Improving few-shot learning-based protein engineering with evolutionary sampling Supplementary Material

M. Zaki Jawaid
EpiCRISPR Biotechnologies
zaki.jawaid@epic-bio.com

Aayushma Gautam
EpiCRISPR Biotechnologies
aayushma.gautam@epic-bio.com

T. Blair Gainous
EpiCRISPR Biotechnologies
blair.gainous@epic-bio.com

Daniel O. Hart
EpiCRISPR Biotechnologies
dan.hart@epic-bio.com

Robin W. Yeo[†]
EpiCRISPR Biotechnologies
robin.yeo@epic-bio.com

Timothy P. Daley [*†]
timy.pdaley@gmail.com

1 Supplementary Methods and Results

1.1 Training Data

An initial library of 34217 85aa (85 amino acid long) putative modulator peptides from diverse biological origins were experimentally screened for their ability to activate a synthetic genetic locus using a nuclease inactive Cas system. In this system the modulator peptide is fused to the Cas molecule and directed towards the synthetic genetic locus with gRNAs [1].

The 85aa length was chosen because of technical limitations in ordering a large number of sequences from Twist Biosciences. This was the maximum length that we could obtain in a cost effective manner after including sequencing primers and barcodes.

In an independent follow-up screen, a subset of these sequences were re-tested, resulting in 173 sequences that we classified as validated gene activators ("positive hits"), giving a hit rate of 0.51%.

The low number of positive examples presents a particular problem for ML-guided engineering because it is difficult to ensure that the fitness function will extrapolate well outside the small neighborhood of the positive examples in the training set.

1.2 Featurization

We treat each peptide sequence as a string of 20 unique amino acid characters. Under this characterization, each peptide sequence needs to be numerically featurized to be used as input to train supervised classification models. We explored several featurizations.

OneHot Encoding The simplest form of peptide featurization, OneHot encoding, transforms each of the 20 possible amino acid characters into a unique 20-dimensional vector with all values equal to zero except the amino acid index position with a 1. For our training set, this results in a list of 34217 arrays of size 85x20.

[*]Formerly affiliated with EpiCrispr Biotechnologies
[†]Co-corresponding Author

NeurIPS 2023 Generative AI and Biology Workshop.

ESM-2 Embedding Evolutionary Scale Model 2 (ESM-2) is a transformer-based protein language model trained on over 60M sequences from the UniRef database [2, 3]. Training on such a huge diversity of protein sequences allows large protein language models (LPLMs) like ESM-2 to implicitly capture a variety of evolutionary, biochemical, and biophysical features that can be leveraged to improve functional predictions via transfer learning. In this study we use the 650M parameter 33-layer ESM-2 model which converts each 85aa peptide sequence into an array of size 85x1280. We additionally tested the larger 3B and 15B ESM-2 models but ultimately decided to use the 650M parameter model embeddings since the lower parameter embedding performed comparably well. Furthermore, the computational cost of MHMCS and EMCS is dominated by the embeddings, therefore the smaller model enabled faster exploration.

Mean featurization One transformation of OneHot encoding is to transform them to a "bag of words", or rather a "bag of embeddings" featurization [4, 5]. For example, an 85x20 amino acid OneHot encoding will be transformed to a 1x20 encoding of average amino acid content. Similarly, the 85x1280 ESM-2 embeddings will be transformed to a 1x1280 of average embeddings.

1.3 Model Training

OneHot and ESM-2 protein sequence embeddings were then used as input to train two classification models: a decision-tree based model (XGBoost) [6], where we flatten the features by taking the mean featurization; and a 1-dimensional convolutional neural network (CNN) [7]. Due to the extreme class imbalance, during all training we upsampled our positive hits so they accounted for 30% of the observations. All models were trained using using AWS Sagemaker Accelerated Computing P3 instances with NVIDIA V100 Tensor Core GPUs with PyTorch (v1.13). Model training results (PR-AUC and F1 scores) can be found in Supplementary Table 1 and Supplementary Table 2. For both XGBoost and CNN models, we observed a significant increase in PR-AUC and F1 scores with the use of transfer learning with the ESM-2 embeddings compared to OneHot encoding.

XGBoost We performed hyperparameter tuning with a grid search (parameters: "learning_rate", "n_estimators", "gamma", "subsample", "max_depth", "min_child_weight") and determined the top model based on PR-AUC score from 5-fold stratified cross-validation. XGBoost models were trained with tree_method="gpu_hist" and objective="binary:logistic". The top ESM-2 model included 500 trees with a max depth of 4 and was trained with the following parameters: learning_rate=0.1, gamma=0, subsample=0.75, min_child_weight=0. The top model was then re-trained on the entire training set with upsampling of the minority class (the positive hits) and used as the XGBoost fitness function for exploring the fitness landscape.

CNN Model architecture for the 1-dimensional CNN models included 3 convolutional layers with rectified linear unit (ReLU) activation functions, with batch-normalization, dropout, and max-pooling, followed by two densely connected layers. We employed hyperparameter tuning with a grid search (parameters: "n_epochs", "weight_decay", "learning_rate", "batch_size", "dropout", "conv_size") and determined the top model based on F1 score from 5-fold stratified cross-validation. CNN models were trained using the AdamW optimizer [8] with "BCEWithLogitLoss" loss function. The top ESM-2 model was trained for 30 epochs with a batch size of 16 using the AdamW optimizer with parameters learning_rate=1e-06 and weight_decay=0.05; this resulted in a network architecture with three convolutional layers containing 100 convolutional filters (kernel_size=10, stride=1) with batch-normalization and max-pooling followed by two densely-connected layers (200 neurons in first hidden layer and 10 neurons in second hidden layer). The top model was then re-trained on the entire training set with upsampled positive hits and used as the CNN fitness function for exploring the fitness landscape.

We believe that the power of our approach lies in the combination of transfer learning via LPLMs and EMCS. Since LPLMs are trained on an immense number of diverse protein sequences, modern LPLM embeddings implicitly contain a wealth of features describing a protein's biochemical, biophysical, evolutionary, and even 3-dimensional structure information [2]; as such, we reason that LPLM embeddings of novel proposed sequences are capable of capturing the predicted functional consequences of genetic crossovers from EMCS such that swaps resulting in misfolded or non-active proteins are assigned low fitness and thus not selected by EMCS. Conversely, potential swaps and

Model performance by PR-AUC			
Embedding	Fitness Model	PR-AUC (5-fold CV)	PR-AUC (30% test set)
OneHot	XGB	0.487	0.549
OneHot	CNN	0.331	0.303
ESM-2	XGB	**0.573**	**0.629**
ESM-2	CNN	0.533	0.576

Supplementary Table 1: PR-AUC scores based on 5-fold cross-validation and testing on 30% holdout test set for top XGBoost and CNN models with OneHot and ESM-2 embeddings respectively.

Model performance by F1 score			
Embedding	Fitness Model	F1 score (5-fold CV)	F1 score (30% test set)
OneHot	XGB	0.450	0.472
OneHot	CNN	0.457	0.432
ESM-2	XGB	0.590	0.615
ESM-2	CNN	0.562	0.519

Supplementary Table 2: F1 scores based on 5-fold cross-validation and testing on 30% holdout test set for top XGBoost and CNN models with OneHot and ESM-2 embeddings respectively.

domains that can act synergistically will be assigned a high fitness by our semi-supervised transfer learning-based model and selected for by EMCS, even if those domains are not evolutionarily related.

In contrast, since GANs and diffusion models sample from a low-dimensional latent space, and then pass the sample through the model to obtain the proposed sequence, only sequences that are close to the training data in latent space can be designed by these methods; additionally, there's no guarantee that high synergy domains will be close in the latent space (especially if they're not evolutionarily related) limiting the potential diversity of sequences that can be proposed by generative algorithms trained on limited and skewed training data.

1.4 Sampling

All sampling runs were implemented on AWS Elastic Cloud Compute 2 g5.2xlarge instances with NVIDIA A10 Tensor Core GPU's. When using the ESM-2_650M parameter model to generate sequence embeddings, we found that the g5.2xlarge instances could evaluate roughly 50 independent 85aa sequences per second. We verified that the vast majority of the computational time was spent on sequence featurization.

Metropolis Hastings Monte Carlo Search (MHMCS) Algorithm 1 summarizes our implementation of MHMCS, with our choice of default parameters given in Supplementary Table 3. In MHMCS, the search space for a given molecule is explored via the iterative application of a mutation operator. The mutation operator proposes a new molecule via making a random number of single point mutations to the current molecule. The fitness of this new molecule is then evaluated, and the new molecule is accepted with probability $min(1, r_{mh})$, where r_{mh} is the standard metropolis hasting criterion.

The Metropolis-Hastings Monte Carlo Search (MHMCS) method [9–11] is the standard method for the exploration of high-dimensional discrete landscapes, including those generated by machine learning algorithms [12–14]; however MHMCS suffers from an inability to escape deep local optima. Other approaches for sampling the sequence space include gradient-based sampling [15, 16], and modified Gibbs sampling [17]. While powerful, these approaches require significant computation near the local neighborhood of the fitness landscape and are therefore too computationally intensive for sequences of any significant length, (e.g. gradient-based methods require $19 \cdot L$ computations and Gibbs requires L computations per iteration).

Evolutionary Monte Carlo Search (EMCS) Algorithm 2 details our implementation of EMC as a search tool with our choice of default parameters given in Supplementary Table 3. EMCS is highly versatile and allows for vastly different exploratory behaviors compared to traditional sampling techniques due to the implementation of a custom temperature ladder, as well as predefined

crossover, mutation, and swap rates [18]. These parameters can be tuned for more efficient exploration depending on the specific design problem and the complexity of the discrete high-dimensional fitness landscape.

Each primary iteration in EMCS can potentially change the state of the algorithm in one of three ways, namely, point mutations, swaps, and crossovers between different temperature chains. The possibility of the acceptance of sub-optimal moves for each of these three classes depends on how we define the acceptance criterion.

We use r_{mh}, the standard Boltzmann Metropolis-Hastings acceptance criterion, for mutation-based moves, which as described earlier, accepts sub-optimal moves with probability weighted by the ratio of the proposed fitness to the current fitness. For swaps between two consecutive chains, we use r_{re}, the standard parallel tempering criterion also used in [18]. Using this criterion, any proposed swap in which the higher fitness sequence in proposed to move to the lower temperature chain is accepted. In a swap in which a higher fitness sequence is proposed to move to the higher temperature, the move is accepted with probability inversely proportional to the magnitude of the difference of the temperatures of the two chains, as well as the fitness of the two sequences. Finally, the crossover criterion r_c, also adapted from [18], accepts crossover moves taking into account the difference in fitness between the set of old and new sequences, in addition to the difference of temperatures of the two chains involved in the crossover.

Crossover Criterion The crossover criterion ($min(1, r_c)$) is used to determine whether the set of new sequences i_2, j_2 generated by crossing over two sequences i_1, j_1 are accepted or rejected. Here, r_c is defined as:

$$r_c = \exp\left(\frac{f(i_2) - f(i_1)}{T_i} - \frac{f(j_2) - f(j_1)}{T_j}\right) \tag{1}$$

With the condition that $T_i \leq T_j$, and the sequences i_2, j_2, and i_1, j_1 are ordered such that $f(i_1) \geq f(j_1)$ and $f(i_2) \geq f(j_2)$. The proposed sequences i_2, j_2 are accepted with probability $min(1, r_c)$ and assigned to temperatures T_i and T_j respectively.

By rearranging the terms in r_c, we can see that r_c will always be greater than 1 if

$$\frac{f(i_2)}{T_i} - \frac{f(j_2)}{T_j} > \frac{f(i_1)}{T_i} - \frac{f(j_1)}{T_j} \tag{2}$$

This condition always holds true for optimal moves. Including the move in which the fitness of both sequences improves as in $f(i_1) < f(i_2)$ and $f(j_1) < f(j_2)$, thereby moving both sequences to a region with a higher fitness landscape. If $f(i_1) < f(i_2)$ but $f(j_1) > f(j_2)$, that is, the fitness of one sequence improves while the fitness of of the other sequence is worsened, then the proposed move is only guaranteed to be accepted if the condition in Supplementary Equation 2 holds. If the condition in Supplementary Equation 2 does not hold, then the proposed move is sub-optimal and is only accepted with probability equal to that defined in Supplementary Equation 1. In general, the likelihood of acceptance of a sub-optimal moves rapidly decreases as the temperature difference $T_j - T_i$ increases, thereby suggesting that a large temperature difference penalizes sub-optimal moves more than a smaller temperature difference.

The immensity of the protein sequence space coupled with the computational cost of embedding a protein using LPLMs like ESM-2 called for an efficient sampling algorithm that could escape local optima without compromising resolution. The EMC algorithm is ideally suited to this use case, as the incorporation of a temperature ladder allows for the simultaneous existence of multiple acceptance ratios. Furthermore, the genetic crossover steps allow for more efficient exploration of the fitness landscape, as shown by sequence diversity and average entropy change per iteration of MHMCS vs. EMCS.

This method generalizes well to proteins of any length with one caveat. The changes proposed by the crossover operator would be too large for proteins of significant length (>150 aa residues), yielding a set of sequences with edit distances too high from the starting sequences. Given the highly rugged and sporadically peaked nature of the fitness landscape, the crossover operator is significantly more likely to propose low fitness sequences that will be rejected, thus effectively reducing the EMCS

Algorithm 1 Metropolis Hastings Monte Carlo Search (MHMCS)

Initialize a peptide sequence i and assign a temperature T_i.
Set minimum and maximum number of iterations k_{min} and k_{max}, respectively.
Set maximum number of mutations μ.
Set convergence condition $f(i) \geq C$ where $f(i)$ is the fitness of sequence i.
repeat
  Make random point mutations at q loci for sequence i to yield a new sequence s', where $q \in \{1, \ldots, \mu\}$ is chosen uniformly at random
  Accept or reject sequence s' using the metropolis hasting criterion i.e. with probability $min(1, r_{mh})$, where $r_{mh} = \exp(\frac{f(j)-f(i)}{T})$
until $iterations > k_{max}$ **or** $((f_i \geq C)$ **and** $iterations > k_{min})$

algorithm to a parallel tempering (PTP) algorithm. To remedy this problem, we propose replacing the crossover operator with a segment swap operator, where instead of crossing over entire sequences we swap randomly chosen segments between two proteins to yield two new sequences. While this introduces additional tuning parameters (such as length of the segment that is to be swapped, number of segments to be swapped per iteration), it preserves and improves the critical domain swapping feature of EMCS. The acceptance criterion for this operator would be the same criterion that we used for the crossover operator, as in both cases we are evaluating the acceptance of two new sequences i_2, j_2 created from i_1, j_1.

Ablation: EMC-NPT Search (Evolutionary Monte Carlo without Parallel Tempering) This is the EMCS algorithm with the temperature differences and swap rates set to zero.

Ablation: PTP Search (Parallel Tempering) This is the EMCS algorithm with the crossover rates set to zero.

Default Sampling Parameters Supplementary Table 3 summarizes the default parameters used for our sampling runs. Usually, the number of minimum and maximum iterations are determined by first trying a few numbers and then manually adjusting them to find a set of numbers where the algorithm is able to converge to a state with an acceptable fitness. This is done by simply plotting the fitness vs. the number of iterations.

The temperature T decides how liberally non-optimal moves are accepted. A very high temperature will result in a very large proportion of non-optimal moves being accepted (>50-60%). This would make it almost impossible for the algorithm to converge, as the algorithm is unable to reject unfavorable moves when it is close to an optimum. A smaller temperature makes the MHMCS algorithm more conservative, thereby only strictly accepting moves that increase the fitness or decrease it by a very small amount (favoring a higher resolution search). This makes it easier to move towards an optimum, but restricts the search space to directions where the fitness only increases. Therefore the selection of the temperature is an important task, defining the tradeoff between the resolution (depth) and the width (area) of the search. EMCS allows us to search using different temperatures simultaneously. The temperatures were chosen by a combination of analyzing the fitness distribution of the training set (where the fitness scores spanned over multiple orders of magnitude between 10^{-4} and 1), as well as trial and error. The number of chains was set to 4 in order to achieve a good balance between fitness score coverage and computational cost. The algorithm was set to converge when one or more sequences were discovered with a fitness score ≥ 0.95 and the maximum number of iterations was set to 10^5.

Fitness Evaluation: Ensemble Model When sampling using the ensemble model, each new proposed sequence was evaluated using both XGBoost and CNN models, where fitness was defined to be the average of the respective fitness scores of the XGBoost and CNN models. This 'combined' fitness was then used to accept or reject any proposed moves.

Algorithm 2 Evolutionary Monte Carlo Search (EMCS)

Select N chains of amino acid sequences $[0, 1, .., i, .., N]$, with corresponding temperature ladder $[T_1, T_2, .., T_i, .., T_N]$ such that $T_i \geq T_j$ for $i > j$, with fitness $f(i)$
Set crossover rate γ such that $\gamma \subseteq [0, 1)$, and define maximum mutation, crossover, and swap events as μ, α, β
Set minimum and maximum number of iterations k_{min} and k_{max}, respectively
Set convergence condition $f(i) \geq C$ where $f(i)$ is the fitness of sequence i
repeat
  Sample random number p from uniform distribution $[0, 1)$
  if $p > \gamma$ **then**
    Make random point mutations at q loci for each sequence i to yield new set of proposed sequences denoted by j, where $q \in \{1, \ldots, \mu\}$ is chosen uniformly at random
    Update each sequence by accepting or rejecting each proposed sequence using the metropolis hasting criterion i.e. with probability $min(1, r_{mh})$, where $r_{mh} = \exp(\frac{f(j)-f(i)}{T})$
  else
    for number of crossover events α **do**
      Let i_1, j_1 be two random sequences corresponding to temperatures T_i, T_j. Pick a random crossover locus between [2, N-1], where N is the length of the peptide.
      Propose a set of two sequences i_2 and j_2 by crossing over i_1, j_1 at the chosen crossover locus. This results in i_2 being identical to sequence i_1 prior to our crossover locus, and identical to sequence j_1 post our crossover locus. Similarly, j_2 is identical to sequence j_1 prior to the crossover locus, and identical to sequence i_1 post the crossover locus.
      For two temperatures T_i and T_j such that $T_i \leq T_j$, order i_2, j_2, and i_1, j_1 such that $f(i_1) > f(j_1)$ and $f(i_2) > f(j_2)$
      Accept the new set of sequences i_2, j_2 with probability $min(1, r_c)$ where r_c is defined as $r_c = \exp\left(\frac{f(i_2)-f(i_1)}{T_i} - \frac{f(j_2)-f(j_1)}{T_j}\right)$. If accepted, assign i_2, j_2 to chains at temperatures T_i, T_j respectively.
    end for
  end if
  for number of swap events β **do**
    Select two sequences i and j at chains corresponding to T_i, T_j, such that $j = i \pm 1$, and swap their sequence positions such that $i \to j$ and $j \to i$ with probability $min(1, r_{re})$, where r_{re} is defined as $r_{re} = \exp\left(-(f(i) - f(j))\left(\frac{1}{T_i} - \frac{1}{T_j}\right)\right)$
  end for
until $iterations > k_{max}$ **or** $((f_i \geq C)$ for any sequence **and** $iterations > k_{min})$

1.5 Experimental Screening

Library Synthesis For experimental validation, we designed a library of 4600 novel peptide sequences (and included 300 previously validated negative control peptides with random sequences) based on our exploration of the modulator fitness landscape (Supplementary Table 4). The 85aa

Default Sampling Parameters (MHMCS/EMCS)						
Model	Number of Chains	Temperature	μ	γ	α	β
MHMCS	1	2.5×10^{-3}	5	-	-	-
MHMCS	1	2.5×10^{-4}	5	-	-	-
EMCS	4	$2.5 \times [10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]$	5	0.5	1	1
EMC-NPT	4	$2.5 \times [10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}]$	5	0.5	1	-
PTP	4	$2.5 \times [10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]$	5	-	-	1

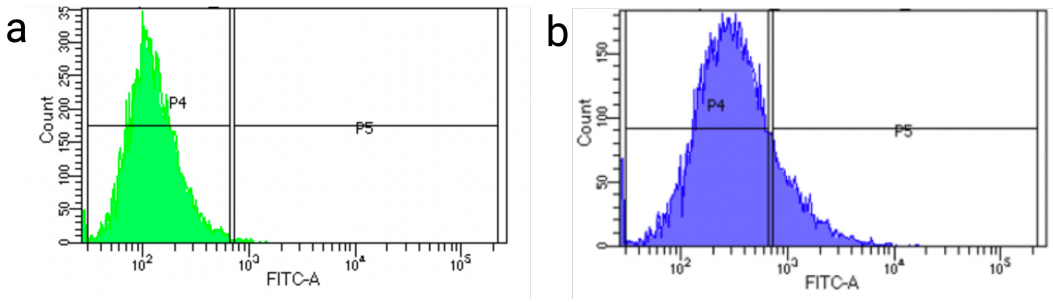
Supplementary Table 3: Default sampling parameters for EMCS and MHMCS, and ablations. Key (as defined in main text): μ (Max. Mutation Events Per Iteration), γ (Crossover Rate), α (Max. Crossover Events Per Iteration), β (Max. Swap Events Per Iteration)

peptide sequences were reverse-translated into 255bp nucleotide sequences using randomized codons to minimize GC bias [19], which were then synthesized as a pooled library by Twist Bioscience. To ensure that we could accurately identify gene activators in our experimental validation, we also included 300 previously validated negative controls (random sequences) to the library.

Validation library composition		
Algorithm	Initialization	Total Sequences
EMCS	Known	1310
EMCS	Random	1290
MHMCS	Random	2000
Negative Controls	n/a	300

Supplementary Table 4: Overall composition of peptides in validation library sorted by choice of sampling algorithm. Initialization column denotes the sampling algorithm's starting sequence as either randomly initialized ("Random"), or known positive hit ("Known").

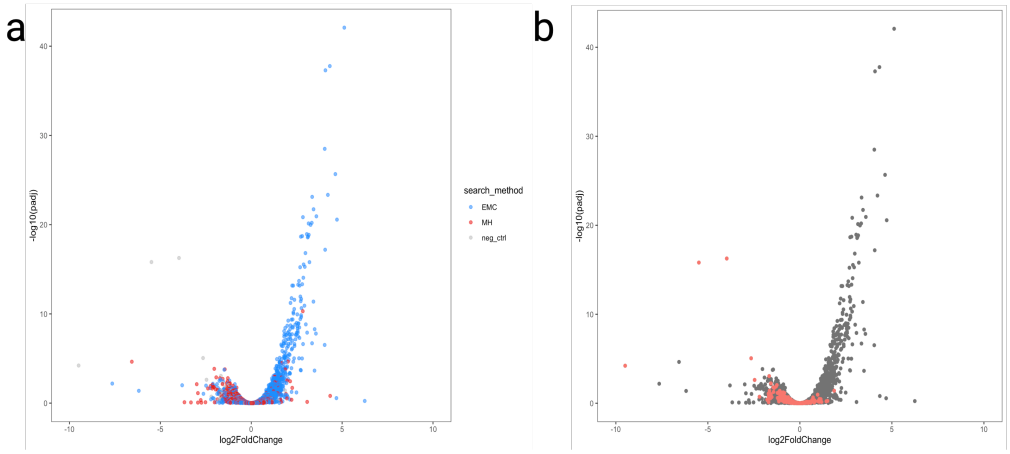
Wetlab Screening The library was cloned downstream of dCasMINI in a lentiviral expression plasmid at high coverage ($\approx$ 300x) and packaged into lentiviral particles (2 lentivirus batches were produced by transfecting separate dishes of 293T cells). K562 cells bearing a GFP reporter were infected in quadruplicate (2 biological replicates receiving different lentiviral batches, each with 2 technical replicates) at 1000x coverage with low multiplicity of infection (MOI = 0.3) to ensure that each infected cell received only one dCasMINI-modulator fusion construct. Infected cells were then enriched by Puromycin selection. At 7 days post-transduction, cells were sorted using fluorescence-activated cell sorting (FACS) to isolate GFP_OFF and GFP_ON populations (Supplementary Figure 1). For each replicate, $\approx 10^6$ GFP_ON cells were harvested. Genomic DNA was prepared from GFP_ON, GFP_OFF, and unsorted populations, and barcoded modulator sequences were amplified using PCR primers to add Illumina i5 and i7 indexed adapter sequences. Pooled libraries were then sequenced on an Illumina NextSeq (Gladstone Genomics Core).



Supplementary Figure 1: Representative FACS histograms illustrating cell counts within GFP_OFF (P4) and GFP_ON (P5) gates in un-infected cells (a) and cells infected with the validation library (b). FITC-A is a readout for GFP levels.

Bioinformatic Analysis Raw sequencing reads were first aligned to a list of DNA barcodes (each putative modulator peptide is associated with a unique 12bp DNA barcode) in order to generate a count matrix. Read counts of technical replicates were then summed by barcode resulting in 2 biological replicates of GFP_ON and GFP_OFF libraries respectively. DESEQ2 [20] was used to determine which peptides were differentially enriched in the GFP_ON bins vs the GFP_OFF bins resulting in 357 positive hits (FDR<0.05 and log2FC>0) that were capable of activating the synthetic gene locus ((Supplementary Figure 2a, Supplementary Table 5, Supplementary Table 6). As expected, negative controls were depleted from the GFP_ON bin as these randomly encoded peptides were unable to activate the synthetic fluorescent reporter (Supplementary Figure 2b).

Principal Component Analysis (PCA) In order to visualize how the library of 4600 novel proposed sequences compared to the original training data, we performed principal component analysis (PCA) with Scikit-learn in Python (v3.9). Briefly, we combined peptide sequences from our trianing data



Supplementary Figure 2: Volcano plots illustrating statistical signficance (adjusted p-value) and activation strength (Log2FoldChange) of experimentally screened modulator peptides colored by sampling method (a) and highlighting negative controls (b).

Experimental Results: Sampling Algorithm				
Algorithm	Initialization	Total Sequences	Number of Hits	Hit Percentage
EMCS	All	2600	338	13%
MHMCS	All	2000	18	0.9%
EMCS	Known	1310	270	20.6%
EMCS	Random	1290	68	5.3%
MHMCS	Random	2000	18	0.9%
Negative Controls	n/a	300	1	0.33%

Supplementary Table 5: Final hit percentage of novel sequences sorted by choice of sampling algorithm. Initialization: Notes the sampling algorithm starting sequence as either randomly initialized, or known positive. Total Sequences: Number of sequences that were identified as high fitness by the machine learning model. Number of Hits: Number of sequences that validated experimentally.

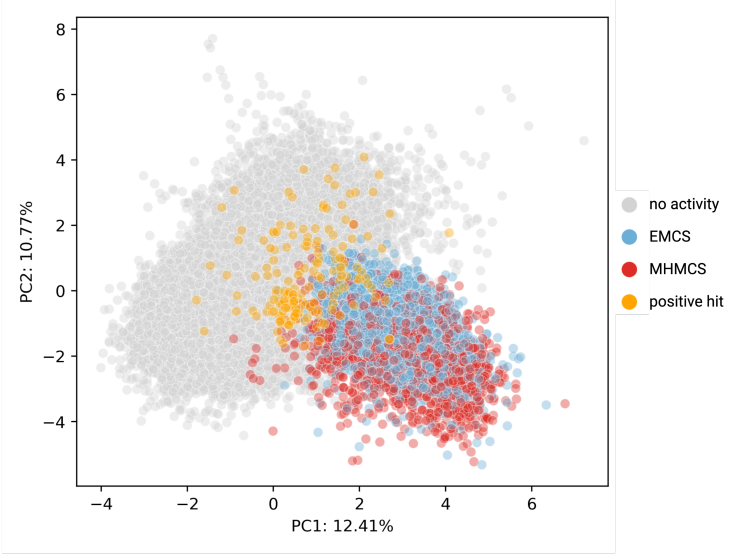
Experimental Results: Model Choice			
Fitness Model	Total Sequences	Number of Hits	Hit Percentage
Ensemble (XGB+CNN)	1000	135	13.5%
XGB	1800	145	8.06%
CNN	1800	76	4.2%
Negative Controls	300	1	0.33%

Supplementary Table 6: Final hit percentage of novel sequences sorted by choice of machine learning model. Total Sequences: Number of sequences that were identified as high fitness by the machine learning model. Number of Hits: Number of sequences that validated experimentally.

with the 4600 sequences in the validation library, represented each sequence with OneHot encoding and then flattened the encoding by taking the mean. Features were scaled using StandardScaler() prior to calculating and visualizing principal components.

1.6 In-Silico Experiments

To compare sequence proposals and convergence efficiency between EMCS and MHMCS, we performed an *in silico* sampling experiment where we explore the fitness landscape a minimum of 1000 times with each algorithm using identical and controlled initial conditions. For ablation studies, we also considered *in silico* sequence proposals generated by a parallel tempering setup (PTP: EMCS



Supplementary Figure 3: Principal Component Analysis (PCA) of original training set (grey and orange points) with novel sequences designed by EMCS (blue) and MHMCS (red) using OneHot encoding.

without crossovers), as well as those generated by a non-parallel tempered EMCS which we refer to as EMC-NPT (EMCS with crossovers, but with all chains run at the same temperature).

Convergence With default parameters, we achieved convergence for 1171 EMCS runs where we obtained at least one sequence per run that had a fitness ≥ 0.95. In addition, due to the inclusion of 4 chains, EMCS yielded an average of 2.322 sequences per run that had fitness ≥ 0.5, thereby giving us a total of N = 2720 sequences of fitness ≥ 0.5 for 1171 runs. For MHMCS, chains that started at temperatures greater than 2.5×10^{-2} had a minimum failure rate of 50%, and were dropped from the experiment. When excluding those sequences, we obtained a total of N = 2571 sequences from 2571 runs. 2361 of those sequences had fitness ≥ 0.95. The remaining 210 failed to reach convergence, but still had final fitness ≥ 0.5. In supplementary table 10 we show that the average number of primary iterations to convergence for EMC based methods is significantly lower than MHMCS or PTP.

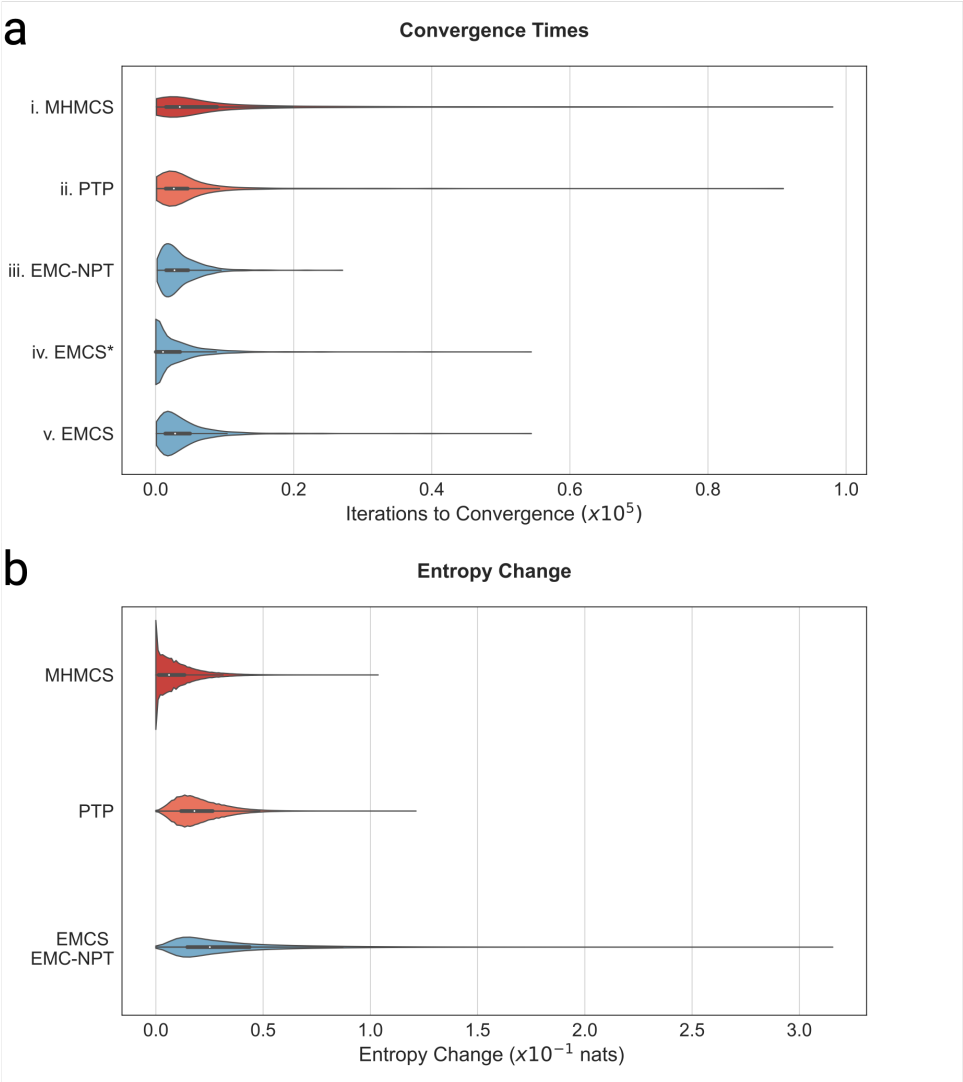
Entropy Calculations The change in entropy for a given iteration gives us a measure of the change in information observed for that iteration. Fig. S4 shows the entropy change distribution of two simulated MHMCS and EMCS runs under default parameters. However, the number of iterations was preset to 10^7 to average over a sufficiently large number of iterations (for reference, Supplementary Table 10 shows the convergence times of a general run under default parameters to be $< 10^4$).

The information entropy of a given sequence is defined as:

$$S = -\sum_i p_i log(p_i), \tag{3}$$

where i runs over the 20 possible amino acid residues, and p_i is the ratio of the count of a given amino acid residue divided by the length of the peptide. For example, a peptide sequence of 'AAASST' would have p_i equal to $\frac{3}{6}$, $\frac{2}{6}$, $\frac{1}{6}$ for residues 'A', 'S', and 'T' respectively, and zero for all other residues. For MHMCS, the entropy change for a given iteration is defined as $\Delta S = S_{proposed} - S_{initial}$. Where $S_{initial}$ is the entropy of the sequence at the beginning of the iteration, i.e. prior to the application of a mutation operator.

For EMCS, EMC-NPT and PTP, the entropy change is first independently calculated for each of the chains ($N_{chains} = 4$ for default parameters). Then, the value of the maximum entropy change is retained and recorded for a given iteration. It is important to note that the entropy changes in each of the chains occur due to the application of the mutation and/or the crossover operator, which in turn depends on the choice of parameters we use. This is also why EMCS and EMC-NPT have identical



Supplementary Figure 4: **a.** i) Iterations to convergence ($f \geq 0.95$) starting from random pre-defined sequences for 2361 sequences from 2571 MHMCS runs, ii) 617 sequences obtained from 617 PTP runs, iii) 1387 sequences obtained by 1387 EMC-NPT runs, iv) 1171 sequences obtained by 1171 EMCS runs, and v) number of iterations to convergence and/or positive hits ($f \geq 0.5$) for 2720 sequences obtained by 1171 EMCS runs under default parameters, yielding an average of 2.32 positive hits per EMCS run of 4 chains. In i), 210 sequences failed to reach convergence within 10^5 iterations, yet their final fitness was greater than 0.5. **b.** Entropy change distribution for 10^7 MHMCS, PTP, and EMCS primary iterations using default parameters. From an information perspective, EMCS amd EMC-NPT explore a larger region of the fitness space per iteration compared to MHMCS and PTP.

entropy change calculations, as the choice of mutation rate and crossover operators chosen were identical. In Fig. S4, we only show this difference for the default parameters given in Supplementary Table 3. Supplementary Table 9 provides additional data for Supplementary Figure 4.

Sequence diversity to training dataset To ensure that our sequences were sufficiently different from the training data when initialized from known positive sequences, we calculated the minimum

Entropy Change Distribution for 10^7 iterations using default parameters (Fig.5)					
Model	Mean	Standard Deviation	25^{th}	50^{th}	75^{th}
EMCS/EMC-NPT	0.334	0.229	0.146	0.252	0.437
PTP	0.202	0.114	0.118	0.18	0.26
MHMCS	0.092	0.101	0.013	0.061	0.133

Supplementary Table 7: Additional data for Supplementary Figure 5

edit (Levenshtein) distance of all high fitness sequences to the full set of sequences in the training dataset, thereby giving us a measure of sequence diversity. The average minimum edit distance of the entire set of sequences generated by a given algorithm can also be interpreted as a measure of novelty [21].

Supplementary Figure 5 summarizes this result. We note that, in general, EMC based methods yield sufficiently diverse sequences compared to MHMCS when starting from positive examples. While high temperature MHMCS runs yielded slightly more diverse sequences, they also failed to converge the majority of the time (275/500 runs).

Even with initialization from known positive hits, the sequences proposed by EMCS were highly dissimilar from anything in the training set, which suggests that EMCS is capable of escaping deep local optima to efficiently traverse the fitness landscape and identify diverse high fitness peptides.

Sequence Diversity measured by edit distance when starting from known positives						
Model	Number of Sequences	Mean	Standard Deviation	25^{th}	50^{th}	75^{th}
i. EMCS	1500	61.10	1.25	60	61	62
ii. EMC-NPT	1500	62.46	0.65	62	62	66
iii. PTP	1500	57.89	3.57	55	58	61
iv. MHMCS	500	49.21	5.96	38	42	46
v. MHMCS	500	57.4	4.21	55	58	61
vi. MHMCS*	225	61.82	1.43	61	62	63

Supplementary Table 8: Additional data for Figure S5. The mean of the edit distance from the training dataset is also reported as the novelty score of a generative sequence algorithm [21].

Sequence diversity amongst discovered sequences To evaluate the diversity of sequences discovered by a given algorithm, we calculated the diversity score as defined in [21], and described here as follows.

$$\text{Diversity}(D) = \frac{\sum \sum_{i,j}^{i \neq j} d(x_i, x_j)}{|D||D-1|} \tag{4}$$

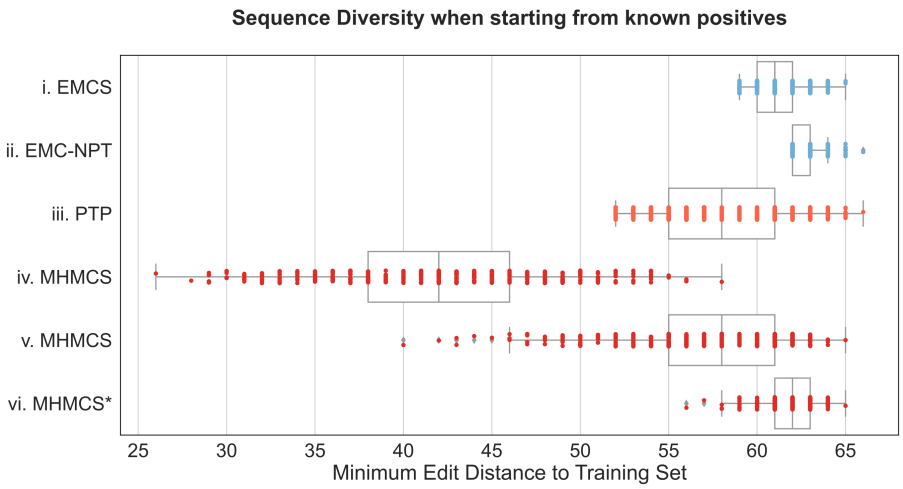
Where D is the set of sequences discovered by a given algorithm, and $|D|$ is the number of sequences in that set.

We postulate that the increased proposed sequence diversity and increased entropy per iteration seen with EMCS is due to the genetic crossover steps, where functionally beneficial protein domains can be exchanged between known sequences which are then further refined via point mutations. Escape from local minima is further encouraged by the incorporation of a temperature ladder, which allows for an increase in the search radius. In contrast, MHMCS is restricted to a single temperature and can only access domains in the fitness function that are accessible via point mutations alone. This hinders the ability of MHMCS to converge at a domain that corresponds to a diverse sequence when starting from a known positive sequence because it will require many sub-optimal moves to escape for the local optima of the initial sequence.

We believe our framework has a number of advantages over both prior ML-guided protein design approaches with traditional sampling techniques as well as the classic laboratory protein engineering approach. Firstly, assays that screen diverse, natural proteins for peptides of specific function typically have extremely low hit rates whereas novel sequences proposed by our approach had significantly

Diversity Score			
Algorithm	Initialization	Number of Sequences	Diversity Score
EMCS	Known	1500	68.548
EMC-NPT	Known	1500	69.898
PTP	Known	1500	70.032
MHMCS	Known	500	54.885
MHMCS*	Known	500	67.695
MHMCS**	Known	225	71.794
EMCS	Random	1171	74.522
EMC-NPT	Random	1387	74.399
PTP	Random	617	74.033
MHMCS	Random	2361	73.399

Supplementary Table 9: Diversity score: Mean pairwise edit distance of sequences discovered by a given algorithm. *MHMCS run at $T = 2.5 \times 10^{-3}$. **MHMCS run at $T = 2.5 \times 10^{-2}$.



Supplementary Figure 5: Minimum edit distance of converged sequences to the original training dataset initialized from known positive hits when using i-iii) EMCS, EMC-NPT, and PTP under default parameters (N = 1500), iv) MHMCS with $T = 2.5 \times 10^{-4}$ (N = 500), v) MHMCS with $T = 2.5 \times 10^{-3}$ (N = 500), and vi) $T = 2.5 \times 10^{-2}$ (N = 500). Although the average edit distance of the sequences obtained with vi) were slightly higher than i), the failure rate at vi) exceeded 50%. Refer to Table S8 for additional data.

higher hit rates in the validation experiment. Additionally, the small number of positive hits in the training data of protein engineering problems inherently limits the accuracy and generalizability of the fitness function; by leveraging information from LPLMs and incorporating multiple positive hits in the proposal of novel sequences through EMCS domain swapping, we believe our approach is capable of attenuating these disadvantages.

The approach described here should be of benefit to the wider scientific community, especially those involved in protein engineering challenges, and has the potential to accelerate the design and testing of novel proteins for a variety of purposes including therapeutic medicines.

Additional Data: Number of Iterations to Convergence (Fig. S4								
Model	Inclusion Criteria	Number of Runs	Number of Sequences	Mean	Standard Deviation	25^{th}	50^{th}	75^{th}
EMCS	$f > 0.95$	1171	1171	3950	4172	1500	2825	4985
EMC-NPT	$f > 0.95$	1387	1387	3457	2752	1529	2736	4709
PTP	$f > 0.95$	617	617	4402	7052	1466	2644	4641
MHMCS	$f > 0.95$	2571	2361	8837	14409	1509	3501	8892
EMCS	$f > 0.5$	1171	2720	2453	3822	35	1075	3546
EMC-NPT	$f > 0.5$	1387	1495	3510	2791	1564	2762	4775
PTP	$f > 0.5$	617	1062	4974	8541	1601	2839	5262
MHMCS	$f > 0.5$	2571	2571	15990	28159	1629	3960	12534

Supplementary Table 10: Additional data for Supplementary Figure 4

References

[1] Giovanni A. Carosso, Robin W. Yeo, T. Blair Gainous, M. Zaki Jawaid, Xiao Yang, Vincent Cutillas, Lei Stanley Qi, Timothy P. Daley, and Daniel O. Hart. Discovery and engineering of hypercompact epigenetic modulators for durable gene activation. *bioRxiv*, 2023. doi: 10.1101/2023.06.02.543492. URL `https://www.biorxiv.org/content/early/2023/06/15/2023.06.02.543492`.

[2] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, March 2023. doi: 10.1126/science.ade2574. URL `https://doi.org/10.1126/science.ade2574`.

[3] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15), April 2021. doi: 10.1073/pnas.2016239118. URL `https://doi.org/10.1073/pnas.2016239118`.

[4] Ben Coleman. Why is it okay to average embeddings? https://randorithms.com/2020/11/17/Adding-Embeddings.html. Accessed: 2023-09-22.

[5] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations*, 2017.

[6] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL `http://doi.acm.org/10.1145/2939672.2939785`.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. doi: 10.1038/nature14539. URL `https://doi.org/10.1038/nature14539`.

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[9] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6): 1087–1092, June 1953. doi: 10.1063/1.1699114. URL `https://doi.org/10.1063/1.1699114`.

[10] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970. doi: 10.1093/biomet/57.1.97. URL `https://doi.org/10.1093/biomet/57.1.97`.

[11] Sam Sinai and Eric D Kelsic. A primer on model-guided exploration of fitness landscapes for biological sequence design, 2020. URL `https://arxiv.org/abs/2010.10614`.

[12] Surojit Biswas, Grigory Khimulya, Ethan C. Alley, Kevin M. Esvelt, and George M. Church. Low-n protein engineering with data-efficient deep learning. *Nature Methods*, 18(4):389–396, April 2021. doi: 10.1038/s41592-021-01100-y. URL `https://doi.org/10.1038/s41592-021-01100-y`.

[13] Ivan Anishchenko, Samuel J Pellock, Tamuka M Chidyausiku, Theresa A Ramelot, Sergey Ovchinnikov, Jingzhou Hao, Khushboo Bafna, Christoffer Norn, Alex Kang, Asim K Bera, et al. De novo protein design by deep network hallucination. *Nature*, 600(7889):547–552, 2021.

[14] Egbert Castro, Abhinav Godavarthi, Julian Rubinfien, Kevin Givechian, Dhananjay Bhaskar, and Smita Krishnaswamy. Transformer-based protein generation with regularized latent space optimization. *Nature Machine Intelligence*, 4(10):840–851, 2022.

[15] Will Grathwohl, Kevin Swersky, Milad Hashemi, David Duvenaud, and Chris Maddison. Oops i took a gradient: Scalable sampling for discrete distributions. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3831–3841. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/grathwohl21a.html.

[16] Zhizhou Ren, Jiahan Li, Fan Ding, Yuan Zhou, Jianzhu Ma, and Jian Peng. Proximal exploration for model-guided protein sequence design. In *International Conference on Machine Learning*, pages 18520–18536. PMLR, 2022.

[17] Tianfan Fu and Jimeng Sun. Sipf: Sampling method for inverse protein folding. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 378–388, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539284. URL https://doi.org/10.1145/3534678.3539284.

[18] Faming Liang and Wing Hung Wong. Evolutionary monte carlo: applications to $\mathcal{C}_p$ model sampling and change point problem. *Statistica sinica*, pages 317–342, 2000.

[19] Valentin Zulkower and Susan Rosser. DNA chisel, a versatile sequence optimizer. *Bioinformatics*, 36 (16):4508–4509, July 2020. doi: 10.1093/bioinformatics/btaa558. URL https://doi.org/10.1093/bioinformatics/btaa558.

[20] Michael I Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, 15(12), December 2014. doi: 10.1186/s13059-014-0550-8. URL https://doi.org/10.1186/s13059-014-0550-8.

[21] Moksh Jain, Emmanuel Bengio, Alex-Hernandez Garcia, Jarrid Rector-Brooks, Bonaventure F. P. Dossou, Chanakya Ekbote, Jie Fu, Tianyu Zhang, Micheal Kilgour, Dinghuai Zhang, Lena Simine, Payel Das, and Yoshua Bengio. Biological sequence design with gflownets, 2022. URL https://arxiv.org/abs/2203.04115.