A ADDITIONAL GFP ANALYSIS

Design-bench difficulty. Prior works have used the GFP task introduced by design-bench (DB), a suite of model-based reinforcement learning tasks (Trabucco et al., 2022), which samples a starting set of 5,000 sequences from the 50-60th percentile fitness range. However, we found this task to be too easy in the sense only one mutation was required from sequences in the training set to achieve the 99th percentile. We quantify this difficulty using the mutational gap described in eq. (6). Our proposed medium and hard difficulties (Appendix C.2.1) require many more mutations to reach the top fitness percentile, see Figure 4. Similar issues may be present in other benchmarks.

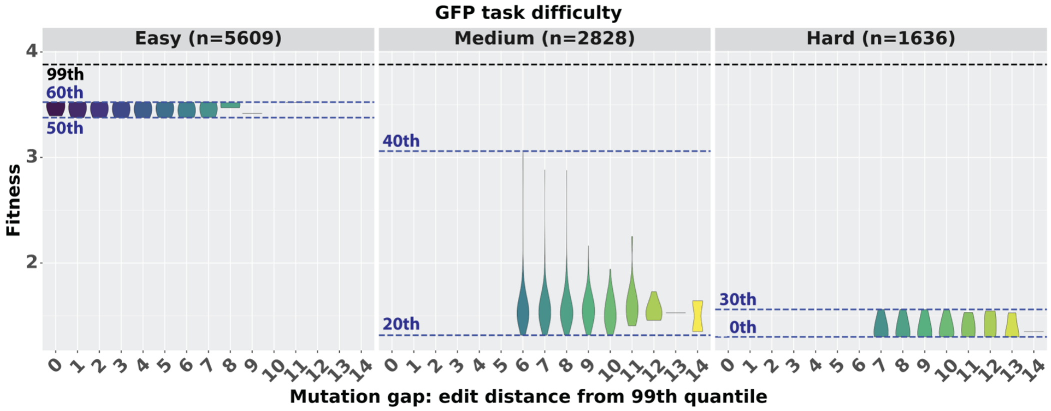

Figure 4: **Easy** is taken from design-bench where sequences between the 50-60th percentile are used in training regardless of edit distance to sequences in the 99th percentile. Data leakage is present due to multiple measurements that allows the wild-type and other top sequences to be included during training. **Medium** filters the training dataset to have sequences in the 20-40th percentile and be 6 or more mutations away from anything in the top 99th percentile. **Hard** similarly filters for sequences in at most the 30th percentile and 7 or more mutations away.

B ADDITIONAL METHODS

B.1 CNN ARCHITECTURE

We utilize a 1D convolutional neural network (CNN) architecture in our model and oracle. The CNN takes in a one-hot encoded sequence as input then applies a 1D convolution with kernel width 5 followed by max-pooling and a dense layer to a single node that outputs a scalar value. It uses 256 channels throughout for a total of 157,000 parameters. Despite its simplicity, we find the CNN to outperform Transformers. Indeed, this corroborates the results in Dallago et al. (2021) that a simple CNN can be effective in low data regimes.

Training is performed with batch size 1024, ADAM optimizer (Kingma & Ba, 2014) (with $\beta_1 = 0.9, \beta_2 = 0.999$), learning rate 0.0001, and 50 epochs, using a single A6000 Nvidia GPU.

B.2 METRICS

We provide mathematical definitions of each metric. Note g_ϕ is the evaluator trained to predict the approximate fitness as a proxy for experimental validation.

- **(Normalized) Fitness** = $\texttt{median}(\{\xi(\hat{x}_i; Y^*)\}_{i=1}^{N_{\text{samples}}})$ where $\xi(\hat{x}; Y^*) = \frac{g_\phi(\hat{x}_i) - \texttt{min}(Y^*)}{\texttt{max}(Y^*) - \texttt{min}(Y^*)}$ is the min-max normalized fitness based on the lowest and highest known fitness in Y^*.
- **Diversity** = $\texttt{median}(\{\texttt{dist}(x, \tilde{x}) : x, \tilde{x} \in \hat{X}, x \neq \tilde{x}\})$ is the average sample similarity.
- **Novelty** = $\texttt{median}(\{\eta(\hat{x}_i; X)\}_{i=1}^{N_{\text{samples}}})$ where $\eta(x; X) = \texttt{min}(\{\texttt{dist}(x, \tilde{x}) : \tilde{x} \in X^*, \tilde{x} \neq x\})$ is the minimum distance of sample x to any of the starting sequences X.

Algorithm 2 `Smooth`: Graph-based Smoothing

```
Require: Sequences: $X$
Require: Noisy model weights: $\tilde{\theta}$
$V, E \leftarrow \texttt{CreateGraph}(X)$                              ▷ Construct graph (Algorithm 4).
$L \leftarrow \texttt{GraphLaplacian}(V, E)$                           ▷ Compute graph Laplacian.
$Y \leftarrow [f_{\tilde{\theta}}(x_1), \dots, f_{\tilde{\theta}}(x_{N_{\text{nodes}}})]^\top$
$\hat{Y} \leftarrow (\mathbb{I} + \gamma L)^{-1} Y$                    ▷ Compute smoothed fitness labels.
$\theta \leftarrow \arg\max_\theta \mathbb{E}_{(x,\hat{y}) \sim (V,\hat{Y})} \left[(\hat{y} - f_\theta(x))^2\right]$   ▷ Train on smoothed dataset.
Return $\theta$
```

Algorithm 3 `GWG`: Gibbs With Gradients

```
Require: Parent sequences: $X$
Require: Model weights: $\theta$
$X' \leftarrow \emptyset$
for $x \in X$ do
    for $i = 1, \dots, N_{\text{prop}}$ do                 ▷ Number of proposals per sequence.
        $x' \leftarrow x$
        $(i^{\text{loc}}, j^{\text{sub}}) \sim q(\cdot|x)$        ▷ Sample index and token eq. (3)
        $x'_{i^{\text{loc}}} \leftarrow \mathcal{V}_{j^{\text{sub}}}$            ▷ Apply mutation
        if accept using eq. (4) then
            $X' \leftarrow X' \cup \{x'\}$
        end if
    end for
end for
Return $X'$                                           ▷ Return accepted sequences.
```

Algorithm 4 `CreateGraph`

```
Require: Sequences: $X$
$V \leftarrow X$                                        ▷ Construct nodes.
while $|V| \leq N_{\text{nodes}}$ do
    $x \sim \mathcal{U}(V)$
    $x' \leftarrow \texttt{PointMutation}(x)$           ▷ Sample a point mutation uniformly at random.
end while
$E \leftarrow \bigcup_{x \in V} \texttt{kNN}(x; V)$       ▷ Construct edges (Algorithm 5).
Return $(V, E)$
```

Algorithm 5 `kNN`

```
Require: Current node: $x$
Require: All nodes: $V$
$\mathcal{D}(x) \leftarrow \bigcup_{x' \in V/\{x\}} \text{dist}(x', x)$   ▷ Levenstein distance between every pair of sequences.
$\mathcal{X}' \leftarrow \texttt{TopK}(\mathcal{D}(x), V)$             ▷ Compute K closest sequences to $x$.
$\text{E}(x) \leftarrow \bigcup_{x' \in \mathcal{X}'} (x, x')$          ▷ Construct neighborhood around $x$.
Return $\text{E}(x)$
```

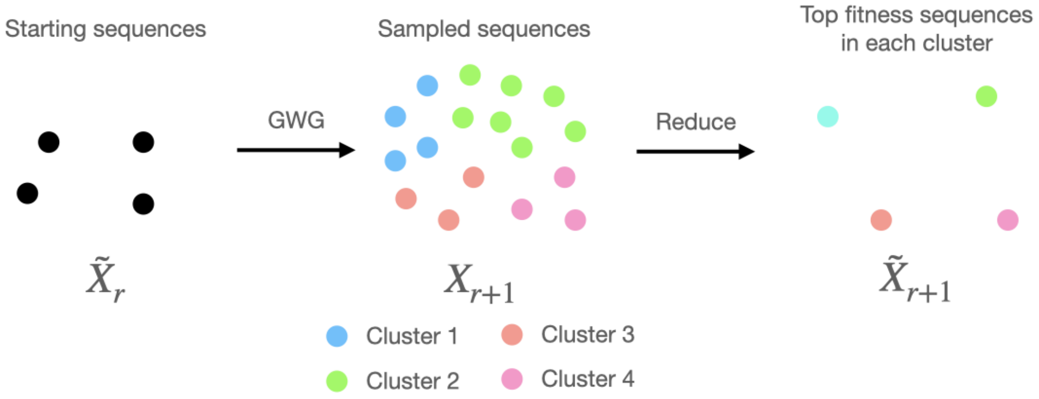

Figure 5: **Illustration of clustered sampling.** $\tilde{V}_r$ is the starting set of sequences for sampling in round r. GWG (Algorithm 3) is ran to generate many sample sequences, V_{r+1}. To control computation, we hierarchically cluster all sampled sequences based on Levenshtein distance and take the top fitness sequence in each cluster, using our trained fitness prediction model f_θ to score each sequence – we refer to this subroutine as `Reduce` (eq. (5)). The top sequences, $\tilde{V}_{r+1}$ are used for the next round.

C ADDITIONAL RESULTS

C.1 SAMPLING TEMPERATURE SWEEP

We determine the effect of different tmperatures γ when running GGS on the hard difficulty for GFP and AAV. All other hyperparameters follow those used in the main results, see Section 4.2. Table 5 shows the results where clearly $\gamma = 0.1$ performs the best for both AAV and GFP.

Table 5: Temperature sweep.

	GFP hard			AAV hard		
Temperature (γ)	Fitness	Diversity	Novelty	Fitness	Diversity	Novelty
0.01	0.65 (0.0)	5.3 (0.8)	7.4 (0.5)	0.45 (0.0)	15.2 (1.1)	9.0 (0.0)
0.1	0.74 (0.0)	3.6 (0.1)	8.0 (0.0)	0.6 (0.0)	4.5 (0.2)	7.0 (0.0)
1.0	0.0 (0.1)	28.2 (0.8)	11.4 (0.5)	0.45 (0.0)	11.9 (0.5)	8.0 (0.0)
2.0	0.0 (0.1)	36.1 (1.0)	13.0 (0.0)	0.33 (0.0)	16.7 (0.9)	8.5 (0.5)

C.2 SMOOTHING ANALYSIS

In this section, we provide further analyses into the effect of smoothing on performance of GGS, extrapolation to unseen data, and acceptance rate of the GWG sampling procedure. Throughout, we use the same parameters $\tau = 0.1, \gamma = 1, r = 15, N_{nodes} = 250,000$ as in the main text.

C.2.1 ADDITIONAL BENCHMARKS

We first define additional benchmarks, one easier, and three harder, for each protein dataset.

Table 6: GFP extra tasks

Difficulty	Range (%)	Gap	$\lvert\mathcal{D}\rvert$
Easy	50th-60th	0	5609
Harder1	$<$ 30th	8	1129
Harder2	$<$ 20th	8	792
Harder3	$<$ 10th	8	397

Table 7: AAV extra tasks

Difficulty	Range (%)	Gap	$\lvert\mathcal{D}\rvert$
Easy	50th-60th	0	4413
Harder1	$<$ 30th	13	1157
Harder2	$<$ 20th	13	920
Harder3	$<$ 10th	13	476

We note that the “easy” GFP task is equivalent to the design-bench baseline that is sometimes used as a benchmark in protein engineering tasks. Due to experimental noise, protein variants are assayed multiple times, and can be assigned multiple fitness values, which means the fitness values of one sequence may occupy a large percentile *range*. In the case of this task, multiple measurements of the wildtype GFP fitness are found in the 50th-60th percentile range. Because WT GFP is also a “top sequence,” this task necessarily has a mutational gap of 0. Due to this leakage, we develop our own benchmarks in the main text, and extend those to AAV.

C.2.2 HOW SMOOTHING AFFECTS PERFORMANCE

The following two tables show how a smoothed model outperforms its unsmoothed counterpart according to our evaluator across all GFP/AAV benchmarks except AAV Harder2 (see $(*)$), and GFP Harder3, where the smoothing was not sufficient to induce successful GWG sampling (see Table 10).

Table 8: Smoothing improves GGS performance on GFP tasks

Difficulty	Smoothed	Median Fitness	Diversity	Novelty
Easy	No	0.05	24.83	13.36
	Yes	**0.84**	5.45	3.51
Medium	No	0.51	10.5	15.4
	Yes	**0.76**	3.7	5.0
Hard	No	0.10	23.02	16.8
	Yes	**0.74**	3.6	8.0
Harder1	No	0.00	22.86	17.0
	Yes	**0.67**	4.45	9.12
Harder2	No	0.00	22.22	16.5
	Yes	**0.60**	5.42	9.82
Harder3	No	0.00	23.02	16.8
	Yes	0.00	15.73	21.2

For the GFP task, our model fails (achieves 0 median fitness) when we restrict the data to the 10th percentile and mutation gap 8 for GFP where $|\mathcal{D}| = 397$.

Table 9: Smoothing improves GGS performance on AAV tasks

Difficulty	Smoothed	Median Fitness	Diversity	Novelty
Easy	No	0.47	2.69	7.81
	Yes	**0.49**	9.18	7.99
Medium	No	0.37	6.60	6.62
	Yes	**0.48**	4.66	5.59
Hard	No	0.33	12.32	13.8
	Yes	**0.60**	4.5	7.0
Harder1	No	0.47	2.69	7.81
	Yes	**0.49**	9.18	7.99
Harder2	No	**0.28***	8.067	2.067
	Yes	0.27	15.98	19.41
Harder3	No	0.25	3.08	5.63
	Yes	**0.38**	7.05	9.486

($*$): The unsmoothed model only outperforms its smoothed counterpart when applying GWG to the unsmoothed model generates only a few unique sequences nearby to the starting set (as evidenced by the low novelty for this benchmark)

For AAV, we find the model is able to still find signal and achieve 0.384 evaluated fitness despite the data being limited to the 10th percentile and mutation gap of 13 where $|\mathcal{D}| = 476$. It is notable, though, that the performance improvements gained from smoothing are smaller than in the case of GFP. Presumably, this is due to the vastly reduced dimension of the AAV sequence space in comparison to that of GFP, which may result in a neural network to learn a smoother landscape without any regularization.

C.2.3 HOW SMOOTHING AFFECTS EXTRAPOLATION + SAMPLING

The following tables show the benefits of smoothing on extrapolation to held out ground truth experimental data, up to a certain difficulty benchmark, as well as how smoothing vastly improves the acceptance rate for the GWG sampling procedure.

Table 10: Smoothing improves extrapolation and GWG sampling, up to GFP Harder3

Difficulty	Smoothed	Train MAE	Holdout MAE	Acc. Rate
Easy	No	**0.03**	0.99	0.02
	Yes	0.71	**0.61**	**0.99**
Medium	No	**0.10**	1.29	0.61
	Yes	0.20	**0.88**	**0.62**
Hard	No	**0.06**	1.44	0.01
	Yes	0.15	**0.93**	**0.43**
Harder1	No	**0.07**	1.39	0.01
	Yes	0.15	**0.94**	**0.43**
Harder2	No	**0.01**	1.41	0.01
	Yes	0.12	**0.90**	**0.59**
Harder3	No	0.01	**1.41**	0.01
	Yes	0.01	1.42	0.01

Table 11: Smoothing improves extrapolation up to AAV Hard and GWG sampling on all AAV tasks

Difficulty	Smoothed	Train MAE	Holdout MAE	Acc. Rate
Easy	No	**0.28**	2.82	0.01
	Yes	1.76	**2.28**	**0.99**
Medium	No	**0.35**	3.12	0.01
	Yes	0.44	**2.76**	**0.82**
Hard	No	**0.48**	3.70	0.30
	Yes	0.55	**3.09**	**0.78**
Harder1	No	**0.66**	**3.99**	0.01
	Yes	0.69	4.24	**0.47**
Harder2	No	**0.56**	**4.13**	0.01
	Yes	0.58	4.37	**0.55**
Harder3	No	0.47	**4.58**	0.01
	Yes	0.47	4.59	**0.64**

For each benchmark category, we evaluated the impact of smoothing on extrapolation abilities by analyzing the Mean Absolute Error (MAE) of the models on that benchmark's training and holdout datasets from the experimental ground truth. The effectiveness of smoothing was indicated by reduced MAE values on the holdout set. We also find that the MAE on the training set is lower for the unsmoothed models, as expected. In line with the results of the previous section, the effect of smoothing is reduced for AAV. As task difficulty increases, for both proteins, the effectiveness of smoothing on extrapolation decreases, which we expect as any signal leading from the training set to the fitter sequences gets obscured as training set size decreases.

Finally, we note that in every case except two, smoothing dramatically increases acceptance rate of the GWG sampling procedure, which aligns with the inversely proportional relationship between smoothness of the energy function and sampling efficiency. In the case of the hardest GFP task, even the the smoothed model had overfit to the training set. As for the GFP medium task, we suspect that this particular section of the experimental dataset allowed the unsmoothed model to learn a smooth landscape initially.