FLIGHTED: Inferring Fitness Landscapes from Noisy High-Throughput Experimental Data

Vikram Sundar[1*], Boqiang Tu[2], Lindsey Guan[1], Kevin Esvelt[3*]

[1]Computational and Systems Biology Program, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, 02139, MA, USA.
[2]Department of Biological Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, 02139, MA, USA.
[3]Media Lab, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, 02139, MA, USA.

*Corresponding author(s). E-mail(s): vsundar@mit.edu; kesvelt@mit.edu;
Contributing authors: btu@mit.edu; lguan@mit.edu;

Abstract

Machine learning (ML) for protein design requires large protein fitness datasets generated by high-throughput experiments for training, fine-tuning, and benchmarking models. However, most models do not account for experimental noise inherent in these datasets, harming model performance and changing model rankings in benchmarking studies. Here, we develop FLIGHTED, a Bayesian method for generating fitness landscapes with calibrated errors from noisy high-throughput experimental data. We apply FLIGHTED to single-step selection assays such as phage display and to a novel high-throughput assay DHARMA that ties fitness to base editing activity. Our results show that FLIGHTED robustly generates fitness landscapes with accurate errors. We demonstrate that FLIGHTED improves model performance and enables the generation of protein fitness datasets of up to $\mathbf{10^6}$ variants with DHARMA. FLIGHTED can be used on any high-throughput assay and makes it easy for ML scientists to account for experimental noise when modeling protein fitness.

Keywords: Machine learning, Bayesian inference, protein design, fitness landscape

1 Introduction

Machine learning (ML) approaches have been remarkably successful at solving a variety of protein design problems [1, 2]. Many of these approaches rely on data generated by high-throughput experiments; this data is used as training or fine-tuning data for a large ML model that optimizes the desired protein function [1, 3–5]. Since ML models become more powerful when trained on larger datasets, the generation of large protein fitness datasets from high-throughput experiments is essential for success. Protein fitness measurements from high-throughput experiments are also broadly used for benchmarking and *in silico* retrospective model performance evaluations [6–10].

However, high-throughput experiments are inherently noisy. For example, single-step selection experiments such as deep mutation scanning (DMS) exhibit substantial noise in the measured enrichment ratio [11–14]. Despite warnings in the literature, most ML protein modeling efforts completely ignore any experimental noise present in large fitness datasets. Using noisy datasets for training, benchmarking, and evaluating models can lead to inaccurate conclusions and subpar performance, as ML models train to noise instead of signal in a given dataset.

Past work on incorporating experimental noise into protein fitness modeling has primarily focused on DMS experiments. Some studies have attempted fully analytic solutions to this problem, which generally limits the downstream modeling to linear regression [11] or makes assumptions about the experimental parameters [12]. Other related work has focused on mitigating sequencing noise, optimal library design, and multiple rounds of selection, which we do not explicitly model here [13–15].

We aim to develop a paradigm that can be generally applied to any high-throughput experiment and any ML model. In this paper, we present FLIGHTED (Fitness Landscape Inference Generated by High-Throughput Experimental Data), a new approach based on Bayesian modeling to generate reliable fitness landscapes with calibrated errors from noisy high-throughput experiments. FLIGHTED is applicable to any high-throughput experiment and an arbitrarily complex ML model. We apply FLIGHTED to single-step selection experiments and a new assay DHARMA; in both cases, it generates fitness landscapes with robust, calibrated errors. We then generate two FLIGHTED-calibrated landscapes on the GB1 protein and TEV protease and benchmark a number of standard ML protein fitness models on these landscapes to learn how to improve protein fitness prediction.

2 Results

2.1 Training a FLIGHTED Model

FLIGHTED applies Bayesian modeling to the problem of inferring fitness landscapes from noisy high-throughput experimental data. Practitioners must train a version of FLIGHTED for each class of high-throughput experiment with similar sources of noise; in this paper, we train two versions of FLIGHTED for single-step selection experiments and for DHARMA, a molecular recording assay that links fitness to base editing mutations in a canvas [16]. For each class of high-throughput experiment, FLIGHTED requires a calibration dataset of noisy experimental results mapped to

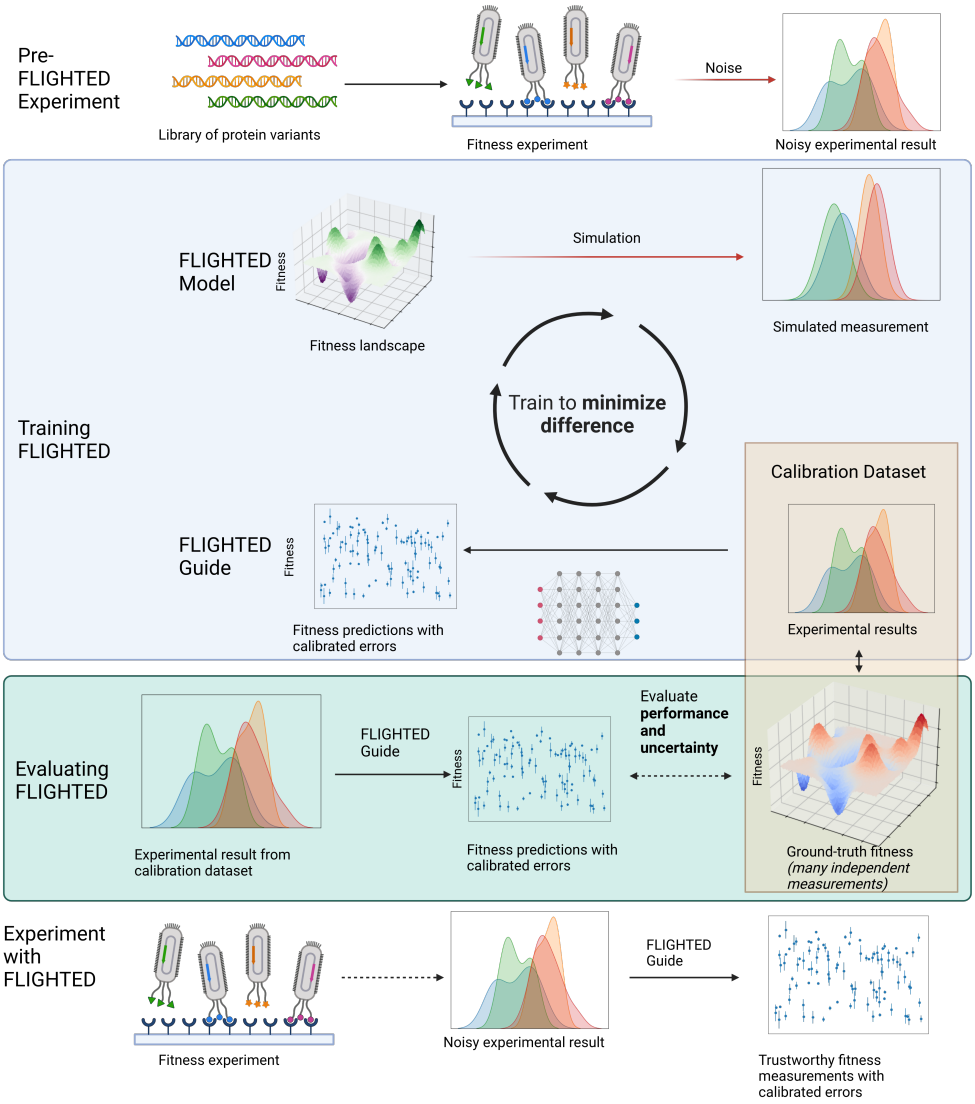

Fig. 1 Summary of the FLIGHTED Approach to Incorporating Experimental Noise from High-Throughput Experiments. Top: A high-throughput experiment without FLIGHTED generates a readout that measures fitness and incorporates considerable noise. Middle: to train FLIGHTED, we simulate the noise generation process of a given experiment in the model. We then use a guide (neural network) to convert experimental readouts to fitness predictions with calibrated errors, and train to minimize the difference, i.e. the ELBO loss between the two distributions. Training does not require ground-truth fitnesses. Middle-bottom: the FLIGHTED guide is evaluated using ground-truth fitnesses from a calibration dataset for predictions and uncertainty. Bottom: once validated, the FLIGHTED guide can be used to denoise other high-throughput experiments.

ground-truth fitnesses; these ground-truth fitnesses can be obtained by simulation or by many repeated experimental measurements. Training FLIGHTED only uses a selection of the noisy experimental results, ignoring the fitnesses, and generates two components: a FLIGHTED model, which simulates experimental results from a fitness landscape, and a FLIGHTED guide, which predicts fitnesses from experimental results (see Figure 1, middle). To evaluate FLIGHTED, practitioners use the ground-truth fitnesses from the calibration dataset to measure performance of the FLIGHTED guide. Importantly, FLIGHTED is trained in a fully unsupervised fashion using only noisy experimental results from a calibration dataset; ground-truth fitnesses are only used for evaluation.

We now describe the mathematical framework that justifies FLIGHTED. Abstractly, a high-throughput experiment generates a library of protein variants that map onto a fitness landscape defined by their molecular activity under a certain set of conditions, but the noisy experimental process imperfectly samples from the actual landscape, generating a noisy experimental result (see Figure 1, top). We can write down a probabilistic graphical model for this noise generation process using our biological knowledge of the experiment. That is, we generate an mathematical model for each source of noise, then simulate experimental results for a given fitness landscape. This constitutes the FLIGHTED model for a given experiment (see Figure 1, middle); any sources of noise not modeled in this process will not be accounted for by FLIGHTED. The use of this probabilistic graphical model explains why learning from only experimental readouts – and not ground-truth fitness – is possible: the biological knowledge used to build the model constrains the space of possible models.

Because inverting an arbitrary noise generation model to determine fitness is generally infeasible, we use stochastic variational inference (SVI) to generate a FLIGHTED guide, or variational distribution, that predicts fitness with error from experimental results. Specifically, the guide approximates the posterior distribution of fitness as a normal distribution. We then train our model and guide simultaneously to minimize the difference between predicted fitness landscapes and experimental results on the calibration dataset. Using SVI has two main advantages. First, model parameters can be used for any number of subsequent experimental measurements after being determined once on a calibration dataset; second, all models can have arbitrary complexity (including neural networks).

To evaluate FLIGHTED's performance (see Figure 1, middle-bottom), we measure both performance and uncertainty prediction of the FLIGHTED guide's predicted fitnesses and errors. The FLIGHTED guide can then be used to denoise other experimental results generated from the same high-throughput experimental assay (see Figure 1, bottom).

The procedure we have outlined allows for the generation of trustworthy fitness measurements with calibrated errors solely from using the FLIGHTED guide on a given experiment. While we demonstrate just two applications of FLIGHTED, it can be easily generalized to other assays; the only arbitrary step in the procedure described above is selection of neural network architecture for the FLIGHTED guide.

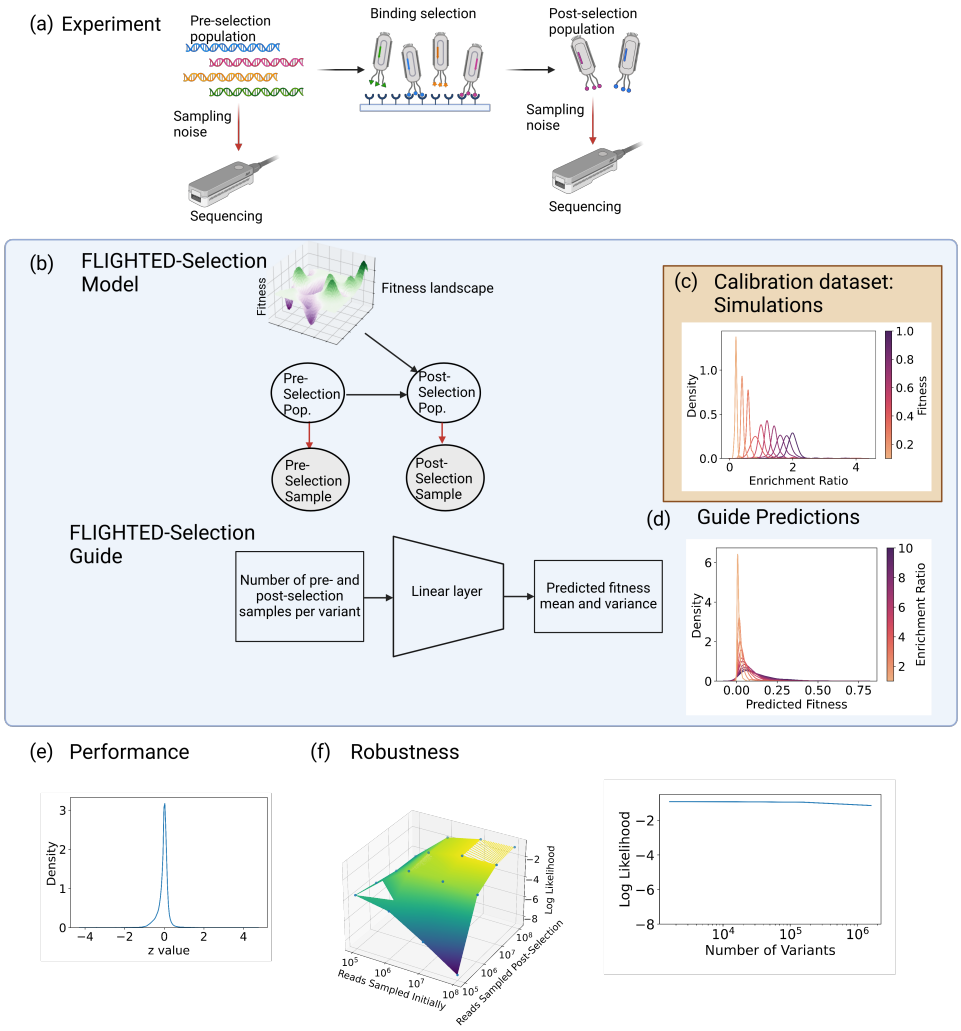

Fig. 2 FLIGHTED-Selection Model Performance. (a) Single-step selection assays always include sampling noise when reads are sampled for sequencing. (b) The probabilistic graphical model used for FLIGHTED-Selection and model architecture for the guide. (c) Simulations indicate that the enrichment ratio has considerable noise in a single-step selection experiment due to sampling. (d) Typical FLIGHTED-Selection predictions for a given enrichment ratio. (e) FLIGHTED-Selection model predictions are well-calibrated. (f) FLIGHTED-Selection model performance is robust to changes in number of reads selected pre- and post-selection if there are more reads than variants (20^4 here). Model performance is also robust to the number of variants.

2.2 FLIGHTED-Selection

We now turn to applications of FLIGHTED to two types of experimental assays: single-step selection assays and DHARMA assays. Single-step enrichment-based selection

assays are experimental assays in which a library of variants is produced, a selection step occurs to select fitter variants, and the remaining selected population is measured; see Figure 2a [17–19]. These include mRNA display, phage display, and many deep mutational scan (DMS) studies, and comprise a substantial portion of fitness landscapes used for ML [6, 20]. Fitness is usually measured as the enrichment ratio, or the ratio of variant sequences sampled post- to pre-selection [17]. However, the enrichment ratio is an inherently noisy measurement due to sampling noise, the noise associated with sampling the reads to be sequenced from a larger library [17]. Sampling noise is unavoidable in a single-step selection assay and the only source of noise we choose to model in FLIGHTED [17].

To understand the impacts of sampling noise and generate calibration datasets for downstream training, we ran simulations of a single-step selection assay on a randomly generated fitness landscape with 20^4 variants. Figure 2c and Extended Data Figure A1a plot the probability distributions of the enrichment ratio for a given fitness, showing there is considerable noise and that this noise increases with increasing fitness.

To directly show the extent of the impact on downstream ML, we measure the Pearson r between fitness and enrichment ratio. Extended Data Figure A1c demonstrates that the Pearson r for the roughly 1000 variants with highest enrichment ratio is roughly 0, indicating single-step selection experiments offer no useful information about variants with the highest fitness. Many papers [20] filter their data by using a minimum number of reads to reduce noise. We simulate this practice in Extended Data Figure A1c and find no substantial change. This is particularly problematic for downstream ML applications attempting to select variants with high fitness, since the most useful data points for this task are also the noisiest. Therefore, enrichment ratio is not a reliable measure of fitness.

The probabilistic graphical model for FLIGHTED-Selection is shown in Figure 2b. Given a fitness landscape and a pre-selection population, we randomly sample and sequence pre-selection, select a post-selection population based on the provided fitnesses, and randomly sample and sequence post-selection. The guide model as shown in Figure 2b uses the logit of the enrichment ratio as the mean and predicts the variance using linear regression on the number of pre- and post-selection samples. FLIGHTED-Selection is trained on a calibration dataset of simulations on a randomly generated landscape. We use simulations as the training data since we need to know the ground-truth fitness to accurately evaluate FLIGHTED-Selection's performance; there is no easy way to measure the ground-truth fitness for any given real-world single-step selection experiment, and alternative measurement assays are generally not high-throughput and consequently fail to yield sufficient data. Typical model predictions are shown in Figure 2d for given enrichment ratios; for more active variants, single-step selection experiments produce less reliable fitness measurements and FLIGHTED-Selection assigns higher variance, as expected.

To evaluate model performance, we focused on calibration, since the enrichment ratio is the optimal fitness value. We compute the z-value of the true fitness compared to the model prediction in Figure 2e; the resulting distribution looks very similar to a normal distribution, suggesting that FLIGHTED-Selection is well-calibrated. The mean log likelihood of our predictions is -0.97, very close to that expected for a normal distribution.

Next, we evaluated the robustness of FLIGHTED-Selection as a function of various parameters of the data generation process. In Figure 2f we showed the log likelihood as a function of number of reads sampled pre- and post-selection and demonstrated that the model was very reliable as long as the number of reads sampled was greater than the number of variants (at 20^4). Next, we fix the number of reads sampled pre- and post-selection and evaluate robustness as a function of number of variants. In Figure 2f, we see that the log likelihood of FLIGHTED-Selection does not substantially worsen as a function of the number of variants. We also experimented with varying different parameters of the random distributions in the simulation to see whether that affected FLIGHTED-Selection model performance. In Extended Data Figure A1b, we see that these factors generally have very little effect on robustness.

2.3 FLIGHTED-DHARMA

We next turn to DHARMA (Direct High-throughput Activity Recording and Measurement Assay), a recently developed high-throughput protein fitness assay [16]. DHARMA measures fitness by linking protein fitness to transcription of a base editor; this base editor is targeted to a canvas where it randomly causes C→T edits, as shown in Figure 3a. Higher fitness leads to higher base editor transcription and more C→T edits.

DHARMA can be run in high-throughput on up to 10^6 variants at once, since the readout is measured by cheap long-read sequencing. It can measure any protein function that can be linked to base editor activity, which includes protease activity, gene editing, and protein and DNA binding [21]. However, base editor production and activity are inherently stochastic processes [22] so DHARMA output is noisy; Figure 3c demonstrates that repeated DHARMA experiments on the same variant can result in differing C→T edit counts. Therefore accounting for noise via an ML model like FLIGHTED is essential for reliable fitness measurements.

The probabilistic graphical model for FLIGHTED-DHARMA is shown in Figure 3b. Given a fitness landscape, we compute a base editing probability for each base in the canvas as a logistic function of fitness and then sample to determine whether each base was mutated. This model treats all canvas bases as independent but allows each base to have a different learned edit propensity. The guide predicts both the mean and variance of fitness given a single DHARMA read using a two-layer feedforward neural network. We generated a calibration dataset by running a DHARMA experiment on a 3-site library of T7 polymerase. Concretely, in Figure 3d we show the predicted number of C→T edits with error given a particular fitness.

We now assess performance and uncertainty of the trained FLIGHTED-DHARMA model. "Ground-truth" fitness measurements were produced by Fluorescence-Activated Cell Sorting (FACS) on a subset of 119 T7 variants, in which T7 polymerase drives the expression of GFP instead of base editor. Error was minimized in these FACS measurements by using a large number of cells per variant. To eliminate data leakage, FLIGHTED-DHARMA was not trained on any DHARMA read from any variant for which FACS was performed. In order to assess model performance using the FACS dataset, we must convert fitnesses predicted by FLIGHTED-DHARMA to the FACS readout, which could be an arbitrary nondecreasing function. We used a validation set

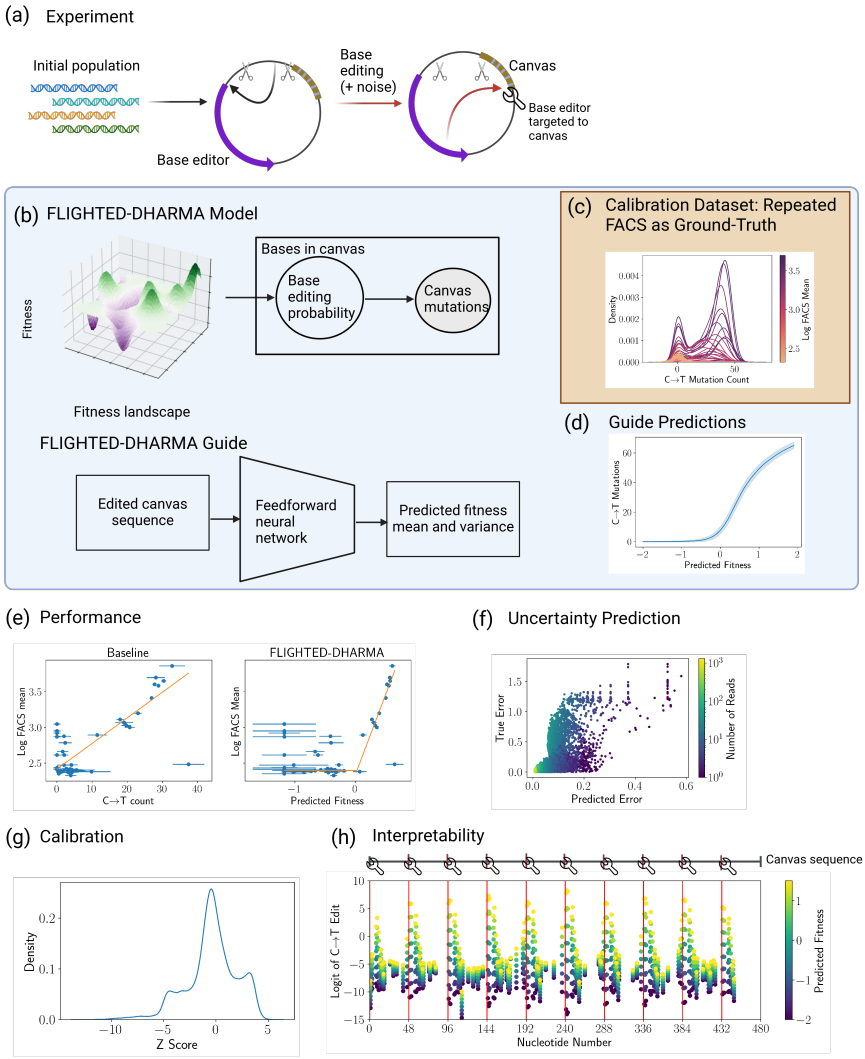

Fig. 3 FLIGHTED-DHARMA Model Performance. (a) DHARMA links fitness to edit rates of a DNA segment (canvas) via transcriptional control of a base editor, an inherently noisy process. (b) The probabilistic graphical model for FLIGHTED-DHARMA and the architecture for the FLIGHTED-DHARMA guide. (c) Noise in DHARMA as a function of true fitness. (d) FLIGHTED-DHARMA's prediction of number of C→T mutations as a function of fitness. (e) FLIGHTED-DHARMA improves performance of fitness predictions when compared to a baseline model that predicts number of C→T mutations. (f) FLIGHTED-DHARMA errors correlated with true error on even small subsets of DHARMA reads. (g) FLIGHTED-DHARMA is reasonably well-calibrated. (h) Canvas nucleotides most likely to be edited according to FLIGHTED-DHARMA are found at guide RNA binding sites, as expected.

to fit this to a piecewise linear function as shown in Extended Data Figure A2a. The initial flat section of the piecewise linear functions corresponds to low-activity variants where background fluorescence dominates the FACS measurement. We then evaluated model performance on the remaining test set in Figure 3c. FLIGHTED-DHARMA was compared to a baseline consisting of the mean and variance of C→T mutation count. We found that FLIGHTED-DHARMA's MSE was 0.72, an improvement over the baseline MSE of 0.78. Most of the improvement is observed on the portion of the fitness landscape where both DHARMA and FACS are measuring fitness.

We also examined uncertainty by evaluating the calibration of our predicted variances. First, we directly examine calibration on the test data, using all reads available for each datapoint in the test set in Extended Data Figure A2b. FLIGHTED-DHARMA's predicted errors increase as true error increases. The log likelihood of the baseline model on this dataset was -19.8 while the log likelihood of FLIGHTED-DHARMA was -10.0. In Figure 3d, we selected random subsets of DHARMA reads and measured the true and predicted error of FLIGHTED-DHARMA. Random subsampling tests our ability to predict error given even very small subsets of DHARMA sequencing reads. The mean log likelihood here was -3.93, suggesting that FLIGHTED-DHARMA is slightly overconfident but reasonably well-calibrated. Baseline model performance was substantially worse with a mean log likelihood of -8.00 (see Extended Data Figure A2c). Since our calibration tests included small subsets of DHARMA reads, we can be confident that FLIGHTED-DHARMA will produce reasonable errors even when given few DHARMA reads. This aids experimentalists by informing them whether a given number of DHARMA reads is sufficient for a fitness measurement.

Finally, since FLIGHTED uses a biological model of DHARMA, we can examine the parameters of that biological model directly. In Figure 3g, we compute the logit of the C→T edit probability at a given canvas residue as a function of fitness. The probability of a C→T edit increases and is more fitness-dependent every 48 residues. The base editor is targeted at these locations, so these results are expected. This is an example of how FLIGHTED can yield an interpretable model.

2.4 Benchmarking Landscapes on FLIGHTED Data

Having established that FLIGHTED can generate fitness landscapes with reliable, calibrated errors from single-step selection and DHARMA experiments, we now turn to benchmarking ML models on the resulting landscapes. We focus on two landscapes: a 4-site landscape for the GB1 protein that measures stability and binding to IgG via an mRNA display single-step selection assay [20], and a 4-site landscape on TEV protease that measures protease activity on the wild-type substrate using a DHARMA assay. The first landscape has frequently been used to benchmark ML models in the field [6]; we release the second landscape publicly as a source for the ML community. Basic statistics about the TEV landscape may be found in Extended Data Figures A3b - f.

The models we chose to benchmark are outlined in Figure 4a. Specifically, given a sequence, we generate an embedding from a protein language model or from a one-hot baseline [23, 24]. We then feed this embedding into a task-specific top model –

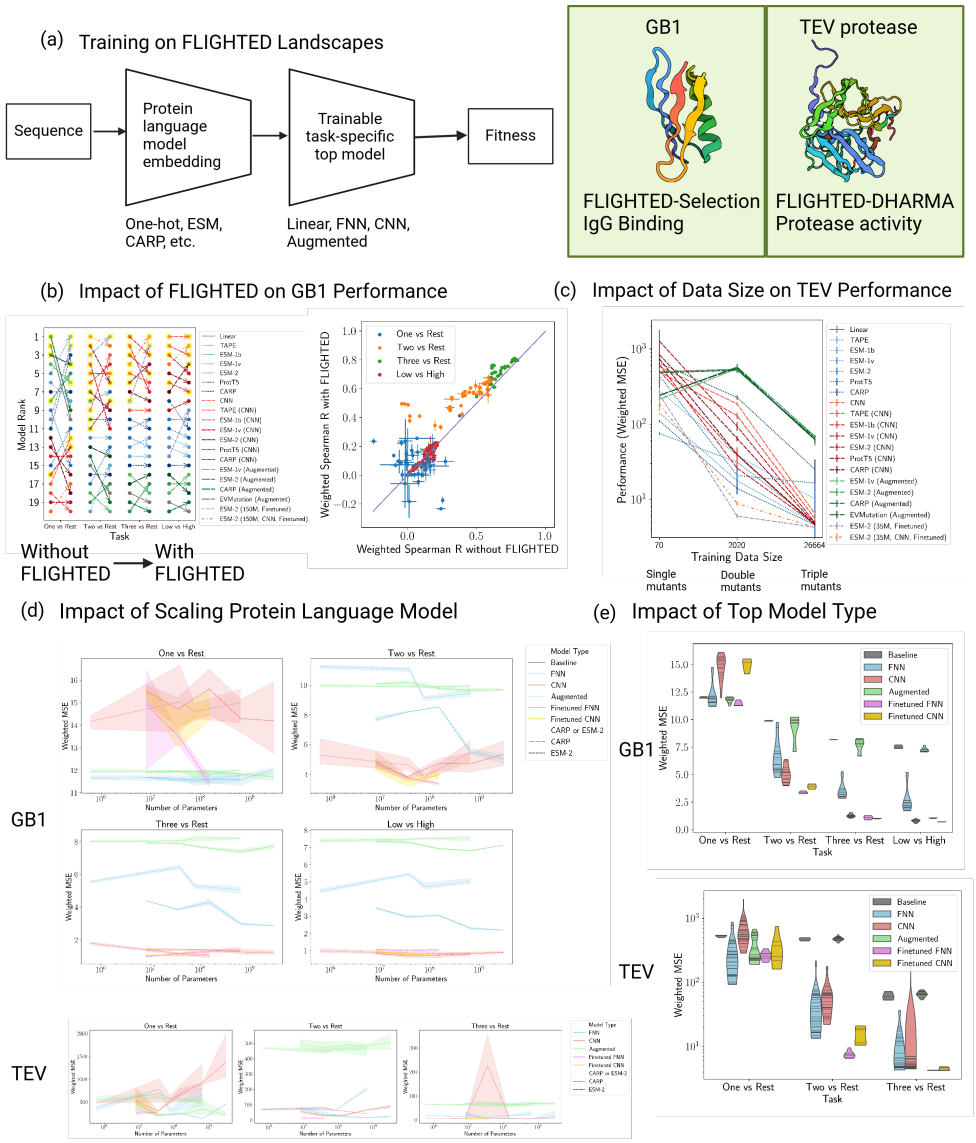

Fig. 4 Benchmarking Protein Fitness Models on FLIGHTED Landscapes. (a) The typical model architecture we tested and the two landscapes: GB1, based on FLIGHTED-Selection [20], and TEV protease, based on FLIGHTED-DHARMA. (b) FLIGHTED changes model ranking on the GB1 dataset, indicating the importance of denoising when benchmarking ML models. FLIGHTED also modestly improves model performance on the GB1 dataset. (c) Exponential increases in data on the TEV dataset result in exponential increases in accuracy. (d) Scaling up protein language models does not substantially improve model performance. (e) Convolutional neural networks (CNNs) and finetuned models are the highest-performing top models across both the GB1 and TEV datasets.

either a linear layer, an augmented model [25], a feedforward neural network (FNN), a convolutional neural network (CNN), or a finetuned FNN or CNN – which is trained on this particular landscape. This strategy of blending a protein language model with a task-specific top model has repeatedly proven successful [5, 6, 23, 24, 26]. While these models are certainly not novel, to our knowledge this is the first systematic exploration of the use of CNNs as a top model.

To incorporate errors generated by FLIGHTED, we train and evaluate models using the inverse-variance-weighted mean squared error (MSE) as a proxy for log likelihood. We use 4 data splits on the GB1 dataset as proposed previously by the FLIP benchmark [6]: one-vs-rest, two-vs-rest, and three-vs-rest splits with only single, double, and triple mutants respectively in the training set, and a low-vs-high split with only inactive variants in the training set. We prepare similar one-vs-rest, two-vs-rest, and three-vs-rest splits of our own TEV protease dataset. See Extended Data Tables A1 and A2 for raw GB1 model performance and Table A3 for raw TEV model performance.

We first examine the impact of using FLIGHTED on models trained on the GB1 dataset in Figure 4b. The ladder plot (Figure 4b, left) shows that FLIGHTED significantly affects benchmarking outcomes. This is particularly important because of how frequently ML practitioners use single-step selection datasets to benchmark their models; without use of a method like FLIGHTED, the results of this benchmarking are unreliable.

Further, we also observe a general performance improvement upon using FLIGHTED data for the training set (see Figure 4b, right). Here performance is measured using weighted Spearman ρ so results with and without FLIGHTED can be directly compared; the weighted Spearman ρ accounts for variance on the test set [27]. We present correlations with FLIGHTED on the test set here; see Extended Data Figure A3a for results without FLIGHTED on the test set. 37 of the 45 models on the two-vs-rest split and 41 of the 45 models on the three-vs-rest split had statistically significant improvements. We saw an average improvement of 0.15 on the two-vs-rest split and 0.05 on the three-vs-rest split. Results on the other two splits were not as consistent. These results indicate that using FLIGHTED will generally improve model performance, though this is not guaranteed.

Next, we examine the impact of various other parameters on model performance. In Figure 4c we look at the importance of data size on the TEV dataset, controlling the test set to include only quadruple mutants. As training data increases from single to double to triple mutants, model error decreases exponentially. This dramatic improvement in model accuracy indicates an increasing ability to generalize and highlights the importance of high-throughput experimental methods: with more data, our models become substantially more powerful.

We then examined which model architecture choices mattered most for performance. We found in Figure 4e that the top model was the most impactful, with all models equally unable to learn from single mutants, and CNNs performing better on larger datasets, especially three-vs-rest. Fine-tuning generally improves performance, especially for FNNs on small datasets, but is substantially more computationally

expensive. Since CNNs performed well as a top model and are relatively computationally efficient compared to fine-tuning, we recommend practitioners use CNNs as a top model when they have large enough protein fitness datasets.

More surprisingly, increasing the number of parameters in the protein language model did not have a large impact on performance for most models. In Figure 4d, we ran published variants of CARP and ESM-2 with smaller numbers of parameters [24, 28]. We found no substantial or consistent performance improvement, across models ranging from millions to billions of parameters, aside from fine-tuning FNNs on small datasets. Our results suggest that scaling up protein language models will not improve performance in predicting protein fitness given current data limitations. This finding has recently been corroborated [29].

3 Discussion

FLIGHTED has consistently generated fitness landscapes with robust, calibrated errors for two very different high-throughput experiments. Since it is based on Bayesian modeling and variational inference, it can be easily extended to other high-throughput experiments or more complicated noise generation models of single-step selection by taking into account effects such as biased library design or sequencing error. We have demonstrated that FLIGHTED models can be used reliably under a wide range of experimental conditions and offer a modest performance improvement in fitness prediction. FLIGHTED results can also be used to combine fitness readouts from different experiments and optimize experimental design by maximizing signal-to-noise.

Our results on FLIGHTED-Selection suggest that common ML benchmarks like FLIP [6] or ProteinGym [10] need to be corrected to account for noise inherent in single-step selection. Correcting these benchmarks will change downstream evaluations of protein models and should improve performance on average.

FLIGHTED-DHARMA enables the use of DHARMA as a data generation method for machine learning. The noisy nature of biological circuits presents challenges in effectively using DHARMA to quantify activities. We provide calibrated error estimates that inform practitioners when they have enough DHARMA reads to make a reliable fitness estimate. DHARMA can be used to cheaply generate large datasets of up to 10^6 variants using nanopore sequencing alone. Comparing FLIGHTED predictions for DHARMA and single-step selection, we see that DHARMA may be more accurate for distinguishing between highly active variants while selection experiments may be more accurate at identifying the fitnesses of comparatively inactive variants.

Our benchmarking results support two important conclusions about the development of future protein fitness ML models. First, data size is currently the most important factor, underscoring the importance of gathering large, high-quality datasets through methods such as DHARMA. Second, the top model architecture matters much more than the embedding, so practitioners should focus on development of top model architecture and less on scaling up protein language models. Specifically, in the first systematic exploration of CNNs as top models, we found that they performed very well while being computationally cost-efficient; we encourage more exploration of their use as a top model, as well as development of other architectures for top models.

4 Methods

All models were implemented using the probabilistic programming package Pyro [30] alongside PyTorch [31]. Use of Pyro makes model development substantially easier due to rapid iteration of model architectures.

4.1 Single-Step Selection

4.1.1 Simulation and Dataset Generation

Single-step selection experiments were simulated on a landscape of $N_{\text{var}} = 20^4$ variants, with the ith variant having a fitness or selection probability p_i uniformly distributed between 0 and 1, i.e. $p_i \sim \mathcal{U}([0, 1])$. The total initial population $N_{\text{init}} = 10^{11}$. Each variant i had an initial population drawn from the Dirichlet distribution $N_{\text{init},i} \sim N_{\text{init}}\text{Dir}(\alpha), \alpha = (1, \ldots, 1)$. An initial pre-selection sample $N_{\text{sampled, init},i}$ was drawn according to a multinomial distribution $\text{Mult}(N_{\text{init},i})$, with $N_{\text{sampled, init}} = 10^8$ samples being drawn.

In the selection step, the ith variant had a post-selection population drawn from the binomial distribution, i.e. $N_{\text{final},i} \sim B(N_{\text{init},i}, p_i)$. The post-selection sample $N_{\text{sampled, final},i}$ was similarly drawn according to a multinomial distribution $\text{Mult}(N_{\text{final},i})$ with $N_{\text{sampled, final}} = 10^8$ samples being drawn. We ran 100 simulations on a single landscape to generate training data and to assess the noise levels in the enrichment ratio as shown in Figure 2c.

The parameters of this simulation were selected to be similar to those used in generating the GB1 dataset [20]. Simulations with smaller numbers of reads drawn both pre- and post-selection were used to assess robustness. Simulations were also run with varying Dirichlet parameters and using a beta distribution for the selection probability instead of a uniform distribution; these did not significantly affect the noise found in the enrichment ratio. Specifically, we replaced the uniform distribution used to draw the selection probabilities with a beta distribution, and varied the argument of the beta distribution. This represents fitnesses that peak at lower or higher values as opposed to being uniform between 0 and 1. We also replaced the argument of the Dirichlet distribution used for drawing the initial population with $(\alpha, \ldots, \alpha)$, allowing for biases in the initial population towards one variant.

4.1.2 The FLIGHTED-Selection Model

The model is an implementation of the probabilistic graphical model shown in Figure 2b in Pyro. The sequence-to-fitness function is simply a dictionary lookup, with the fitness of each variant i defined as a parameter to be optimized. Fitness means m_i and variances σ_i are predicted by the sequence-to-fitness function/dictionary, and the variance is transformed by a softplus function F to ensure it is positive. The fitness of each variant is then sampled from a normal distribution $\mathcal{N}(m_i, F(\sigma_i))$ and transformed by a sigmoid function to be between 0 and 1. To avoid issues near the boundaries, fitnesses are clamped to be between 0.001 and 0.999. The initial population of each variant $N_{\text{init},i}$ is a per-data-point parameter in Pyro. We then sample from

the multinomial and binomial distributions as described in the simulation section to implement the desired probabilistic graphical model.

The guide predicts the fitness mean for a given experiment as logit $\left(\frac{\text{ENR}}{100}\right)$, where ENR is the enrichment ratio. The scale factor of 100 ensures that the transformed enrichment ratio is between 0 and 1 before being fed into the logit function; it can be changed arbitrarily with no impact on model performance after retraining. The variance is predicted by a linear model with 2 inputs: $\frac{N_{\text{sampled, init},i}}{N_{\text{sampled, init}}}$ and $\frac{N_{\text{sampled, final},i}}{N_{\text{sampled, final}}}$. The predicted variance is also transformed by a softplus function to ensure that it is positive.

Each data point for the FLIGHTED-Selection model consists of an experiment or simulation, not the outcome of a single variant within an experiment, because the number of reads of a given variant sampled pre- and post-selection depends on the population of other variants in the sample. To combine predictions from the guide model for different experiments in a single batch, we simply multiply the relevant Gaussians and sample the fitness from the resulting combined Gaussian distribution.

We used a learning rate of 10^{-2}, a learning rate on the landscape model (the sequence-to-fitness function) of 10^{-1}, a batch size of 10, 150 epochs, 10 particles in the ELBO loss function in Pyro, and a plateau learning rate scheduler with a patience of 4 epochs. Other hyperparameters can be found in the released code. The model was trained on 80 of the 100 simulations, validated on another 10 (randomly split), and tested on the final 10.

The FLIGHTED-Selection robustness tests were done by changing some parameters of the simulation. We replaced the uniform distribution used to draw the selection probabilities with a beta distribution, and varied the argument of the beta distribution. This represents fitnesses that peak at lower or higher values as opposed to being uniform between 0 and 1. We also replaced the argument of the Dirichlet distribution used for drawing the initial population with $(\alpha, \ldots, \alpha)$, allowing for biases in the initial population towards one variant.

4.2 DHARMA

4.2.1 T7 Polymerase Dataset Generation

We engineered a biological circuit to associate the enzymatic characteristics of T7 RNA polymerase variants with mutations accumulating on a designated DNA sequence, known as the canvas. T7 RNA polymerase is an enzyme responsible for transcribing DNA into RNA, and we constructed a library of variants with different recognition and activity profiles.

The transcription of the base editor, an enzyme that can induce mutations, is controlled by a T3 promoter. This promoter is selectively recognized by a subset of the T7 RNA polymerase variants. To moderate the expression levels of the T7 polymerase library and prevent rapid saturation of the canvas—which would compromise the collection of meaningful activity data—we used both a weak constitutive promoter and a weak ribosomal binding site.

We incorporated this biological circuit into cells already containing plasmids that express the single guide RNA (sgRNA), through a technique called electroporation.

The sgRNA serves to guide the base editor to the specific canvas sequence where mutations are to be introduced. After electroporation, the cells were grown continuously in a bioreactor. Following this growth period, the region of the T7 polymerase library and the canvas with mutations was selectively amplified as contiguous fragments of DNA. The amplified material was then subjected to long-read Nanopore sequencing for detailed analysis of the mutations and activities of different T7 RNA polymerase variants.

To facilitate data analysis, we implemented a data processing pipeline. Its core function is to identify, tabulate, and assign mutations present in each sequencing read to the corresponding library member. This assignment is based on internal barcodes represented as degenerate codons in the T7 RNA polymerase sequence. The pipeline accepts raw sequence reads in standard genomic formats and performs length-based filtering and optional sequence trimming. An algorithm incorporating local sequence alignment is used for barcode recognition and classification during the demultiplexing of reads. Additionally, each read is aligned to the reference sequence of the canvas to identify the location of each C→T mutation, which is then stored in a matrix for downstream ML model training.

4.2.2 The FLIGHTED-DHARMA Model

The model is an implementation of the probabilistic graphical model shown in Figure 3b in Pyro. The sequence-to-fitness function is simply a dictionary lookup, with the fitness of each variant i defined as a parameter to be optimized. Fitness means m_i and variances σ_i are predicted by the sequence-to-fitness function/dictionary, and the variance is transformed by a softplus function F to ensure it is positive. The fitness of each variant is then sampled from a normal distribution $\mathcal{N}(m_i, F(\sigma_i))$ and fitnesses are clamped at -2.

For each position i in the canvas, we set a learnable parameter r_i for a baseline rate of generic mutations (which accounts for most sequencing error from the long-read sequencing). For positions that are cytosines, we set a learnable parameter m_i for the fitness-dependent slope and b_i for the intercept. Then for all cytosines, given a fitness f of the variant, the logit of the C→T edit is set to $m_i + b_i f$ and the logit of any other mutation or deletion is r_i. For non-cytosine residues, we simply have a logit of any mutation set to r_i. We sample from the one-hot categorical distribution for each residue independently to get the output DHARMA read, i.e. from $\mathrm{Cat}((1, m_i + b_i f, r_i))$ for cytosine residues and $\mathrm{Cat}((1, -\infty, r_i))$ for non-cytosine residues.

The guide predicts fitness (with variance) from a single DHARMA read. We used a feedforward network with 2 layers with a hidden dimension of 10 and a ReLU activation between the two layers. To simplify the learning problem, only cytosine residues were fed into the guide and each position was featurized as either a C→T edit or a non-C→T edit (including both no mutation or other mutations or deletions). Variances were transformed under the softplus function. Fitnesses of variants with multiple reads can be predicted by multiplying the appropriate predicted Gaussians of the individual read.

The ELBO loss used 1 ELBO particle. The landscape model (sequence-to-fitness function) learning rate was 10^{-2}, the learning rate elsewhere was 10^{-4}, and the batch

size was 2. We used a plateau learning rate scheduler with a patience of 1 epoch and a factor of 0.1, stepped based on the training loss (not validation loss). The model was trained for 25 epochs.

To minimize data leakage, all reads corresponding to sequences in the fluorescence dataset were eliminated from training. 10% of the remaining reads were held out as a validation set to pick the optimal epoch for the model.

Many of the hyperparameters were carefully tuned using a grid search. For tuning, we wanted to use the MSE with the FACS data, so we needed to fit the predicted fitnesses to the FACS data. We did not use Spearman correlation so we could measure true error (as opposed to rank-order error) and so we could account for situations where the function relating the FACS data to predicted fitness was not increasing (as happened here). For this, we split out a 50% of the FACS data to use as a fitness regression/validation set and evaluated the mean of the logarithm of all FACS samples. Each model was given the option of fitting using either a piecewise linear function or a linear regression. The linear regression was fit normally. The piecewise linear function corresponded to the functional form

$$f(x) = \begin{cases} a & x \leq x_c \\ m(x - x_c) + a & x > x_c. \end{cases}$$

This ensured that for low fitnesses, the predicted fluorescence was constant, as would be expected for background fluorescence. We then used orthogonal distance regression, as implemented in `scipy`, to fit the piecewise linear function accounting for errors in both predicted fitness and the FACS data. [32] Between the linear regression and piecewise linear fit, the function with the lower MSE on the validation data was selected.

This validation set MSE was also used for selecting hyperparameters, leaving the remaining 50% of the FACS data as a test set used solely for evaluating model performance as shown in Figure 3c. Hyperparameters tuned included the batch size, learning rate, learning rate scheduler step, and number of layers in the guide model. We also evaluated the optimal number of hours to grow the cells in the bioreactor, as an example of an experimental parameter than can be tuned using FLIGHTED.

4.2.3 Model Evaluation

Model performance was measured with the 50% held-out test set of the FACS data, as described above, using the fitnesses predicted by the guide model in FLIGHTED-DHARMA and the fitness-to-fluorescence function fit on the validation set. The results of that performance evaluation are shown in Figure 3e.

We then evaluated calibration of the model. To increase the number of data points (since we only had 119 variants measured through FACS), we subsampled subsets of reads for each variant. Specifically, for all test set variants, we sampled 10 subsets each of sizes ranging from 1 to the maximum possible number of reads. We then predicted the fitness of each variant using the given subset of reads with the guide model. We computed the true fitness using the FACS data, eliminating any data points where the true fitness was below the baseline of the piecewise linear function. We then computed

the z-score by comparing this predicted and true fitness. This generates the plots shown in Figure 3f and 3g.

4.3 Benchmarking Models on FLIGHTED Datasets

4.3.1 The GB1 Dataset

To compute the corrected GB1 landscape with FLIGHTED-Selection, we took the guide and ran inference on the released GB1 landscape data, which provided both pre- and post-selection read counts in addition to the enrichment ratio [20]. We omitted all data points that were omitted in the original study due to not being observed. We then followed the published data splits provided by FLIP [6] to generate datasets both with and without FLIGHTED-Selection for the GB1 problem. Models trained on datasets with corrections from FLIGHTED-Selection are trained with a weighted mean-squared-error (MSE) loss, weighted by the inverse variance. Weighted MSE is used to account for the variance, assuming that the likelihood of the data is the same as that of a normal distribution with the provided variance. Performance results are regular MSEs for the datasets without corrections and weighted MSEs for datasets with corrections. Weighted Spearman ρ measurements were made with 100 Monte Carlo replicates [27].

4.3.2 The TEV Dataset

High-throughput quantification of TEV protease fitness using DHARMA was made possible by a biological circuit that couples the activity of TEV protease to base editor transcription driven by a T7 promoter. As part of the circuit, the T7 RNA polymerase, normally repressed by its natural inhibitor T7 lysozyme, becomes functional after the inhibitor is disabled by active TEV protease variants, thus allowing the transcription of base editor to proceed, which in turn introduce mutations to the canvas. We engineered a variant of T7 lysozyme in which the TEV protease substrate sequence is inserted in the middle of the coding sequence. This modified T7 lysozyme, expressed on a separate plasmid under the control of a medium constitutive promoter, together with the T7 lysozyme tethered to the T7 RNAP, provides an enhanced dynamic range of base editing coupled to the activity of TEV protease variants.

The library of TEV protease used in this study was obtained by performing site-saturation mutagenesis on 4 amino acid residues (T146, D148, H167, and S170) in the TEV protease S1 pocket, which is known to interact with P1 residue on the substrate and determine substrate specificity. Briefly, NNK degenerate primers was used to introduce mutations at the targeted residues in a PCR reaction. The pool of amplicons was then cloned into a Golden Gate vector comprising the rest of the TEV protease expression cassette, the sgRNA and the canvas sequence. Commercial electrocompetent cells were then transformed with the cloning reaction, selected with appropriate antibiotics on agar plate overnight and subjected to DNA extraction. The resulting library was then sequenced to assess for bias and coverage. After quality control, the TEV protease library was transformed into electrocompetent cells that already express the base editor, the T7 RNAP and the engineered T7 lysozyme. The transformants were then selected with appropriate animate and grown continuously

in a bioreactor. Multiple time points were taken during this growth period to find the optimal incubation time. The region of the plasmid containing TEV protease library and the canvas sequence was selectively amplified as contiguous fragments of DNA. To minimize the amplicon size, nucleotides not of interest were removed via self-circularization and re-amplification. The final amplified material was then subjected to long-read nanopore sequencing to simultaneously retrieve both variant identity and the mutations on the canvas sequence for each individual library member. Sequencing data processing and analysis was performed as described in the T7 dataset generation section (Section 4.2.1).

The TEV landscape was generated by processing the raw TEV data with FLIGHTED-DHARMA as trained on the T7 dataset. We split the dataset by mutation distance from the wild-type to create a one-vs-rest, two-vs-rest, and three-vs-rest data split. We then split out a random 10% of the training data to use as validation data. Models trained on the TEV dataset are trained with a weighted mean-squared-error (MSE) loss, weighted by the inverse variance. Performance results are also weighted MSEs. In Figure 4c, we use a separate test set consisting of only quadruple mutants (i.e. the test set in the three-vs-rest split) for all models.

4.3.3 Modeling the GB1 and TEV Datasets

Our models are inspired by but go beyond the models proposed in FLIP (Fitness Landscape Inference for Proteins) [6]. The linear regression model (labeled Linear) takes one-hot embeddings of the full sequence and runs them through 1 linear layer. It uses an Adam optimizer with a learning rate of 10^{-2}, a batch size of 256, and a weight decay of 1. The CNN model (labeled CNN) takes one-hot embeddings of the full sequence and has 1 convolutional layer with 1024 channels and filter size 5, and same padding. It then has a 1D batch normalization layer and a ReLU activation, followed by an embedding neural network consisting of a linear layer to 2048 dimensions and a ReLU activation. Then there is a max-pooling layer over residues, a dropout layer with probability 0.2, and a final linear layer for the output. The CNN is trained with a batch size of 256 and an Adam optimizer with a learning rate of 10^{-3} and weight decay of 0 for the convolutional layer, a learning rate of $5 * 10^{-5}$ and weight decay of 0.05 for the embedding layer, and a learning rate of $5 * 10^{-6}$ and weight decay of 0.05 for the output layer. Unlike the original FLIP paper, we did not use early stopping and trained all models for a full 500 epochs, using validation set performance to select the optimal model.

The models labeled TAPE, ESM-1b, ESM-1v, ESM-2, ProtT5, and CARP all use mean embeddings across the entire sequence that are fed into a feedforward neural network to compute the output [7, 23, 24, 33, 34]. The output feedforward neural network has 2 layers with a hidden dimension equal to the embedding dimension of the protein language model and ReLU activation. All models are trained with an Adam optimizer with batch size 256 and learning rate 10^{-3}. The TAPE model used was the transformer, the ESM-1v model used was version 1, the largest ESM-2 model used was the 3 billion parameter version (due to memory issues with the larger model), the ProtT5 model was the suggested ProtT5-XL-UniRef50 model, and the CARP model was the largest 640 million parameter version. For Figure 4d, we ran smaller versions

of CARP and the ESM-2 models, as specified by their parameter number, with the same architectures as described above [24, 28].

The models labeled TAPE (CNN), ESM-1b (CNN), ESM-1v (CNN), ESM-2 (CNN), ProtT5 (CNN), and CARP (CNN) use residue-level embeddings that are fed into a CNN, similar to the baseline CNN described above. The intermediate dimension output by the embedding neural network is set to be twice the dimension of the output of the protein language model. All other parameters remained the same.

The models labeled ESM-1v (Augmented), ESM-2 (Augmented), CARP (Augmented), and EVMutation (Augmented) used the zero-shot variant effect prediction from the model in question, combined with a one-hot encoding of the entire sequence that was fed into a linear layer [23, 24, 28, 35]. These were trained with the same parameters as the baseline linear model. The ESM models used the masked marginal approach proposed in [23] to compute zero-shot variant effect prediction. EVMutation used the default parameters [35].

Fine-tuned models are trained as described above, but we also fine-tuned the underlying ESM model that generated the embeddings with a learning rate of 10^{-6}. Due to compute and cost limitations, fine-tuning was only done for the ESM models with 8 and 35 million parameters for the TEV dataset and 8, 35, and 150 million parameters for the GB1 dataset.

We observed high run-to-run variance in the performance of many models. As a result, all models were run in triplicate and we have reported both the mean and standard deviation of model performance.

Supplementary information. The supplement contains a detailed discussion of the impact of read count on fitnesses measured in the TEV landscape.

Acknowledgements. V.S. acknowledges funding from the Fannie and John Hertz Foundation. B.T. acknowledges funding from the National Science Foundation Graduate Research Fellowship under Grant No. 2141064. L.G. acknowledges funding from the National Institutes of Health under Grant. No. T32GM087237. We are grateful to gifts from Open Philanthropy Project and the Aphorism Foundation (to K.M.E.). All figures were made with BioRender.

Declarations

Competing Interests

We do not have any competing interests.

Data Availability

Model weights for FLIGHTED-Selection and FLIGHTED-DHARMA, the FLIGHTED GB1 landscape, and all GB1 models have been released at this Zenodo repository. The FLIGHTED TEV landscape and all TEV models have been released at this Zenodo repository.

Code Availability

All code necessary to run FLIGHTED and reproduce all figures in this paper has been released in at this Github repository.

Author Contribution

V.S., B.T., and K.M.E. conceived the study. V.S. designed and implemented FLIGHTED and B.T. developed DHARMA and gathered the experimental data, both under the supervision of K.M.E. L.G. aided in benchmarking and optimization of FLIGHTED-DHARMA. V.S. wrote the first draft of the paper. All authors contributed to editing and approved the submission.

Appendix A Extended Data

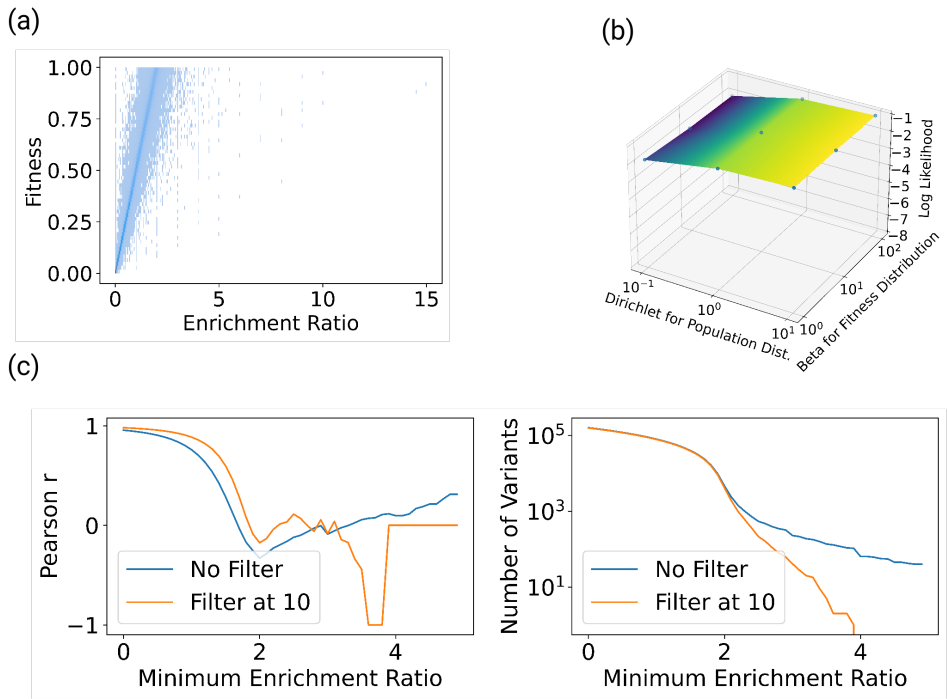

Fig. A1 Noise in Single-Step Selection Simulations and Robustness of FLIGHTED-Selection Model Performance. (a) Another perspective on the noise, showing significant noise in the enrichment ratio. (b) The beta distribution is used to determine the distribution of fitnesses and the Dirichlet distribution is used to determine the distribution of initial populations. We see that these parameters in general have minimal effect on FLIGHTED-Selection log likelihood, but with very low Dirichlet parameters there is a decrease in performance. (c) Left: the x-axis is the minimum enrichment ratio and the y-axis is the Pearson r between enrichment ratio and fitness for all data points above that minimum. The drastic drop-off in Pearson r as minimum enrichment ratio increases shows that high-activity variants essentially provide 0 information from just their enrichment ratio. Right: y-axis is the number of variants, showing that this is still a significant number of variants. We also tried filtering at a minimum of 10 reads, which improves results but still shows significant noise.

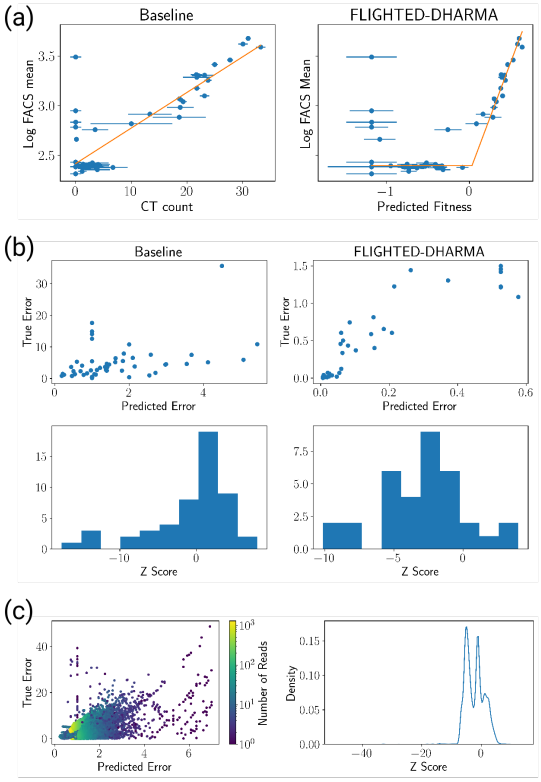

Fig. A2 FLIGHTED-DHARMA Model Performance. (a) Accuracy on Validation set for (left) Baseline and (right) FLIGHTED-DHARMA. This shows the fit between the fitnesses predicted by each model and the log FACS mean, as done by either a piecewise linear function or a linear function. (b) Calibration of FLIGHTED-DHARMA as Compared to Baseline. The baseline model's true errors are not related to the predicted error, while FLIGHTED-DHARMA has higher true error when predicted error is higher. As such, FLIGHTED-DHARMA is considerably better calibrated with many fewer z-score outliers. (c) Calibration of Baseline C→T Model. The baseline model's calibration is considerably worse compared to FLIGHTED-DHARMA's in Figure 3f and g.

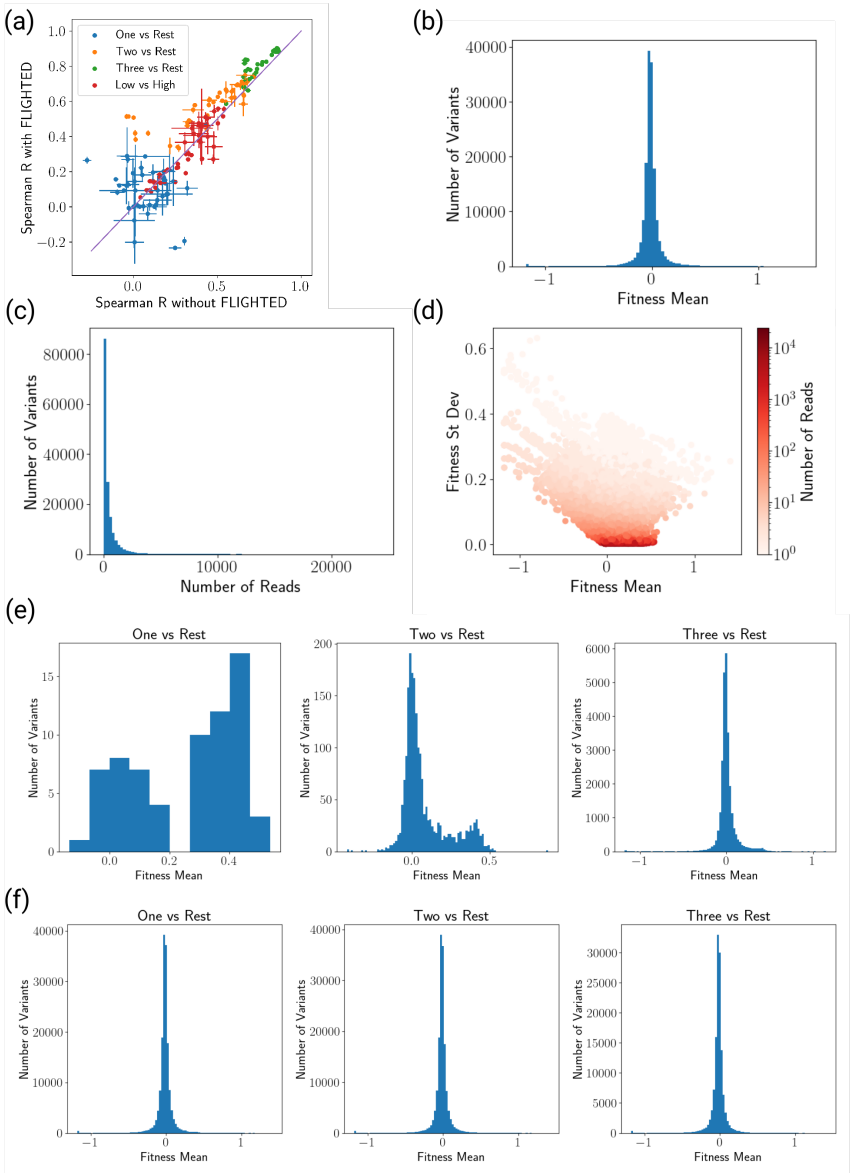

Fig. A3 Benchmarking on FLIGHTED Landscapes. (a) FLIGHTED also modestly improves model performance on the GB1 dataset, even when performance is measured as Spearman ρ to the non-FLIGHTED test set. (b) Distribution of fitnesses in the TEV dataset. (c) Distribution of read counts in the TEV dataset. (d) Fitness mean, variances, and read count in the TEV dataset. (e) Distribution of fitnesses in the training sets for various splits of the TEV dataset. (f) Distribution of fitnesses in the test sets for various splits of the TEV dataset.

Model	One vs Rest	One vs Rest FLIGHTED	Two vs Rest	Two vs Rest FLIGHTED
Linear	1.4706 ± 0.0127	12.0096 ± 0.0709	1.5174 ± 0.0020	9.8847 ± 0.0342
TAPE	1.5062 ± 0.0090	11.8502 ± 0.2601	1.9685 ± 0.1161	7.1776 ± 0.3339
ESM-1b	1.4898 ± 0.0004	**11.2226 ± 0.0750**	1.4427 ± 0.0107	5.7642 ± 0.2590
ESM-1v	1.5132 ± 0.0016	14.7296 ± 0.0125	1.5646 ± 0.0153	6.7795 ± 0.2354
ESM-2 (8M)	1.5169 ± 0.0177	11.6852 ± 0.0226	1.4388 ± 0.0081	7.6910 ± 0.1750
ESM-2 (35M)	1.5218 ± 0.0218	11.5808 ± 0.0899	1.4034 ± 0.0103	8.2036 ± 0.0373
ESM-2 (150M)	1.5129 ± 0.0210	11.5738 ± 0.1378	1.4099 ± 0.0057	8.5423 ± 0.0238
ESM-2 (650M)	1.4963 ± 0.0024	11.5768 ± 0.2312	1.4285 ± 0.0288	5.5143 ± 0.2098
ESM-2	1.9206 ± 0.0334	12.0280 ± 0.1538	1.3472 ± 0.0071	5.0572 ± 0.3065
ProtT5	1.5288 ± 0.0148	11.6078 ± 0.0677	1.3567 ± 0.0259	5.4940 ± 0.0949
CARP (600k)	1.5033 ± 0.0132	11.6508 ± 0.0983	1.5659 ± 0.0472	11.2536 ± 0.1272
CARP (38M)	1.5139 ± 0.0088	11.6778 ± 0.1284	1.5768 ± 0.0059	11.0896 ± 0.1535
CARP (76M)	1.4884 ± 0.0145	11.5002 ± 0.2140	1.5607 ± 0.0031	9.1447 ± 0.1564
CARP	1.5070 ± 0.0082	11.7667 ± 0.2218	1.5402 ± 0.0041	9.4377 ± 0.2619
CNN	2.0506 ± 0.2307	13.8411 ± 1.6964	1.3523 ± 0.0539	4.0959 ± 0.1426
TAPE (CNN)	2.0937 ± 0.1898	15.5570 ± 0.5312	1.1251 ± 0.0631	4.9553 ± 0.6760
ESM-1b (CNN)	1.8592 ± 0.1246	14.7160 ± 1.1399	1.1482 ± 0.0239	4.6976 ± 0.4697
ESM-1v (CNN)	2.4676 ± 0.2503	15.0658 ± 1.1423	1.1188 ± 0.0212	5.6638 ± 0.6273
ESM-2 (8M, CNN)	2.3919 ± 0.1627	15.5196 ± 0.4909	1.0913 ± 0.0416	4.8430 ± 0.5947
ESM-2 (35M, CNN)	2.4945 ± 0.4818	14.7227 ± 0.6925	1.0381 ± 0.0769	3.8103 ± 0.1281
ESM-2 (150M, CNN)	2.2499 ± 0.5033	15.6027 ± 1.0554	1.1085 ± 0.1243	4.7428 ± 0.4609
ESM-2 (650M, CNN)	1.8934 ± 0.3340	14.2969 ± 0.7712	1.1476 ± 0.0261	4.7025 ± 0.6881
ESM-2 (CNN)	1.9511 ± 0.1986	14.2011 ± 1.6117	1.3850 ± 0.0382	5.3430 ± 0.8577
ProtT5 (CNN)	1.8554 ± 0.0745	14.9394 ± 0.4143	1.0373 ± 0.0246	5.0151 ± 1.0703
CARP (600k, CNN)	2.6833 ± 0.3544	14.1731 ± 0.3073	1.2257 ± 0.0350	5.2804 ± 0.9685
CARP (38M, CNN)	2.0301 ± 0.2404	15.0381 ± 1.5099	1.1312 ± 0.0228	4.6168 ± 0.4402
CARP (76M, CNN)	1.9932 ± 0.0644	14.6097 ± 0.3420	1.1944 ± 0.0091	3.8172 ± 0.2085
CARP (CNN)	1.7704 ± 0.0174	15.0078 ± 0.8024	1.3109 ± 0.0596	5.4695 ± 0.7086
ESM-1v (Augmented)	1.4845 ± 0.0018	11.7308 ± 0.2460	1.5685 ± 0.0150	10.0828 ± 0.1070
ESM-2 (8M, Augmented)	1.5088 ± 0.0170	11.9113 ± 0.0402	1.6641 ± 0.0156	10.0938 ± 0.0589
ESM-2 (35M, Augmented)	1.5102 ± 0.0220	11.9370 ± 0.0565	1.6291 ± 0.0127	10.1836 ± 0.1647
ESM-2 (150M, Augmented)	1.5230 ± 0.0233	11.9377 ± 0.1009	1.5985 ± 0.0040	9.7887 ± 0.1380
ESM-2 (650M, Augmented)	1.4985 ± 0.0086	11.8261 ± 0.0717	1.6410 ± 0.0120	9.6624 ± 0.1501
ESM-2 (Augmented)	1.5115 ± 0.0318	11.7135 ± 0.1641	1.6180 ± 0.0112	9.6967 ± 0.0494
CARP (600k, Augmented)	**1.4652 ± 0.0021**	11.9570 ± 0.0983	1.5064 ± 0.0039	9.9571 ± 0.1009
CARP (38M, Augmented)	1.4777 ± 0.0186	11.9582 ± 0.0450	1.4982 ± 0.0025	9.9154 ± 0.0556
CARP (76M, Augmented)	1.4757 ± 0.0199	11.7828 ± 0.1579	1.4985 ± 0.0058	9.8734 ± 0.0189
CARP (Augmented)	1.4660 ± 0.0022	12.0082 ± 0.0337	1.5083 ± 0.0061	9.9470 ± 0.0716
EVMutation (Augmented)	1.4708 ± 0.0093	11.6674 ± 0.3938	1.4526 ± 0.0022	7.0896 ± 0.0080
ESM-2 (8M, Finetuned)	1.5318 ± 0.0264	14.9039 ± 2.5033	1.6052 ± 0.0357	4.7209 ± 0.2025
ESM-2 (35M, Finetuned)	1.5500 ± 0.0967	13.5396 ± 0.3245	1.3019 ± 0.0551	3.7613 ± 0.0887
ESM-2 (150M, Finetuned)	1.5245 ± 0.0026	11.4452 ± 0.2846	1.1094 ± 0.0467	**3.3694 ± 0.1247**
ESM-2 (8M, CNN, Finetuned)	2.0855 ± 0.3826	15.5261 ± 0.3309	1.0087 ± 0.0216	4.5865 ± 0.4793
ESM-2 (35M, CNN, Finetuned)	2.2368 ± 0.4315	13.8723 ± 0.8406	**0.8978 ± 0.0305**	3.5521 ± 0.3629
ESM-2 (150M, CNN, Finetuned)	2.8614 ± 0.2362	14.9521 ± 0.7068	1.0092 ± 0.1637	3.9133 ± 0.2355

Table A1 Raw Model Performance on GB1 Datasets (one-vs-rest and two-vs-rest only). See Methods for a detailed description of all models and dataset splits. Bolded models are the highest-performing in a given task. Highlighted models are not statistically significantly different via a two-sided t-test from the top model at a significance level of $p < 0.05$.

Model	Three vs Rest	Three vs Rest FLIGHTED	Low vs High	Low vs High FLIGHTED
Linear	1.5172 ± 0.0049	8.1622 ± 0.0122	4.6324 ± 0.0126	7.5193 ± 0.1734
TAPE	0.9591 ± 0.0295	3.6993 ± 0.0375	3.5252 ± 0.0394	2.4557 ± 0.1641
ESM-1b	0.9484 ± 0.0104	3.2003 ± 0.0183	3.5314 ± 0.0218	2.5016 ± 0.1014
ESM-1v	0.9261 ± 0.0185	3.4494 ± 0.0882	3.4070 ± 0.0201	1.7771 ± 0.0768
ESM-2 (8M)	1.0639 ± 0.0142	4.3834 ± 0.0081	3.6068 ± 0.0302	3.4855 ± 0.0535
ESM-2 (35M)	0.9534 ± 0.0022	3.8483 ± 0.0362	3.5095 ± 0.0738	2.9552 ± 0.0789
ESM-2 (150M)	0.9726 ± 0.0052	4.3019 ± 0.1824	3.5763 ± 0.0271	3.0476 ± 0.0451
ESM-2 (650M)	0.7834 ± 0.0036	2.9651 ± 0.0779	3.4553 ± 0.0089	2.3014 ± 0.0535
ESM-2	0.8850 ± 0.0038	2.8603 ± 0.0090	3.4237 ± 0.0502	2.1913 ± 0.0138
ProtT5	0.8706 ± 0.0072	3.0169 ± 0.0259	3.3162 ± 0.0289	2.0257 ± 0.0282
CARP (600k)	1.1682 ± 0.0112	5.5622 ± 0.0866	3.9904 ± 0.0258	4.4635 ± 0.1299
CARP (38M)	1.3215 ± 0.0152	6.4299 ± 0.1360	4.2556 ± 0.0272	5.4571 ± 0.0665
CARP (76M)	1.1813 ± 0.0074	5.3000 ± 0.1357	4.1310 ± 0.0059	4.7139 ± 0.2040
CARP	1.2078 ± 0.0052	5.0631 ± 0.2182	4.1793 ± 0.0399	5.0080 ± 0.1777
CNN	0.5158 ± 0.0101	1.5133 ± 0.0458	3.1096 ± 0.0319	**0.6077 ± 0.0747**
TAPE (CNN)	0.4504 ± 0.0039	1.2584 ± 0.0790	3.1693 ± 0.0720	0.7386 ± 0.0647
ESM-1b (CNN)	0.4406 ± 0.0066	1.2828 ± 0.0583	3.2526 ± 0.1097	0.8580 ± 0.1646
ESM-1v (CNN)	**0.4218 ± 0.0148**	1.0939 ± 0.0647	3.2323 ± 0.1479	0.8102 ± 0.0425
ESM-2 (8M, CNN)	0.4587 ± 0.0071	1.3937 ± 0.1154	**3.1004 ± 0.0596**	0.8299 ± 0.0669
ESM-2 (35M, CNN)	0.4736 ± 0.0061	1.3928 ± 0.0585	3.1440 ± 0.0846	0.6806 ± 0.1123
ESM-2 (150M, CNN)	0.4724 ± 0.0051	1.3727 ± 0.0349	3.1400 ± 0.0445	0.7269 ± 0.1354
ESM-2 (650M, CNN)	0.4396 ± 0.0515	1.2475 ± 0.1657	3.3286 ± 0.0471	0.8302 ± 0.0805
ESM-2 (CNN)	0.4440 ± 0.0259	1.2242 ± 0.1160	3.3670 ± 0.1110	0.8903 ± 0.0651
ProtT5 (CNN)	0.4576 ± 0.0232	1.1711 ± 0.0338	3.2488 ± 0.0403	0.8296 ± 0.0278
CARP (600k, CNN)	0.5965 ± 0.0216	1.7806 ± 0.0977	3.2274 ± 0.0671	0.9752 ± 0.1267
CARP (38M, CNN)	0.4854 ± 0.0094	1.1501 ± 0.0552	3.1777 ± 0.0413	0.8558 ± 0.0859
CARP (76M, CNN)	0.5070 ± 0.0409	1.2360 ± 0.0822	3.2070 ± 0.0067	0.8577 ± 0.0944
CARP (CNN)	0.4661 ± 0.0215	1.3619 ± 0.1444	3.2760 ± 0.0515	0.7709 ± 0.1090
ESM-1v (Augmented)	1.4944 ± 0.0063	7.9110 ± 0.1260	4.6129 ± 0.0221	7.1360 ± 0.1249
ESM-2 (8M, Augmented)	1.5203 ± 0.0064	7.9353 ± 0.0718	4.6053 ± 0.0032	7.3604 ± 0.1191
ESM-2 (35M, Augmented)	1.4982 ± 0.0024	7.9133 ± 0.0427	4.6106 ± 0.0130	7.3177 ± 0.0638
ESM-2 (150M, Augmented)	1.4950 ± 0.0080	7.6113 ± 0.0829	4.5462 ± 0.0280	6.9321 ± 0.0473
ESM-2 (650M, Augmented)	1.5175 ± 0.0049	7.3998 ± 0.1201	4.5873 ± 0.0184	6.8160 ± 0.0256
ESM-2 (Augmented)	1.5265 ± 0.0051	7.7374 ± 0.1413	4.6023 ± 0.0446	7.1383 ± 0.0245
CARP (600k, Augmented)	1.5168 ± 0.0064	8.0217 ± 0.0664	4.5907 ± 0.0208	7.4045 ± 0.1267
CARP (38M, Augmented)	1.5153 ± 0.0088	8.0537 ± 0.0542	4.5971 ± 0.0493	7.4798 ± 0.0527
CARP (76M, Augmented)	1.5177 ± 0.0027	8.2109 ± 0.1684	4.6521 ± 0.0142	7.5263 ± 0.0519
CARP (Augmented)	1.5186 ± 0.0034	8.2201 ± 0.0617	4.6398 ± 0.0729	7.5125 ± 0.1570
EVMutation (Augmented)	1.5126 ± 0.0068	6.6964 ± 0.0356	4.5886 ± 0.0487	7.2021 ± 0.0813
ESM-2 (8M, Finetuned)	0.5035 ± 0.0127	**0.9590 ± 0.0316**	3.4907 ± 0.0767	1.0527 ± 0.0363
ESM-2 (35M, Finetuned)	0.5780 ± 0.0566	1.1488 ± 0.0564	3.5533 ± 0.0036	1.0101 ± 0.0532
ESM-2 (150M, Finetuned)	0.4928 ± 0.0451	1.0775 ± 0.1876	3.6195 ± 0.0081	1.0493 ± 0.0379
ESM-2 (8M, CNN, Finetuned)	0.4613 ± 0.0162	1.0441 ± 0.0517	3.2580 ± 0.0426	0.9389 ± 0.0713
ESM-2 (35M, CNN, Finetuned)	0.4359 ± 0.0166	1.0510 ± 0.0885	3.2956 ± 0.0378	0.7850 ± 0.1027
ESM-2 (150M, CNN, Finetuned)	0.4476 ± 0.0211	1.0069 ± 0.0271	3.2438 ± 0.0467	0.6934 ± 0.0123

Table A2 Raw Model Performance on GB1 Datasets (three-vs-rest and low-vs-high only). See Methods for a detailed description of all models and dataset splits. Bolded models are the highest-performing in a given task. Highlighted models are not statistically significantly different via a two-sided t-test from the top model at a significance level of $p < 0.05$.

Model	One vs Rest	Two vs Rest	Three vs Rest
Linear	526.0933 ± 22.7232	473.0123 ± 30.1776	61.9535 ± 9.9049
TAPE	297.5274 ± 16.9196	42.1769 ± 5.7037	10.1017 ± 2.9131
ESM-1b	165.9262 ± 0.8583	19.2814 ± 2.1384	4.7189 ± 0.0789
ESM-1v	**96.5753 ± 4.8990**	31.6219 ± 8.8151	5.0249 ± 0.2268
ESM-2 (8M)	487.1830 ± 67.0827	77.3104 ± 13.6258	26.0598 ± 11.2005
ESM-2 (35M)	201.0534 ± 14.7650	38.3270 ± 14.7239	5.3898 ± 0.0861
ESM-2 (150M)	253.0745 ± 17.7263	21.2191 ± 4.1567	5.8834 ± 0.7763
ESM-2 (650M)	129.0170 ± 3.0636	15.7421 ± 2.0325	5.0022 ± 0.5990
ESM-2	455.1378 ± 8.2257	21.5850 ± 7.1289	16.4862 ± 17.6678
ProtT5	128.9400 ± 5.0140	16.2017 ± 2.1542	4.5883 ± 0.2502
CARP (600k)	370.3106 ± 61.6088	67.1536 ± 5.6832	17.8028 ± 0.6229
CARP (38M)	779.2183 ± 6.6225	69.2424 ± 6.2080	11.5826 ± 1.0835
CARP (76M)	293.7587 ± 52.0933	59.6290 ± 2.5587	14.5622 ± 0.9285
CARP	661.4456 ± 227.2850	201.7350 ± 12.4057	25.1076 ± 8.3689
CNN	613.7185 ± 242.4505	169.1024 ± 7.8148	6.4333 ± 0.1398
TAPE (CNN)	535.2333 ± 132.6787	119.1340 ± 10.3364	5.5736 ± 0.2825
ESM-1b (CNN)	565.8093 ± 203.6468	40.1009 ± 9.9084	4.5211 ± 0.0431
ESM-1v (CNN)	678.3326 ± 164.8031	28.9555 ± 0.4254	4.7168 ± 0.1845
ESM-2 (8M, CNN)	537.5033 ± 238.1350	56.3152 ± 9.9305	5.3183 ± 0.1928
ESM-2 (35M, CNN)	299.9104 ± 14.8647	26.8592 ± 5.2674	224.8132 ± 191.9686
ESM-2 (150M, CNN)	676.5742 ± 103.0750	31.0988 ± 5.3971	4.7040 ± 0.1121
ESM-2 (650M, CNN)	915.3362 ± 23.6130	66.0526 ± 3.5709	4.8076 ± 0.0419
ESM-2 (CNN)	1361.5717 ± 539.0819	87.6785 ± 10.3623	4.9448 ± 0.1656
ProtT5 (CNN)	900.8455 ± 233.7611	61.6325 ± 3.8367	4.7374 ± 0.0591
CARP (600k, CNN)	479.3737 ± 9.2137	69.0239 ± 4.6387	6.8296 ± 0.4412
CARP (38M, CNN)	685.8954 ± 281.9056	79.6063 ± 9.2690	4.8927 ± 0.0288
CARP (76M, CNN)	581.1124 ± 86.5494	64.4380 ± 2.4957	4.7728 ± 0.1322
CARP (CNN)	797.2115 ± 459.2779	40.6028 ± 0.7333	4.6357 ± 0.1348
ESM-1v (Augmented)	238.4065 ± 5.2917	480.2571 ± 33.7659	63.6243 ± 3.8785
ESM-2 (8M, Augmented)	337.0354 ± 13.8129	492.2340 ± 28.7706	69.7858 ± 4.0155
ESM-2 (35M, Augmented)	216.1071 ± 23.8610	472.1725 ± 53.4941	71.2082 ± 6.5696
ESM-2 (150M, Augmented)	222.0371 ± 4.0988	490.2639 ± 30.5448	61.4350 ± 7.4693
ESM-2 (650M, Augmented)	338.1440 ± 190.0764	481.7004 ± 19.7602	69.2233 ± 11.1288
ESM-2 (Augmented)	226.2928 ± 33.0988	507.1823 ± 49.9358	69.0547 ± 3.8129
CARP (600k, Augmented)	555.9357 ± 65.4261	469.3298 ± 3.0498	65.6611 ± 1.8785
CARP (38M, Augmented)	608.4704 ± 69.0148	449.0583 ± 18.4007	61.7768 ± 5.5423
CARP (76M, Augmented)	562.6250 ± 15.5729	464.0757 ± 13.7924	64.1665 ± 7.0719
CARP (Augmented)	508.6772 ± 74.7882	459.4699 ± 36.2165	66.3554 ± 4.3394
EVMutation (Augmented)	247.1387 ± 20.9441	492.7133 ± 31.5477	62.2780 ± 6.3277
ESM-2 (8M, Finetuned)	260.1472 ± 66.3833	7.6464 ± 1.6408	4.2765 ± 0.0316
ESM-2 (35M, Finetuned)	273.4673 ± 15.2778	**7.5348 ± 0.3203**	4.2672 ± 0.0318
ESM-2 (8M, CNN, Finetuned)	517.6429 ± 199.1142	19.5290 ± 1.4134	4.7530 ± 0.1151
ESM-2 (35M, CNN, Finetuned)	211.5166 ± 46.2842	10.7844 ± 0.5173	**4.2564 ± 0.0468**

Table A3 Raw Model Performance on TEV Datasets. See Methods for a detailed description of all models and dataset splits. Bolded models are the highest-performing in a given task. Highlighted models are not statistically significantly different via a two-sided t-test from the top model at a significance level of $p < 0.05$.

References

[1] Johnston, K. E. *et al.* Machine Learning for Protein Engineering (2023). URL http://arxiv.org/abs/2305.16634. ArXiv:2305.16634 [q-bio].

[2] Frappier, V. & Keating, A. E. Data-driven computational protein design. *Current Opinion in Structural Biology* **69**, 63–69 (2021). URL https://doi.org/10.1016/j.sbi.2021.03.009. Publisher: Elsevier Ltd.

[3] Bryant, D. H. *et al.* Deep diversification of an AAV capsid protein by machine learning. *Nature Biotechnology* **39**, 691–696 (2021). URL http://www.nature.com/articles/s41587-020-00793-4.

[4] Biswas, S., Khimulya, G., Alley, E. C., Esvelt, K. M. & Church, G. M. Low-N protein engineering with data-efficient deep learning. *Nature Methods 2021 18:4* **18**, 389–396 (2021). URL https://www.nature.com/articles/s41592-021-01100-y. Publisher: Nature Publishing Group ISBN: 4159202101100.

[5] Wu, Z., Jennifer Kan, S. B., Lewis, R. D., Wittmann, B. J. & Arnold, F. H. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proceedings of the National Academy of Sciences of the United States of America* **116**, 8852–8858 (2019). ArXiv: 1902.07231 Publisher: National Academy of Sciences.

[6] Dallago, C. *et al.* FLIP: Benchmark tasks in fitness landscape inference for proteins (2022). URL https://www.biorxiv.org/content/10.1101/2021.11.09.467890v2. Section: New Results Type: article.

[7] Rives, A. *et al.* Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences* **118**, 622803 (2021). URL https://doi.org/10.1101/622803. Publisher: bioRxiv.

[8] Rao, R. *et al.* MSA Transformer (2021). URL https://www.biorxiv.org/content/10.1101/2021.02.12.430858v3. Section: New Results Type: article.

[9] Nijkamp, E., Ruffolo, J., Weinstein, E. N., Naik, N. & Madani, A. ProGen2: Exploring the Boundaries of Protein Language Models (2022). URL http://arxiv.org/abs/2206.13517. ArXiv:2206.13517 [cs, q-bio].

[10] Notin, P. *et al.* *Tranception: protein fitness prediction with autoregressive transformers and inference-time retrieval* (arXiv, 2022). URL http://arxiv.org/abs/2205.13760. Number: arXiv:2205.13760 arXiv:2205.13760 [cs].

[11] Fernandez-de Cossio-Diaz, J., Uguzzoni, G. & Pagnani, A. Unsupervised Inference of Protein Fitness Landscape from Deep Mutational Scan. *Molecular Biology and Evolution* **38**, 318–328 (2021). URL https://academic.oup.com/mbe/article/38/

1/318/5889958. Publisher: Oxford University Press.

[12] Rubin, A. F. *et al.* A statistical framework for analyzing deep mutational scanning data. *Genome Biology* **18**, 150 (2017). URL https://doi.org/10.1186/s13059-017-1272-5.

[13] Busia, A. & Listgarten, J. Model-based differential sequencing analysis (2023). URL http://biorxiv.org/lookup/doi/10.1101/2023.03.29.534803.

[14] Sesta, L., Pagnani, A., Fernandez-de Cossio-Diaz, J. & Uguzzoni, G. Inference of annealed protein fitness landscapes with AnnealDCA. *PLOS Computational Biology* **20**, e1011812 (2024). URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1011812. Publisher: Public Library of Science.

[15] Zhu, D. *et al.* Optimal trade-off control in machine learning–based library design, with application to adeno-associated virus (AAV) for gene therapy. *Science Advances* **10**, eadj3786 (2024). URL https://www.science.org/doi/10.1126/sciadv.adj3786. Publisher: American Association for the Advancement of Science.

[16] Tu, B., Sundar, V. & Esvelt, K. An ultra-high-throughput method for measuring biomolecular activities (2024). URL https://www.biorxiv.org/content/10.1101/2022.03.09.483646v4. Pages: 2022.03.09.483646 Section: New Results.

[17] Mandecki, W., Chen, J. Y.-C. & Grihalde, N. A Mathematical Model for Biopanning (Affinity Selection) Using Peptide Libraries on Filamentous Phage. *Journal of Theoretical Biology* **176**, 523–530 (1995). URL https://linkinghub.elsevier.com/retrieve/pii/S0022519385702184.

[18] Russel, M., Lowman, H. B. & Clackson, T. in *Introduction to phage biology and phage display* Practical Approach Series, 26 (Oxford University Press, Oxford, 2004).

[19] Matuszewski, S., Hildebrandt, M. E., Ghenu, A.-H., Jensen, J. D. & Bank, C. A Statistical Guide to the Design of Deep Mutational Scanning Experiments. *Genetics* **204**, 77–87 (2016). URL https://doi.org/10.1534/genetics.116.190462.

[20] Wu, N. C., Dai, L., Olson, C. A., Lloyd-Smith, J. O. & Sun, R. Adaptation in protein fitness landscapes is facilitated by indirect paths. *eLife* **5** (2016). Publisher: eLife Sciences Publications Ltd.

[21] Miller, S. M., Wang, T. & Liu, D. R. Phage-assisted continuous and non-continuous evolution. *Nature Protocols 2020 15:12* **15**, 4101–4127 (2020). URL https://www.nature.com/articles/s41596-020-00410-3. Publisher: Nature Publishing Group.

[22] Raj, A. & Oudenaarden, A. v. Nature, Nurture, or Chance: Stochastic Gene Expression and Its Consequences. *Cell* **135**, 216–226 (2008). URL https://www.

cell.com/cell/abstract/S0092-8674(08)01243-9. Publisher: Elsevier.

[23] Meier, J. *et al.* Language models enable zero-shot prediction of the effects of mutations on protein function. *Proceedings of the 35th Conference on Neural Information Processing Systems* (2021).

[24] Yang, K. K., Lu, A. X. & Fusi, N. Convolutions are competitive with transformers for protein sequence pretraining (2022). URL https://www.biorxiv.org/content/10.1101/2022.05.19.492714v1.full.pdf.

[25] Hsu, C., Nisonoff, H., Fannjiang, C. & Listgarten, J. Learning protein fitness models from evolutionary and assay-labeled data. *Nature Biotechnology* (2022). URL https://www.nature.com/articles/s41587-021-01146-5.

[26] Notin, P. *et al.* *ProteinGym: Large-Scale Benchmarks for Protein Design and Fitness Prediction* (2023). URL http://biorxiv.org/lookup/doi/10.1101/2023.12.07.570727.

[27] Curran, P. A. Monte Carlo error analyses of Spearman's rank test (2015). URL http://arxiv.org/abs/1411.3816. ArXiv:1411.3816 [astro-ph, physics:physics, stat].

[28] Lin, Z. *et al.* Language models of protein sequences at the scale of evolution enable accurate structure prediction (2022). URL https://www.biorxiv.org/content/10.1101/2022.07.20.500902v1.full.pdf.

[29] Li, F.-Z., Amini, A. P., Yue, Y., Yang, K. K. & Lu, A. X. Feature Reuse and Scaling: Understanding Transfer Learning with Protein Language Models (2024). URL https://www.biorxiv.org/content/10.1101/2024.02.05.578959v1.full.pdf.

[30] Bingham, E. *et al.* Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research* **20**, 1–6 (2019).

[31] Paszke, A. *et al.* *PyTorch: An Imperative Style, High-Performance Deep Learning Library* (2019).

[32] Jones, E., Oliphant, T., Peterson, P. & Others. SciPy: Open source scientific tools for Python (2001). URL http://www.scipy.org/.

[33] Rao, R. *et al.* *Evaluating protein transfer learning with TAPE* (2019). URL https://www.biorxiv.org/content/10.1101/676825v1. ArXiv: 1906.08230 ISSN: 1049-5258.

[34] Elnaggar, A. *et al.* ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14** (2021). URL http://biorxiv.org/lookup/doi/10.1101/2020.07.12.199554.

[35] Hopf, T. A. *et al.* Mutation effects predicted from sequence co-variation. *Nature Biotechnology* **35**, 128–135 (2017).