprehensive, high-resolution map of a gene's fitness landscape.. *Mol Biol Evol* **31**, 1581–92 (2014).

23.Cui, Y. *et al.*. Computational redesign of PETase for plastic biodegradation by GRAPE strategy. *bioRxiv* (2019) doi:10.1101/787069.

24.Hartman, E. C. *et al.*. Experimental Evaluation of Coevolution in a Self-Assembling Particle. *Biochemistry* **58**, 1527–1538 (2018).

25.Peabody, D. S. Subunit fusion confers tolerance to peptide insertions in a virus coat protein. *Archives of biochemistry and biophysics* **347**, 85–92 (1997).

26.ElSohly, A. M. *et al.*. Synthetically modified viral capsids as versatile carriers for use in antibody-based cell targeting. *Bioconjugate chemistry* **26**, 1590–1596 (2015).

27.Kovacs, E. W. *et al.*. Dual-surface-modified bacteriophage MS2 as an ideal scaffold for a viral capsid-based drug delivery system. *Bioconjugate chemistry* **18**, 1140–1147 (2007).

28.Potter, S. C. *et al.*. HMMER web server: 2018 update.. *Nucleic Acids Res* **46**, W200–W204 (2018).

29.Kummer, A., Jayapurna, I. F. & Ma, E. J. jax-unirep: An accelerated, performant, and user-friendly implementation of UniRep in JAX. (2020).

30.Pokusaeva, V. *et al.*. Experimental assay of a fitness landscape on a macroevolutionary scale. *bioRxiv* 222778 (2018).

31.Sarkisyan, K. S. *et al.*. Local fitness landscape of the green fluorescent protein. *Nature* **533**, 397 (2016).

32.Vaswani, A. *et al.*. Attention Is All You Need. *CoRR* **abs/1706.03762**, (2017).