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

Andrew Favor

Ivan Jayapurna

August 3, 2020

Supplementary Discussion

S1: Evotuning pre-training dataset curation

Pre-training data set curation follows the following steps:

1. Search up the wild-type protein sequence on InterPro and PFAM Protein Sequence Fetch to get what family / clan it is in. Download all of these sequences as well as other related families / clans as suggested by PFAM. We recommend downloading via code snippets that fetch protein sequences using the InterPro API (these can be generated by InterPro).
2. If you have fewer sequences than desired (i.e. <50,000 for example, as note that some of these sequences may be duplicates or be invalid sequences) then search for keywords on InterPro and download those as outlined in the 2MS2 script. A JackHMMer search at this step may also optionally be attempted with the wild type sequence as the seed.
3. Clean the inputs by removing sequences with non-standard residues, anything with length greater than k, remove duplicate sequences.
4. Calculate Levenstein distances from highly desired mutant (if available, if not then from wild type sequence) on all clean sequences.
5. Generate training, in_domain validation and out_domain validation sets. The out_domain validation set is generated first using a distribution proportional to distances^4 - taking 10% of the total sequences. Then 10% of the total (11.1% of the remainder) is taken for the in_domain validation set, with the remaining 80% of original = the training set.

S2: Batching by sequence length versus random batching with sequence padding

In order to maximize computational speedup, the JAX implementation of the mLSTM model originally did not use sequence padding and instead trained on variable sized batches defined by sequences of the same length. Unfortunately, evotuning results using this implementation did not perform well in comparison to the original TensorFlow (TF) implementation used by the Church Lab. In practice, it is often seen that fixed batch-size training will yield better convergence, thus in order to improve performance the JAX implementation was modified to accommodate sequence padding (pad all sequences to maximum sequence length) and training with fixed batch sizes of randomly sampled sequences. A top model performance comparison of these 2 batching implementations is shown in **Supplementary Figure 1.**

S3: Ranking-error function: development and application

To optimize the parameters for the task of fitness score ranking, we were inspired to develop a new type of error function, E_r, defined as follows:

For a validation batch size of n, ranking-matrices $R \in \mathbb{R}^{n\times n}$ were constructed. For any two mutant variants in a given batch, $i, j \in B(n)$, the ranking-matrix elements were assigned binary values to represent whether the variant sequence of row i's fitness score was greater than, less than, or equal to the fitness score of the variant in column j.

$$a_{i,j} = \begin{cases} i > j : 1 \\ i \leq j : 0 \end{cases}$$

The purpose of allowing less-than and equal values to both be represented by $a_{i,j} = 0$ is two-fold: first, it was necessary, as some of the experimentally reported fitness scores had identical values for different mutants; second, in the Metropolis-Hastings selection steps, there is no need to accept a new mutation if it doesn't yield improvement over the current sequence. Thus, the ranking-matrix elements assigned values of $a_{i,j} = 1$ provide a simple way to represent the fitness rankings of variants, by showing which variants of columns j each variant of row i has fitness improvements over.

Two of these ranking-matrices were constructed: one for the experimentally-determined fitness scores of a given batch, Φ_E, and another for the model's predicted fitness scores for the input-representation of variants in that batch, Φ_P. A confusion matrix, Ψ, was produced from an inverse-truth comparison of the two ranking-matrices, such that it had values of 1 where the predicted fitness rankings were false with respect to experimental data, and 0 where the experimentally determined rankings were matched by predictions:

$$\Psi = \Phi_P \oplus \Phi_E$$

The sum of the elements in confusion matrix was computed, and normalized by the total number of elements, yielding our ranking-error function:

$$E_r = \frac{1}{n^2} \sum_{i,j} \psi_{i,j}$$

S4: Derivation of additive-type prediction criteria for linear kernel sequence representations:

Given the fitness score for a single amino acid substitution, $y_{i,u}$, where i denotes the backbone position of the mutation, and u denotes the new amino acid present, we can define the mutant fitness in terms of its difference relative to the wild type fitness, y_{wt}:

$$y_{(i,u)} = y_{wt} + \Delta y_{wt\to i,u}$$

where $\Delta y_{wt\to i,u} = y_{(i,u)} - y_{wt}$. Likewise, the predictive additive fitness of any of double-mutant, characterized by the substitution of amino acids u and v at positions i and j respectively, can be defined by:

$$\begin{aligned} \widehat{y}_{(i,u),(j,v)} &= y_{wt} + \Delta y_{wt\to i,u} + \Delta y_{wt\to j,v} \\ &= y_{(i,u)} + y_{(j,v)} - y_{wt} \end{aligned}$$

A one-hot encoding of a protein sequence composed on n amino acids, can be constructed through the concatenation of n consecutive sparse binary row-vectors ϕ_u, with u assigning the index of the vector element that is equal to 1, based on a defined ordering of the canonical amino acids:

$$\boldsymbol{s}_{wt} = \begin{bmatrix} \boldsymbol{\phi}_1 & \cdots & \boldsymbol{\phi}_i & \cdots & \boldsymbol{\phi}_n \end{bmatrix}$$

Consider a wild type amino acid, w, at position i,

$$\boldsymbol{\phi}_i(w) = \begin{pmatrix} \cdots & 0 & 1 & \cdots & 0 & 0 & \cdots \end{pmatrix}$$

and a mutant amino acid, u, to substitute at position i:

$$\boldsymbol{\phi}_i(u) = \begin{pmatrix} \cdots & 0 & 0 & \cdots & 1 & 0 & \cdots \end{pmatrix}$$

We can define an additive operator to represent the change in one-hot representation when going from the wild-type sequence to the mutant sequence with amino acid u substituted at position i:

$$\begin{aligned} \Delta \boldsymbol{s}_{wt \to (i,u)} &= \boldsymbol{s}_{(i,u)} - \boldsymbol{s}_{wt} \\ &= \begin{bmatrix} (\boldsymbol{\phi}_1(w) - \boldsymbol{\phi}_1(w)) & \cdots & (\boldsymbol{\phi}_i(u) - \boldsymbol{\phi}_i(w)) & \cdots & (\boldsymbol{\phi}_n(w) - \boldsymbol{\phi}_n(w)) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0} & \cdots & (\cdots & 0 & -1 & \cdots & 1 & 0 & \cdots) & \cdots & \mathbf{0} \end{bmatrix} \end{aligned}$$

Additionally, we can extend the use of this operator in the generation of one-hot sequence encodings for multiple amino acid substitutions. Again, considering a double-mutant with amino acids u and v substituted at positions i and j, a one-hot sequence representation can be produced as follows:

$$\begin{aligned} \boldsymbol{s}_{(i,u),(j,v)} &= \boldsymbol{s}_{wt} + \Delta \boldsymbol{s}_{wt \to (i,u)} + \Delta \boldsymbol{s}_{wt \to (j,v)} \\ &= \boldsymbol{s}_{(i,u)} + \boldsymbol{s}_{(j,v)} - \boldsymbol{s}_{wt} \end{aligned}$$

Given any linear model $F(s_{(i,u)}; \boldsymbol{c}, b)$, which maps a one-hot encoding $s_{(i,u)}$ to the corresponding fitness value $y_{(i,u)}$, a fitting operation will be performed to optimize the values of weights $\boldsymbol{c}$ and bias b, such that:

$$\begin{aligned} F(\boldsymbol{s}_{(i,u)}) &= \begin{bmatrix} \boldsymbol{s}_{(i,u)} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ b \end{bmatrix} \\ &= y_{(i,u)} + \sigma_{(i,u)} \end{aligned}$$

where $\sigma_{(i,u)}$ is the prediction error for a given mutation sample, which is to be minimized by fitting the model's parameters. In the fitting operation, training elements include a feature-array, $\boldsymbol{S}$, and a fitness-vector, $\boldsymbol{y}$, corresponding to an experimentally derived data set of single amino acid substitutions.

The predictive model here is a linear operator, and thus abides by the property of being closed under additivity:

$$\begin{aligned} F\left(\sum_{(I,U)} \boldsymbol{s}_{(i,u)} \right) &= \sum_{(I,U)} F(\boldsymbol{s}_{(i,u)}) \\ &= \sum_{(I,U)} \begin{bmatrix} \boldsymbol{s}_{(i,u)} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ b \end{bmatrix} \\ &= \sum_{(I,U)} y_{(i,u)} + \sigma_{(i,u)} \end{aligned}$$

Providing the linear predictive model with the one-hot encoding of a double-mutation for any two amino acids, u and v, substituted at positions i and j, we produce additive fitness scores, offset linearly by the error values of their corresponding single-mutations:

$$\begin{aligned} F(\boldsymbol{s}_{(i,u),(j,v)}) &= F\left(\boldsymbol{s}_{(i,u)} + \boldsymbol{s}_{(j,v)} - \boldsymbol{s}_{wt} \right) \\ &= F(\boldsymbol{s}_{(i,u)}) + F(\boldsymbol{s}_{(j,v)}) - F(\boldsymbol{s}_{wt}) \\ &= \left(y_{(i,u)} + \sigma_{(i,u)} \right) + \left(y_{(j,v)} + \sigma_{(j,v)} \right) - (y_{wt} + \sigma_{wt}) \\ &= (y_{wt} + \Delta y_{wt \to i,u} + \Delta y_{wt \to j,v}) + \left(\sigma_{(i,u)} + \sigma_{(j,v)} - \sigma_{wt} \right) \\ &= \widehat{y}_{(i,u),(j,v)} + \left(\sigma_{(i,u)} + \sigma_{(j,v)} - \sigma_{wt} \right) \end{aligned}$$

If $\boldsymbol{s}_{(i,u)}$,$\boldsymbol{s}_{(j,v)}$, and $\boldsymbol{s}_{wt}$ are all in the training set used for fitting the parameters of the predictive model, and if the fitting occurs within an over-determined system where the number of samples is less than the number of elements in each one-hot representation ($n \times 20$), then the regression will fit closely to all training data, such that the values of $\sigma_{(i,u)}$, $\sigma_{(j,v)}$, and σ_{wt} are negligible.

In such a case, we find the predicted value of the one-hot encoding of a double-mutant to be:

$$F(\boldsymbol{s}_{(i,u),(j,v)}) \simeq \widehat{y}_{(i,u),(j,v)}$$

Therefore, the fitness value for a double-mutant, as predicted by a linear model that is unconstrained in fitting to a training set that includes the constituent single-mutations, is approximately equal to the additive fitness scores for that double-mutant.

Supplementary Figures:

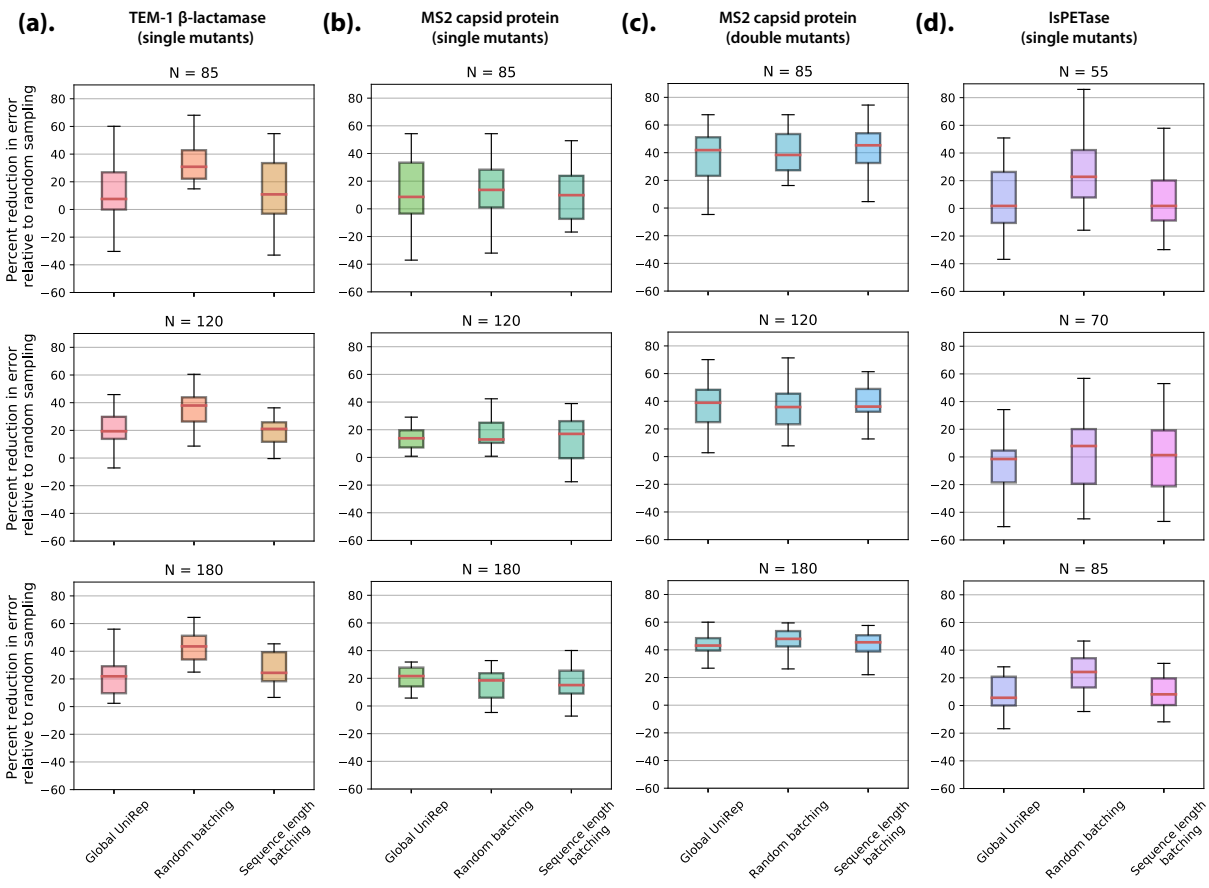

Figure 1: Performance comparison of initial (global) UniRep to evotuned representations trained using random batching and fixed-length batch of amino acid sequences. The percent reduction in ranking error for fixed-length evotuned representations (averaged over three different batch sizes, and with 20 replicate trials for each batch size) are: **(a).** TEM-1 β-lactamase: 19.4 ± 5.89%. **(b).** MS2 single-mutants: 14.3 ± 2.81%. **(c).** MS2 double-mutants: 40.9 ± 3.27%. **(d).** IsPETase: 4.77 ± 3.90%. The fixed-length evotuned representations were produced using the same learning rate and epoch as their random-batched counterparts. Percent ranking-error reduction over random sampling for Global UniRep and randomly-batched evotuned representations are reported and shown in **Figure 3** of the main text.

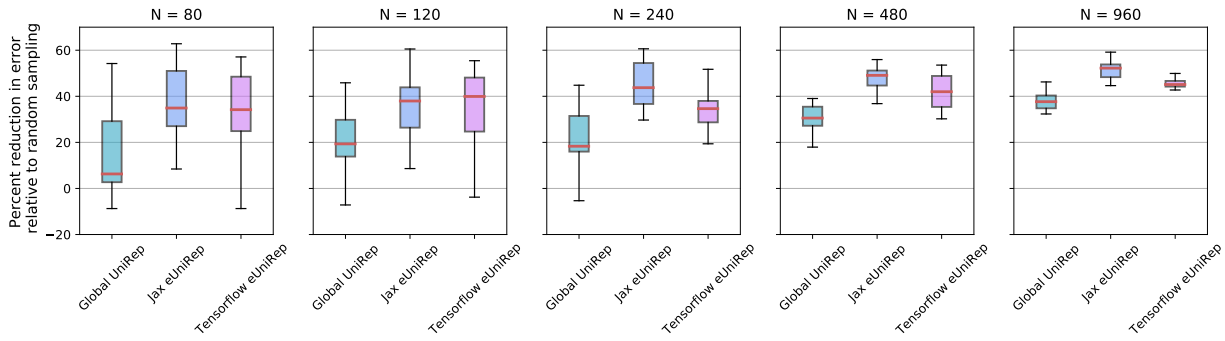

Figure 2: Comparison of percent reduction in ranking-error relative to random sampling when training a top-model (ridge regression, $\alpha = 10^{-2}$) for three sequence representations of TEM-1 β-lactamase: the initial Global UniRep features, globally-initiated evotuned representations using the Jax-based evotuning implementation, and original Tensorflow-based evotuned weights provided by Baswas et al. Across these batch sizes, reduction in error were: Global UniRep = 25.4 ± 7.83, Jax eUniRep = 43.0 ± 6.29, Tensorflow eUniRep = 37.9 ± 4.80.

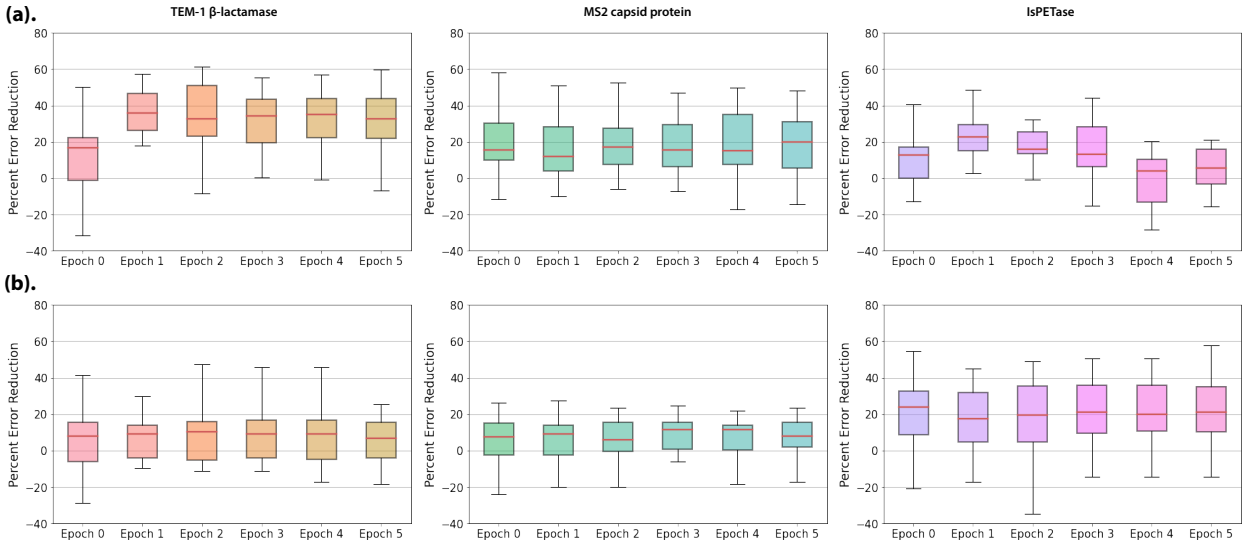

Figure 3: Percent reduction in ranking error relative to random sampling for single-mutant datasets of TEM-1 β-lactamase, MS2 capsid protein, and IsPETase, over the course of five epochs of evolutionary fine-tuning. Percent error reduction in predicting the relative fitness rankings of test-set mutants, relative to random sampling, compared across three different batch sizes, and 20 randomized trials **(a).** Evotuning with parameter initiation using globally trained UniRep weights. **(b).** Local evotuning, using random-value parameter initiation. All conditions were evaluated using total batch sizes of 85 randomly selected samples; training and test sets from each batch were selected using a 80:20 percent batch-size split.

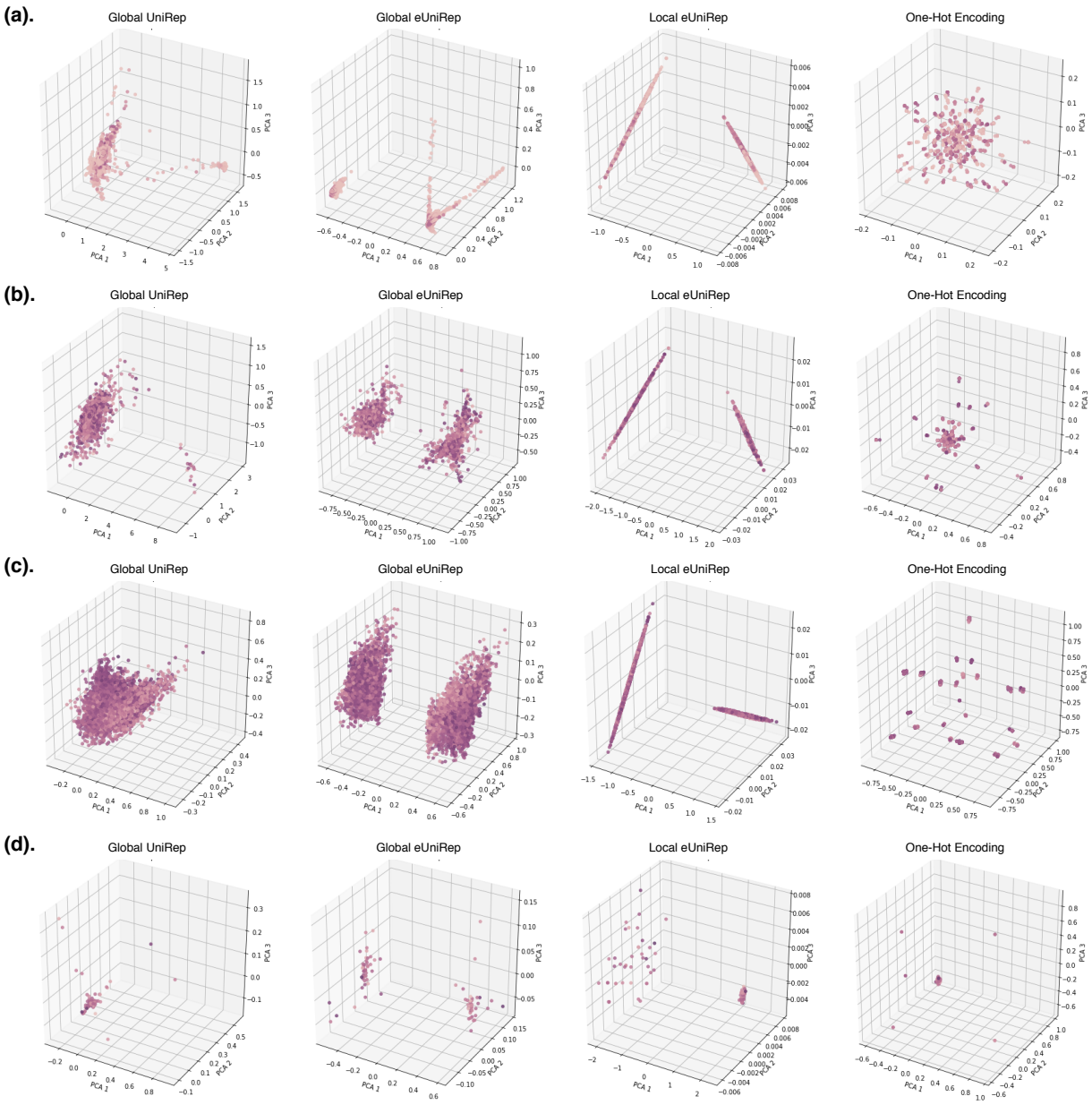

Figure 4: The first 3 principal components of 4 protein mutant datasets with 3 representations: Global UniRep, Global eUniRep, Local eUniRep, and one-hot encodings. (a). TEM-1 β-lactamase, (b) MS2 single mutants, (c). MS2 double mutants, (d). IsPETase.

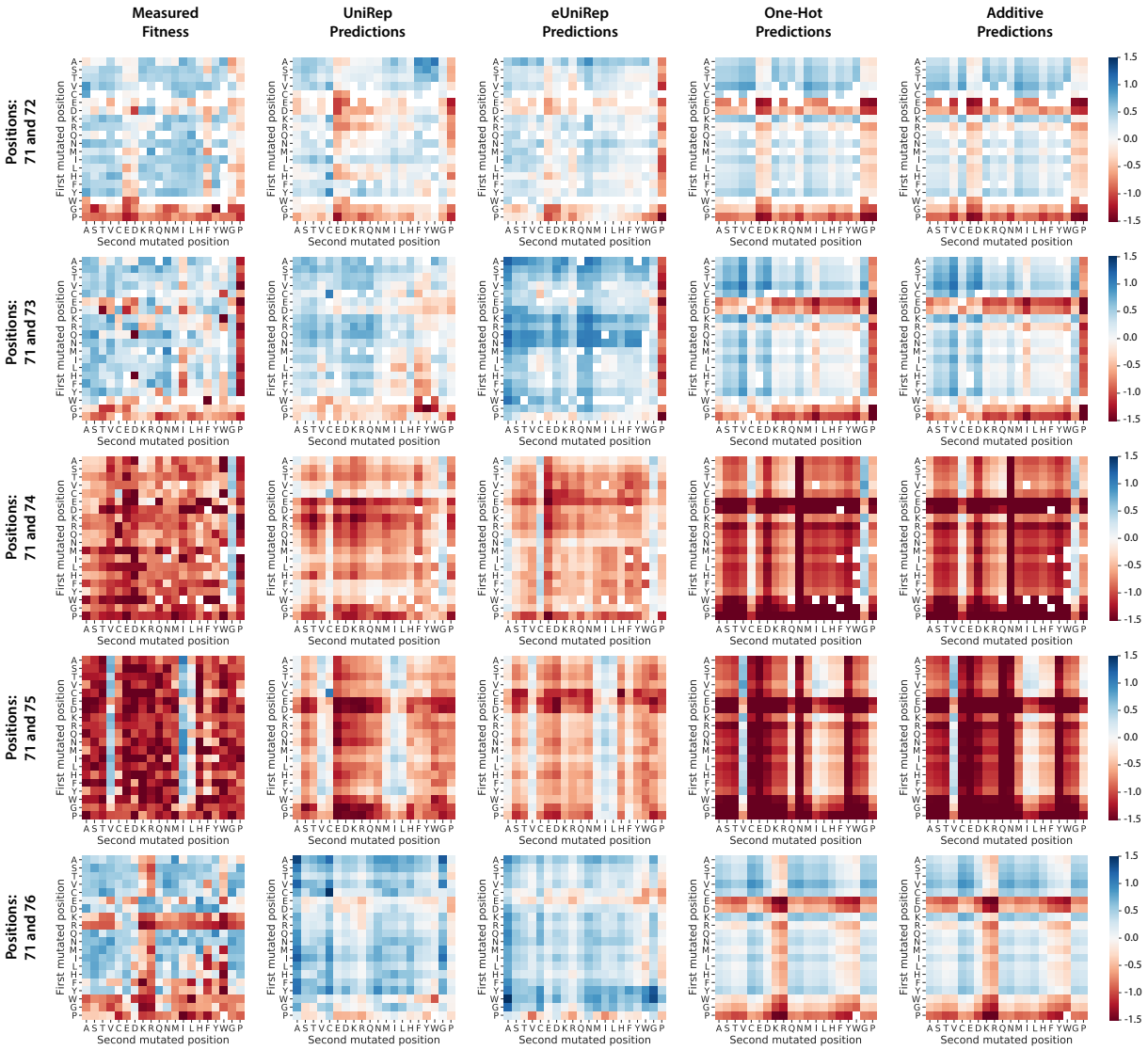

Figure 5: The predictions of MS2 double-mutant fitness, performed by a Ridge regression that was trained on MS2 single-mutant data. Rows correspond to all possible co-mutation combinations of amino acids at position 71 with positions 72, 73, 74, 75 and 76. Column 1 displays the experimentally determined fitness values. Columns 2-5 correspond to different predictive methods tested.

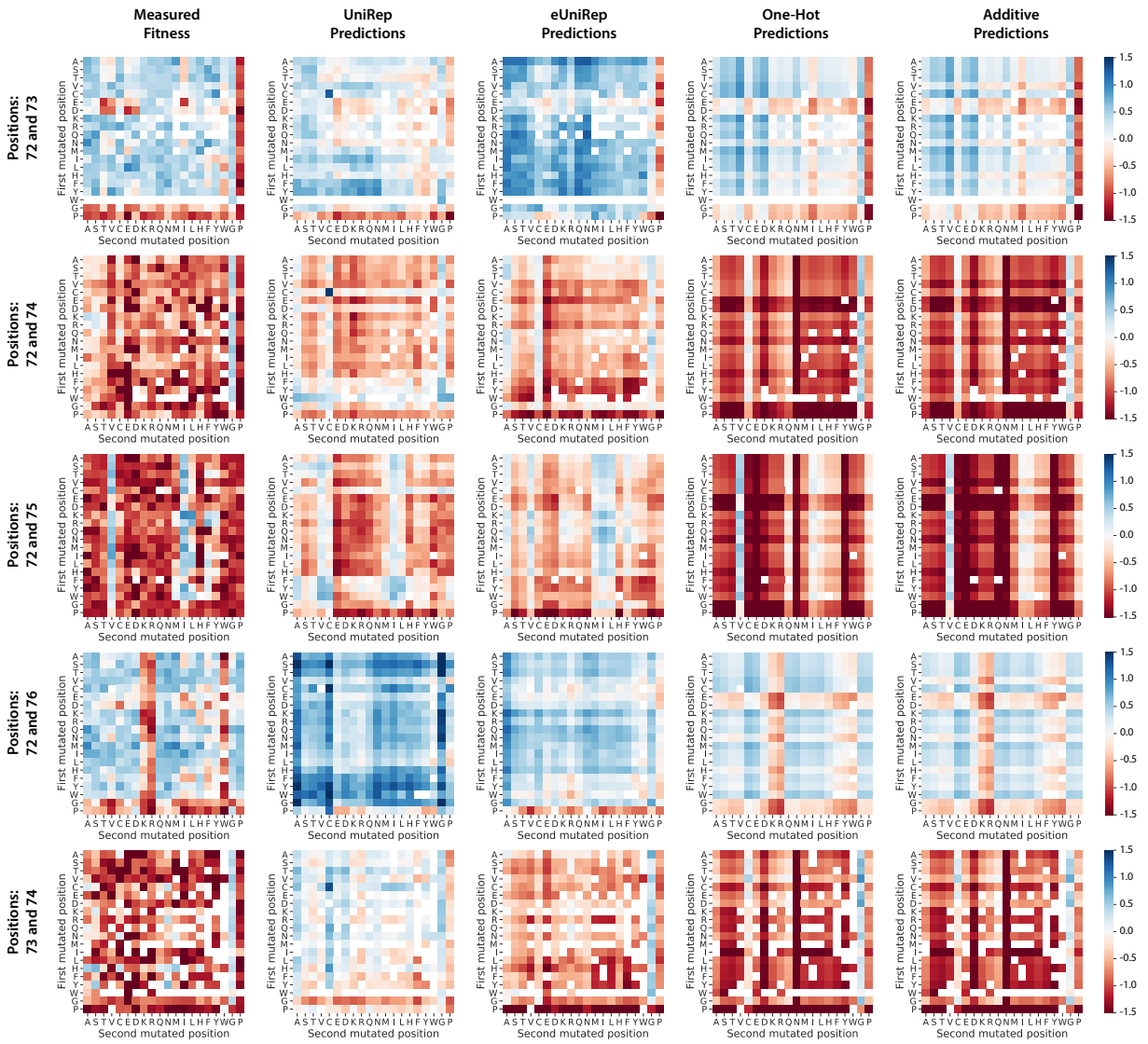

Figure 6: The predictions of MS2 double-mutant fitness, performed by a Ridge regression that was trained on MS2 single-mutant data. Rows correspond to all possible co-mutation combinations of amino acids at position 72 with positions 73, 74, 75 and 76, and position 73 with position 74. Column 1 displays the experimentally determined fitness values. Columns 2-5 correspond to different predictive methods tested.

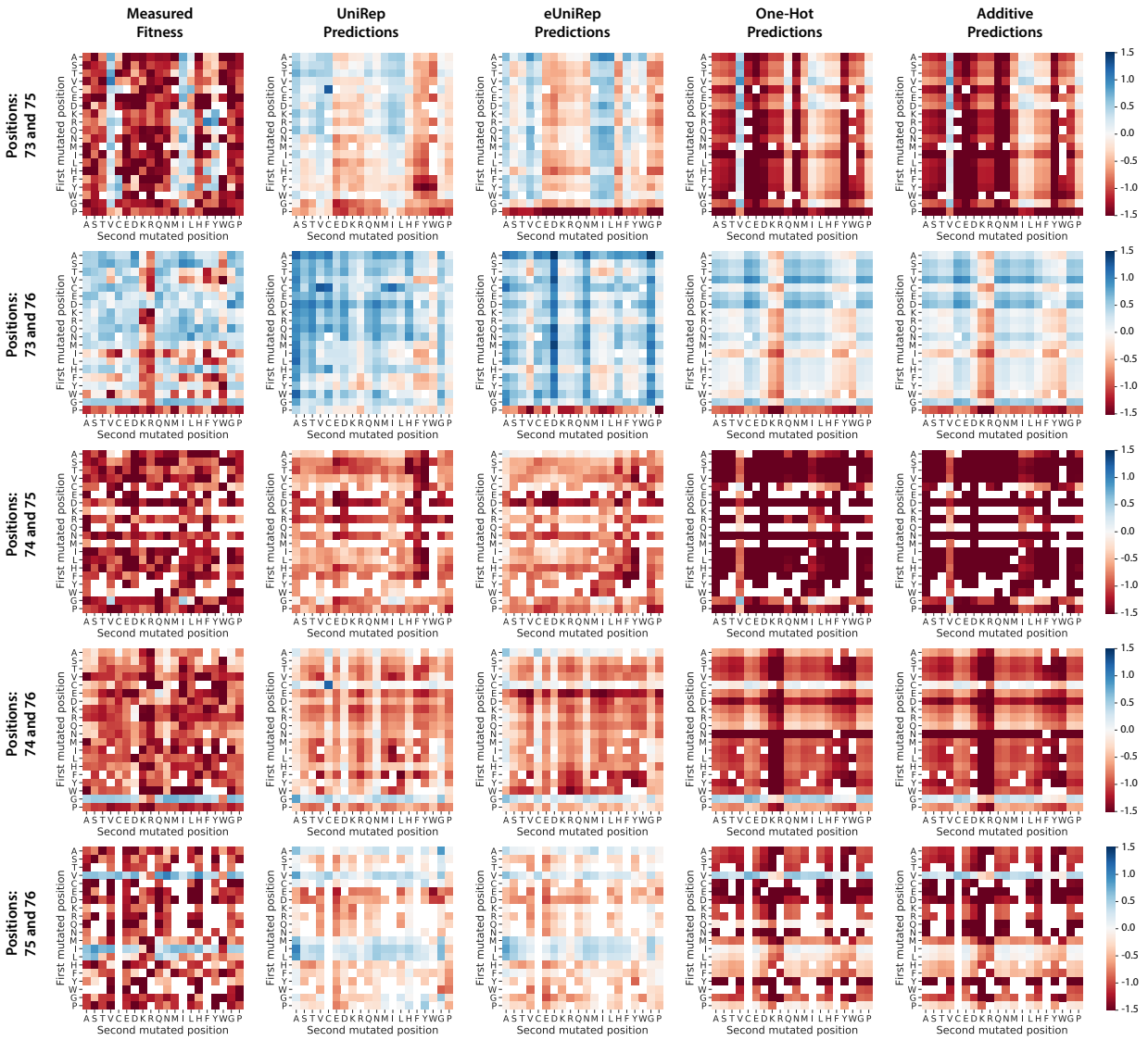

Figure 7: The predictions of MS2 double-mutant fitness, performed by a Ridge regression that was trained on MS2 single-mutant data. Rows correspond to all possible co-mutation combinations of amino acids at position 73 with positions 75 and 76, position 74 with positions 75 and 76, and position 75 with position 76. Column 1 displays the experimentally determined fitness values. Columns 2-5 correspond to different predictive methods tested.