

440 Appendix

441 A Additional GFP analysis

442 **Design-bench difficulty.** Prior works have used the GFP task introduced by design-bench (DB),
 443 a suite of model-based reinforcement learning tasks [37], which *samples* a starting set of 5,000
 444 sequences from the 50-60th percentile fitness range. The wild-type (WT) sequences have 534
 445 fluorescence measurements ranging from the 58th to the 100th percentile (Figure 4) which implies
 446 it may be in the training set depending on the random seed⁵ and allow strong performance by
 447 memorization. Even if there is no data leakage (due to the random seed), the 50-60th percentile
 448 fitness starting range includes sequences that require 1 mutation to reach the 99th percentile (see
 449 Figure 5). The task is overly simplified in this sense and does not provide accurate evaluation of
 450 different methods.

451 **Harder difficulty.** The DB starting set is "easy" in the sense of only requiring a few mutations
 452 to sample the top sequences – BiGGS and AdaLead requires average novelty of 1.0 to sample the
 453 92nd percentile on average (Table 2). We propose the "medium" and "hard" difficulty which test a
 454 method’s ability to search large mutations and extrapolate far beyond the training set. Statistics of
 455 each difficulty are in Table 1 while a plot of their respective starting training sets is in Figure 5.

456 **New oracle.** The DB transformer-based oracle [27] is unreliable in the upper fitness echelon of
 457 sequences Figure 6. The DB oracle also susceptible to false positives. We propose a improved oracle
 458 based on a CNN that alleviates both issues. The CNN architecture is described in Section 3.

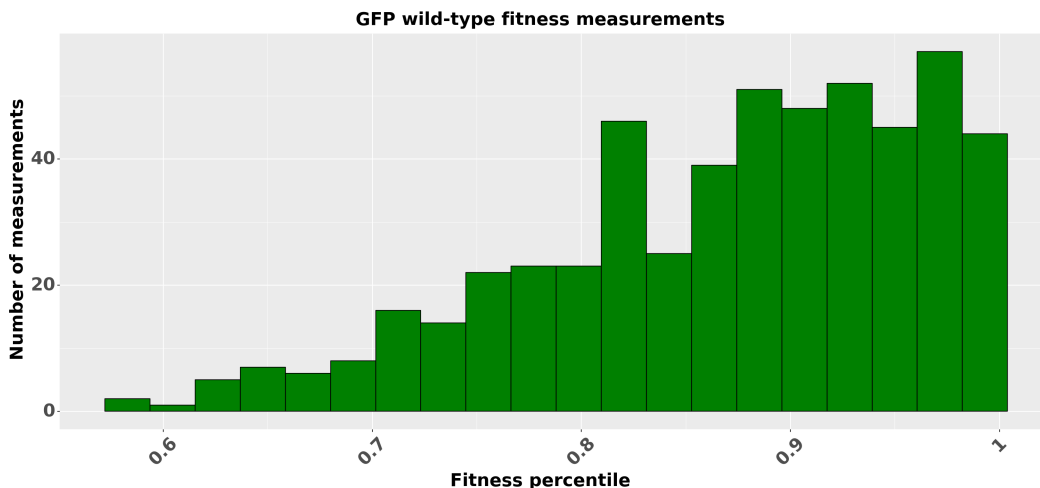


Figure 4: GFP wild-type multiplicity. The same wildtype sequence is measured a total of 534 times in the Sarkisyan et al. [31] dataset with a wide range of fitness measurements from the 58th to 100 percentile. As a result, it is possible for the wild-type or other sequences with multiple measurements to contaminate the training set when filtering only on fitness range.

⁵We confirmed this to be the case in the design-bench codebase.

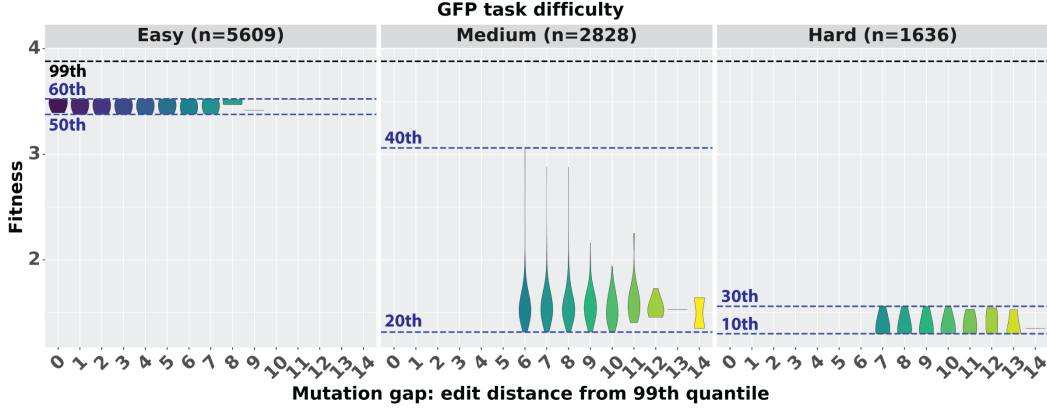


Figure 5: GFP task difficulty comparisons. **Easy difficulty** is taken from design-bench. We take all GFP sequences between the 50-60th percentile regardless of distance to sequences in the 99th percentile of GFP sequences resulting in 5609 sequences the method has access to. Data leakage is present due to the multiplicity of GFP measurements that allows the wild-type sequence and other top sequences to be present in the 50-60th percentile. **Medium difficulty** filters the starting dataset to have sequences in the 20-40th percentile and be 6 or more mutations away from anything in the top 99th percentile resulting in 2628 sequences in the starting dataset. **Hard difficulty** filters the starting dataset to have sequences in the 10-30th percentile and be 7 or more mutations away from anything in the top 99th percentile resulting in 1636 sequences.

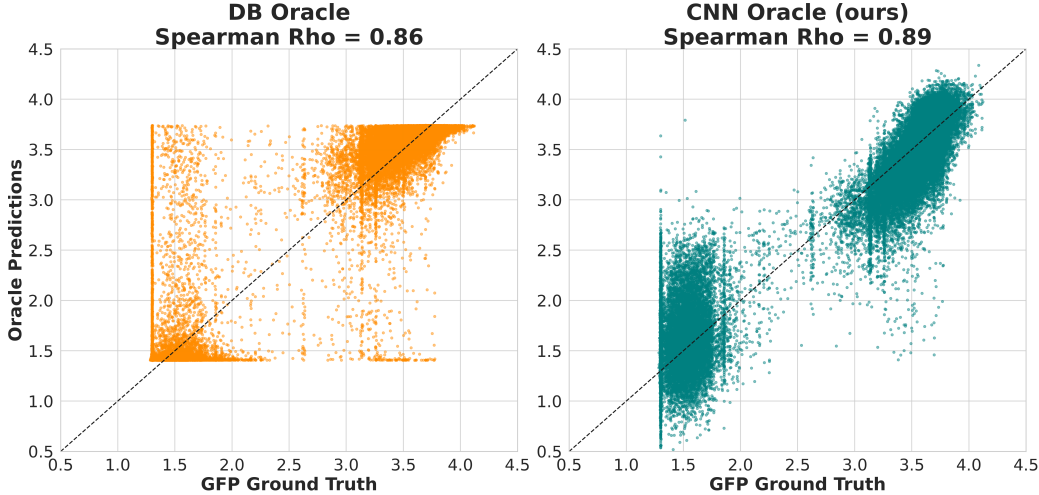


Figure 6: Design-bench (DB) and our CNN oracle Comparison. DB oracle exhibits strange behaviour of predictions being thresholded at min/max values. Yet the Spearman correlation (ρ) is high and closely matches the reported ρ in [36]. Our simpler CNN oracle on the other hand is able to fit the ground truth data more accurately (higher ρ) and has less false positives. We perform experiments with both oracles in our work.

459 B Additional methods

460 In this section, we provide additional details of Graph-based Smoothing (GS) and algorithms.
 461 Algorithm 2 describes pseudo code for Bi-level Gibbs (BiG) sampling introduced in Section 2.2.
 462 Algorithm 3 describes pseudo code for GS described in Section 2.3 with algorithm 4 and algorithm 5
 463 as sub-routines used in GS.

464 We now describe remaining details of GS. We use the same optimization algorithm from Lu et al.
 465 [17] to solve eq. (4) which is reproduced here. As a reminder, eq. (4) is

$$\mathcal{S}^* = \arg \min_{\hat{\mathcal{S}}} \|B\hat{\mathcal{S}}\|_1 + \gamma \|\hat{\mathcal{S}} - \mathcal{S}\|_1.$$

466 Solving the above, a combination of L_1 -optimization problems, is notoriously difficult therefore we
 467 introduce an auxiliary variable F with same dimensions as \mathcal{S} and instead solve the following,

$$\min_{\hat{\mathcal{S}} \geq 0, U} \frac{1}{2} \|\hat{\mathcal{S}} - F\|_{\mathcal{F}}^2 + \lambda \|BF\|_1 + \gamma \|\hat{\mathcal{S}} - \mathcal{S}\|_1 \quad (6)$$

468 where $\|\cdot\|_{\mathcal{F}}$ is the Frobenius norm. The purpose of F is to disentangle $\hat{\mathcal{S}}$ from being part of two
 469 L_1 -norm terms such that we can solve smaller sub-problems. The new term $\|\hat{\mathcal{S}} - F\|_{\mathcal{F}}^2$ enforces F to
 470 be close to $\hat{\mathcal{S}}$. Equation (6) can be solved with two easier sub-problems.

$$F^* = \arg \min_F \frac{1}{2} \|F - \hat{\mathcal{S}}^*\|_{\mathcal{F}}^2 + \lambda \|BF\|_1 \quad (7)$$

$$\hat{\mathcal{S}}^* = \arg \min_{\hat{\mathcal{S}} \geq 0} \frac{1}{2} \|\hat{\mathcal{S}} - F^*\|_{\mathcal{F}}^2 + \gamma \|\hat{\mathcal{S}} - \mathcal{S}\|_1 = \text{SoftThreshold}(F^*, \mathcal{S}, \gamma) \quad (8)$$

471 where $\hat{\mathcal{S}}^* = \mathcal{S}$ initially and λ is a Lagrange multiplier. Equation (8) has a closed-form solution using
 472 the soft-threshold function (`SoftThreshold`) which is defined in eq. (9) of Lu et al. [17] (we omit it
 473 here for ease of exposition). Equation (7) requires iterative optimization due to the computational
 474 intractability of B . This is overcome by a dimensionality reduction,

$$F = U_V A, \quad U_V = [u_1 | \dots | u_{50}]$$

475 where U_V is a matrix whose columns are the 50 smallest eigenvectors of \mathcal{L} and $A = \{a_{ij}\}$ is the
 476 reconstruction coefficients. Lu et al. [17] let the number of projected eigenvectors be a hyperparameter
 477 but we found 50 to work well. Equation (7) can be reformulated as,

$$\begin{aligned} F^* &= \arg \min_A \frac{1}{2} \|U_V A - \hat{\mathcal{S}}^*\|_{\mathcal{F}}^2 + \lambda \|BU_V A\|_1 \\ &= \arg \min_A \sum_j \underbrace{\frac{1}{2} \|U_V A_{\cdot j} - \hat{\mathcal{S}}_{\cdot j}^*\|_{\mathcal{F}}^2 + \lambda \|BU_V A_{\cdot j}\|_1}_{(*)} \end{aligned} \quad (9)$$

478 where $\hat{\mathcal{S}}_{\cdot j}^*$ and $A_{\cdot j}$ denotes the j -th column of $\hat{\mathcal{S}}^*$ and A respectively. Lu et al. [17] proved solving
 479 eq. (9) is equivalent to solving eq. (8) under certain conditions but is a good approximation otherwise.
 480 Each $(*)$ can be solved independently,

$$\begin{aligned} (*) &= \arg \min_{A_{\cdot j}} \frac{1}{2} \|U_V A_{\cdot j} - \hat{\mathcal{S}}_{\cdot j}^*\|_2^2 + \lambda \|BU_V A_{\cdot j}\|_1 \\ &= \arg \min_{A_{\cdot j}} \frac{1}{2} \|U_V A_{\cdot j} - \hat{\mathcal{S}}_{\cdot j}^*\|_2^2 + \lambda \sum_i \Sigma_{ii}^{\frac{1}{2}} |a_{ij}| \end{aligned} \quad (10)$$

481 See Lu et al. [17] for derivation. Recall Σ_{ii} is the i th eigenvalue. Equation (10) can be solved using
 482 off-the-shelf solvers. We can now solve eq. (7) and eq. (8) in iterative fashion. We set a number of
 483 rounds (1000 in our case) and alternate between solving eq. (8) and eq. (7). The full algorithm is
 484 provided as `NoiseReduction` in Algorithm 4 which follows Algorithm 1 of Lu et al. [17].

Algorithm 2 BiG: Bi-level Gibbs

Require: Parent sequence: x **Require:** Predictor weights: θ **Require:** Sampling temperature: τ **Require:** Upper mutation limit: M **Require:** Number of sequences to sample: N_{prop}

```
1:  $\mathcal{X}' \leftarrow \emptyset$ 
2: for  $m = 1, \dots, M$  do                                ▷ Enumerate number of mutations to sample.
3:   for  $i = 1, \dots, m$  do                                ▷ Sample each mutation.
4:      $\ell \sim q(\cdot|x)$                                     ▷ Sample index eq. (2)
5:      $(x')_\ell \sim q(\cdot|x, \ell)$                           ▷ Sample token eq. (2)
6:   end for
7:   if accept using eq. (3) then
8:      $\mathcal{X}' \leftarrow \mathcal{X}' \cup \{x'\}$ 
9:   end if
10: end for
11: Return  $\mathcal{X}'$                                           ▷ Return accepted sequences.
```

Algorithm 3 GS: Graph-based Smoothing

Require: Sequences: \mathcal{X} **Require:** Predictor weights: θ_0 **Require:** Number of perturbations: N_{perturb} **Require:** Number of neighbors: N_{neigh} **Require:** Sparsity weight: γ

```
1:  $V \leftarrow \text{Perturb}(\mathcal{X}, N_{\text{perturb}})$                 ▷ Construct graph nodes algorithm 5.
2:  $W \leftarrow \{\omega_{ij} = 1/\text{dist}(v_i, v_j) : v_i, v_j \in V\}$     ▷ Construct similarity matrix.
3:  $E \leftarrow \text{NearestNeighbor}(\mathcal{X}; W, N_{\text{neigh}})$     ▷  $N_{\text{neigh}}$  nearest neighbor graph edges based on  $W$ .
4:  $\mathcal{S} \leftarrow \{f_{\theta_0}(v) \mid v \in V\}$                 ▷ Node attributes
5:  $\mathcal{S}^* \leftarrow \text{NoiseReduction}(V, E, \mathcal{S}, \gamma)$     ▷ Solves eq. (4) with algorithm 4
6:  $\mathcal{D} \leftarrow (V, \mathcal{S}^*)$ 
7:  $\theta \leftarrow \arg \max_{\tilde{\theta}} \mathbb{E}_{(x,y) \sim \mathcal{D}} [(y - f_{\tilde{\theta}}(x))^2]$     ▷ Train on smoothed dataset.
8: Return  $\theta$ 
```

Algorithm 4 NoiseReduction: follows Algorithm 1 in Lu et al. [17]

Require: Nodes: V **Require:** Edges: E **Require:** Noisy labels: \mathcal{S} **Require:** Sparsity weight: γ

```
1:  $\mathcal{L} \leftarrow$  Compute normalized Laplacian on graph  $G = (V, E)$ .
2:  $V_E \leftarrow$  Find 50 smallest eigenvectors* of  $\mathcal{L}$ .
3:  $\hat{\mathcal{S}} \leftarrow \mathcal{S}$ 
4: for  $i = 1 \dots 1000$  optimization rounds do
5:    $A^* \leftarrow$  Solution in eq. (9) using off-the-shelf solvers.
6:    $F^* \leftarrow U_V A^*$ 
7:    $\hat{\mathcal{S}}^* \leftarrow \text{SoftThreshold}(F^*, \mathcal{S}, \gamma)$ 
8: end for
9: Return  $\hat{\mathcal{S}}^*$ 
```

Algorithm 5 Perturb

Require: Sequences: \mathcal{X} **Require:** Number of perturbations: N_{perturb}

```

1:  $V \leftarrow \mathcal{X}$ 
2: while  $|V| < (|\mathcal{X}| \cdot N_{\text{perturb}})$  do
3:    $x \sim \text{Uniform}(V)$ 
4:    $x' \leftarrow \text{Random point mutation to } x$ 
5:    $V \leftarrow V \cup \{x'\}$ 
6: end while
7: Return  $V$ 

```

C Additional results

AAV evaluation. Our experiments use GFP (Section 3) due to its long sequence length (237 residues) resulting in a large search space and wide coverage of fitness measurements (56,806 data points). The proposed medium and hard difficulties were possible because of GFP’s measurements up to 15 mutations. We sought to experiment on a different fitness landscape to see if (1) mutational gap is a valid criterion for task difficulty and (2) our method isn’t over-optimized for fluorescence fitness in GFP.

A second commonly studied protein fitness dataset is the Adeno-Associated Virus (AAV) [7]. The fitness of the AAV capsid protein is its ability to package a DNA payload, i.e. gene delivery. This dataset encompasses 201,426 subsequences of the AAV2 wild-type known to be important for gene delivery based on the structure. Of these, we consider 44,156 subsequences of length 28 and consist solely of substitutions⁶. The benchmark difficulties on AAV are defined the same way in the GFP task. The same oracle architecture is used to train on all 44,156 sequences while BiGGS uses the same hyperparameters as in GFP.

Results from evaluating BiGGS and baselines on AAV is presented in Table 4. BiGGS outperforms all baselines on this dataset as well, demonstrating its robustness across another protein fitness landscapes. The improvement using BiGGS over baselines is lower for AAV compared to GFP. This could be explained due to the smaller search space (28 residues) than GFP (237 residues) suggesting AAV is easier. We observe a decrease in fitness performance across all methods to suggest mutational gap is a valid criterion for task difficulty.

Table 4: AAV optimization results (our oracle).

AAV Task		Method						
Difficulty	Metric	GFN-AL	CbAS	Adalead	BO-qei	CoMs	PEX	BiGGS
Easy	Fit.	0.02 (0.0)	0.53 (0.0)	0.53 (0.0)	0.42 (0.2)	0.31 (0.1)	0.47 (0.0)	0.57 (0.0)
	Div.	12.5 (0.8)	6.7 (0.6)	5.8 (0.1)	10.1 (7.2)	4.3 (1.71)	2.1 (0.1)	7.1 (0.2)
	Nov.	20.9 (1.2)	5.6 (0.6)	5.0 (0.0)	8.4 (7.6)	5.7 (0.6)	1.0 (0.0)	2.1 (0.2)
Medium	Fit.	0.01 (0.0)	0.35 (0.0)	0.42 (0.0)	0.30 (0.1)	0.32 (0.1)	0.36 (0.0)	0.53 (0.0)
	Div.	15.9 (1.1)	13.7 (0.4)	7.7 (1.4)	17.3 (3.6)	10.2 (3.7)	2.2 (0.2)	9.0 (0.1)
	Nov.	21.8 (0.7)	7.8 (0.5)	1.0 (0.0)	4.2 (9.4)	9.0 (2.0)	1.0 (0.0)	7.8 (0.5)
Hard	Fit.	0.00 (0.0)	0.31 (0.0)	0.00 (0.0)	0.28 (0.0)	0.31 (0.2)	0.27 (0.0)	0.45 (0.0)
	Div.	18.5 (1.2)	15.1 (0.7)	24.7 (0.1)	20.8 (1.6)	9.2 (4.2)	2.0 (0.0)	12.1 (0.4)
	Nov.	22.2 (1.0)	8.4 (0.5)	22.0 (0.0)	0.0 (0.0)	8.2 (2.6)	1.0 (0.0)	4.0 (0.0)

Results with DB oracle. Table 5 shows results of using the design-bench (DB) oracle compared against our CNN oracle. We find BiGGS is still state-of-the-art on all fitness metrics.

⁶AAV contains sequences of varying length with insertions and deletions. We excluded these since our method and the majority of our baselines cannot handle insertion and deletions.

Table 5: GFP optimization results with design-bench (DB) oracle. Note that diversity, novelty and unique are the same regardless of oracle.

GFP Task		Method						
Difficulty	Metric	GFN-AL	CbAS	Adalead	BO-qei	CoMs	PEX	BiGGS
Easy	Fit.	0.1 (0.0)	0.84 (0.0)	0.86 (0.0)	0.84 (0.0)	0.54 (0.3)	0.81 (0.0)	0.86 (0.0)
	Div.	27.9 (2.0)	4.5 (0.4)	2.1 (0.2)	5.9 (0.0)	64.6 (86)	2.2 (0.1)	2.2 (0.0)
	Nov.	215 (2.9)	1.4 (0.5)	1.0 (0.0)	0.0 (0.0)	41.1 (43)	1.0 (0.0)	1.0 (0.0)
Med.	Fit.	0.1 (0.0)	0.0 (0.0)	0.75 (0.0)	0.0 (0.0)	0.45 (0.3)	0.74 (0.0)	0.86 (0.0)
	Div.	30.9 (2.7)	9.2 (1.5)	9.3 (0.1)	20.1 (7.1)	96.8 (92.9)	2.0 (0.0)	4.0 (0.2)
	Nov.	212 (2.0)	7.0 (0.7)	1.0 (0.0)	0.0 (0.0)	82.3 (82.1)	1.0 (0.0)	5.9 (0.2)
Hard	Fit.	0.1 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.21 (0.2)	0.0 (0.0)	0.58 (0.1)
	Div.	29.3 (2.5)	98.7 (16)	6.6 (0.6)	84.0 (7.1)	152 (21)	2.0 (0.0)	4.1 (0.1)
	Nov.	212 (2.0)	46.2 (9.4)	1.0 (0.0)	0.0 (0.0)	160 (55)	1.0 (0.0)	7.0 (0.0)

507 D Additional analysis

508 We include a supporting plot to show the fitness difference when we consider all small mutations
509 ($M < 3$) vs. large mutations ($M \geq 3$).

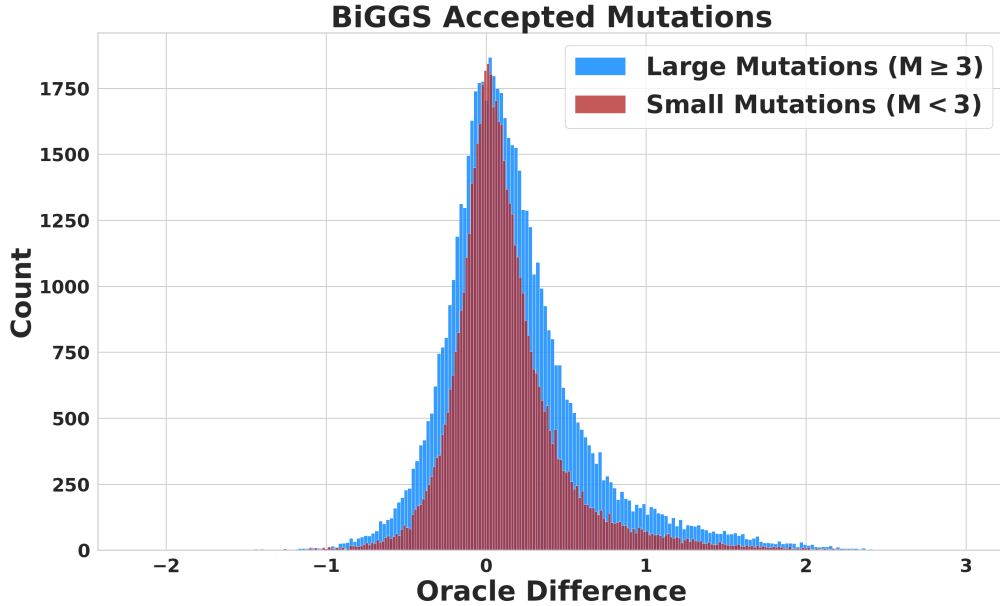


Figure 7: Mutational trajectory analysis for large vs. small accepted mutations. Among mutations that were accepted during the course of running BiGGS, large ($M \geq 3$) mutations resulted in more substantial fitness increases (i.e. > 1) than small ($M < 3$) mutations.