Sparse Epistatic Regularization of Deep Neural Networks for Inferring Fitness Functions

Amirali Aghazadeh[1], Hunter Nisonoff[2], Orhan Ocal[1], Yijie Huang[1], O. Ozan Koyluoglu[1], Jennifer Listgarten[1,2], and Kannan Ramchandran[1,*]

[1]Department of Electrical Engineering and Computer Science,
[2]Center for Computational Biology, University of California, Berkeley
*Correspondence to: Kannan Ramchandran: kannanr@eecs.berkeley.edu

Abstract

Despite recent advances in high-throughput combinatorial mutagenesis assays, the number of labeled sequences available to predict molecular functions has remained small for the vastness of the sequence space combined with the ruggedness of many fitness functions. Expressive models in machine learning (ML), such as deep neural networks (DNNs), can model the nonlinearities in rugged fitness functions, which manifest as high-order epistatic interactions among the mutational sites. However, in the absence of an inductive bias, DNNs overfit to the small number of labeled sequences available for training. Herein, we exploit the recent biological evidence that epistatic interactions in many fitness functions are sparse; this knowledge can be used as an inductive bias to regularize DNNs. We have developed a method for sparse epistatic regularization of DNNs, called the *epistatic net* (EN), which constrains the number of non-zero coefficients in the spectral representation of DNNs. For larger sequences, where finding the spectral transform becomes computationally intractable, we have developed a scalable extension of EN, which subsamples the combinatorial sequence space uniformly inducing a sparse-graph-code structure, and regularizes DNNs using the resulting greedy optimization method. Results on several biological landscapes, from bacterial to protein fitness functions, showed that EN consistently improves the prediction accuracy of DNNs and enables them to outperform competing models which assume other forms of inductive biases. EN estimates all the higher-order epistatic interactions of DNNs trained on massive sequence spaces—a computational problem that takes years to solve without leveraging the epistatic sparsity in the fitness functions.

Keywords: fitness function | epistasis | deep neural network | regularization | sparse-graph codes

Significance Statement

Predicting the properties of small molecules (such as proteins) from their sequence is an important problem in computational biology. The main challenge is in developing a model that can capture the non-linearities in the function mapping the sequence to the property of interest (e.g., fluorescence) using the limited number of available labeled sequences from biological assays. In this paper, we identify a biologically-plausible sparsity prior and develop a method to infuse this prior into the structure of deep neural networks (DNNs) by regularizing their spectral representation. We demonstrate that our method significantly improves the prediction accuracy of DNNs and enables an interpretable explanation of DNNs—a task that is computationally intractable without leveraging the hidden structure in biological functions.

1 Introduction

Recent advances in next-generation sequencing have enabled the design of high-throughput combinatorial mutagenesis assays that measure molecular functionality for tens of thousands to millions of sequences simultaneously. These assays have been applied to many different sequences in biology, including protein-coding sequences [1–3], RNAs [4–6], bacterial genes [7–10], and the SpCas9 target sites [11–13]. The labeled sequences collected from these assays have been used to train supervised machine learning (ML) models to predict functions (*e.g.*, fluorescence, binding, repair outcome, *etc.*) from the sequence—a key step in the rational design of molecules using ML-assisted directed evolution [14]. However, due to the limitations in techniques for library preparation, these assays can only uncover a small subset of all the possible combinatorial sequences. This raises an important question in learning fitness functions: how can we enable supervised ML models to infer fitness functions using only a small number of labeled sequences?

Inferring fitness functions is a challenging task since mutational sites interact *nonlinearly* to form the function, a phenomenon known as *epistasis* in genetics [15, 16]. As a result, linear regression models which assume site-independent interactions achieve poor accuracy in predicting nonlinear functions. Augmenting linear models with pairwise, second-order epistatic interactions improves their prediction accuracy [3]; however, there is now increasing evidence that a large fraction of the variance in the fitness functions can be explained only by higher-order epistatic interactions, which contribute to the 'ruggedness' of fitness landscapes [17, 18]. Modeling rugged fitness landscapes is a hard task since the total number of possible higher-order interactions grows exponentially with the number of mutational sites. As a result, the number of parameters to be estimated (*i.e.*, the problem dimension) also grows with the same exponential rate, which creates statistical challenges in inferring the fitness function since the number of labeled sequences does not scale with the problem dimension. In response, nonlinear ML models constrain the problem dimension by introducing various forms of inductive biases to capture hidden structures in the fitness functions. Random forests, for example, impose a tree structure over sites which favor 'tree-like' hierarchical epistatic interactions. While these inductive biases are effective in some fitness functions [19], they are too restrictive to capture the underlying higher-order epistatic interactions in other fitness functions [3]. Overparameterized models in deep learning (DL), such as deep neural networks (DNNs), are expressive enough to model high-order espistatic interactions given a large number of labeled training sequences; however, when the number of labeled sequences is small, they often overfit to the training data and compromise prediction accuracy. It has been observed that regularizing DNNs to induce domain-specific biases improves their prediction accuracy for various tasks in computer vision and natural language processing [20]. This opens up the question of whether there exists an inductive bias for DNNs trained on biological fitness landscapes that can be imposed using a computationally tractable regularization scheme.

Recent studies in biological landscapes [3, 13, 21] have reported that a large fraction of the variance in many fitness functions can be explained by only a few number of (high-order) interactions between the mutational sites. The epistatic interactions in these functions are a mixture of a small number of (high-order) interactions with large coefficients, and a larger number of interactions with small coefficients; in other words, their epistatic interactions are highly sparse. Promoting sparsity among epistatic interactions is a powerful inductive bias for predictive modeling because it reduces the problem dimension without biasing the model towards a subset of (low-order) interactions. Despite its benefits, promoting sparsity among epistatic interactions has not been studied in DNNs as an inductive bias. The roadblock is in finding a method to promote epistatic sparsity in DNNs. Unfortunately, directly penalizing all or some of the parameters (weights) of DNNs with sparsity-promoting priors is not likely to result in sparse epistatic regularization since the epistatic coefficients are a complex nonlinear function of the weights in DNNs.

Herein, we develop a method for sparse epistatic regularization of DNNs. We call our method *epistatic net* (EN) because it resembles a fishing net which catches the epistatic interactions among all the combinatorially possible interactions in DNNs, without any restriction to a subset of (low-order) interactions. In order to find the epistatic interaction as a function of the weights in DNN, we find its spectral representation (also called the Walsh-Hadamard (WH) transform for binary sequences) by evaluating the DNN on the entire combinatorial space of mutations, and then finding the WH spectral transform of the resulting landscape using the Fast WH Transform (FWHT). The resulting function of the weights in DNN is penalized to promote epistatic sparsity. For larger sequences this approach for epistatic regularization becomes computationally intractable due to the need to enumerate all possible mutations in DNN. Therefore, we leverage the fast sparsity-enabled algorithms in signal processing and coding theory in order to develop a greedy optimization method to regularize DNNs at scale. Our scalable regularization method, called EN-S, regularizes DNNs by sampling only a small subset of the combinatorial sequence space by choosing sequences based on sparse-graph codes. The uniform sampling scheme allows us to find the WH transform of the combinatorial DNN landscape efficiently using a fast peeling algorithm [22]. Results on several biological landscapes, from bacterial to protein fitness functions, shows that EN(-S) enables DNNs to achieve consistently higher prediction accuracy compared to competing models and estimate all the higher-order predictive interactions on massive combinatorial sequence space—a computational problem that takes years to solve without leveraging the epistatic sparsity structure in the fitness landscapes.

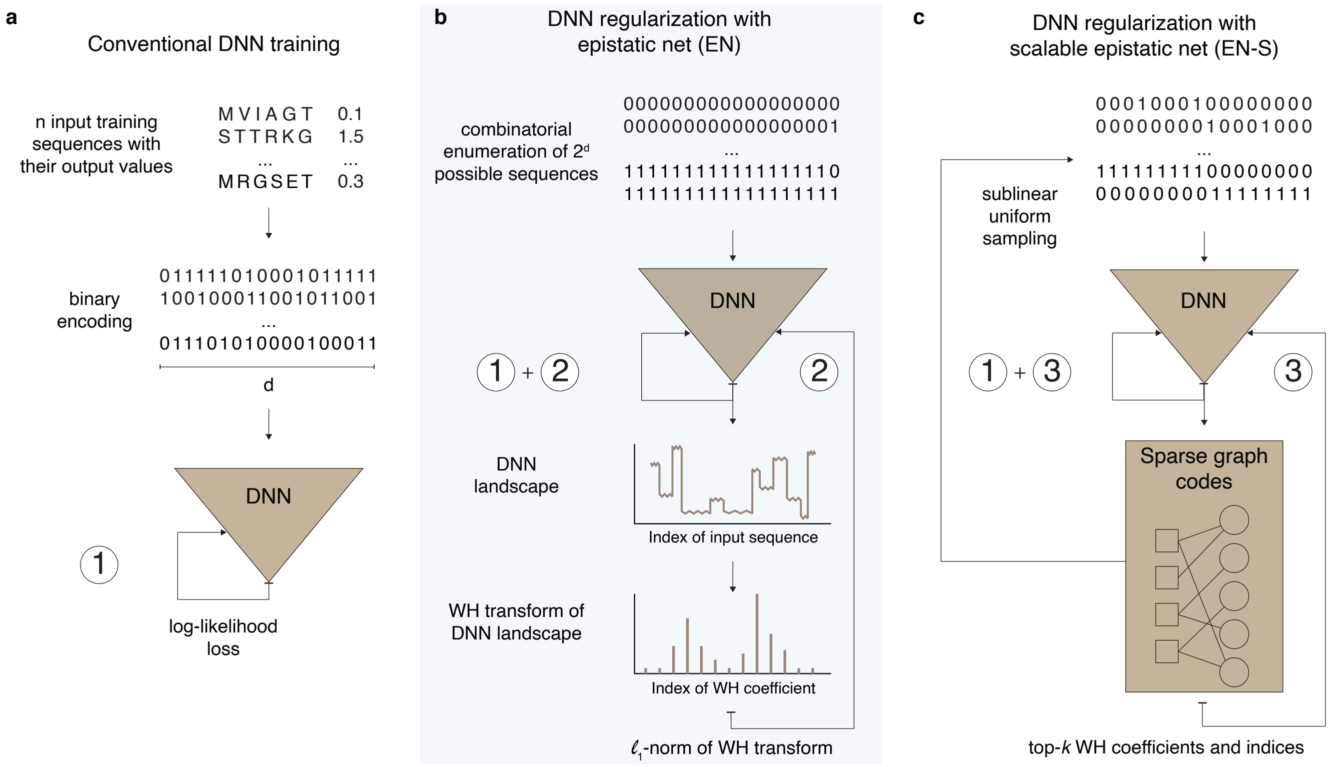

Figure 1: **Schematic illustration of our sparse epistatic regularization method, called epistatic net (EN).** **a**, Conventional deep neural network (DNN) training is depicted, where the log-likelihood loss, computed over n labeled training sequences encoded into binary sequences of length d, is minimized using the stochastic gradient descent (SGD) algorithm. **b**, In every iteration, EN queries DNN for all the 2^d possible binary input sequences, finds the Walsh Hadamard (WH) spectral transform of the resulting landscape using the Fast WH Transform (FWHT), and then adds the ℓ_1-norm of the WH transform to the log-likelihood loss from panel **a**. **c**, In the scalable version of EN, EN-S regularizes DNN using only a few number of uniformly subsampled sequences from the combinatorial input space following the patterns in sparse-graph codes. EN-S iterates between these two subproblems until convergence: 1) finding the sparse WH transform of DNN (using sublinear samples and in sublinear time) with sparse-graph codes, and 2) minimizing the sum of the log-likelihood loss and the WH loss using SGD.

2 Results

Regularization using the epistatic net (EN). EN is a novel regularization scheme (Figure 1b) which evaluates the DNN on all the possible combinatorial mutations of the input sequence; we call the resulting high-dimensional vector the DNN landscape. It then takes the WH transform of the DNN landscape and adds the sparsity-promoting ℓ_1-norm (*i.e.*, the sum of the absolute values) of the WH coefficients to the log-likelihood loss. The resulting WH loss is a differentiable function (except at zero) of the weights in DNN and is weighted by a scalar which strikes a balance between the fidelity of DNN to the labeled sequences and sparsity among epistatic interactions (see Methods for more detail). We used the stochastic gradient descent (SGD) algorithm to minimize the aggregate loss and update the weights of DNN in every iteration.

For larger sequences (of size $d > 25$), EN regularization becomes intractable in time and space complexity. This is because EN needs to query the DNN $p = 2^d$ times to form the DNN landscape (exponential time complexity in d) and then find the WH transform of the queried DNN landscape (exponential time and space complexity in d). To overcome this, EN-S leverages the sparsity in the WH spectral domain to regularize DNN using only a small number of uniformly subsampled sequences from the combinatorial input space (Figure 1c). EN-S decouples the DNN training, following the alternating direction method of multipliers (ADMM) framework [23], into two subproblems: 1) finding the k-sparse WH spectral transform of DNN in a sample and time efficient manner, and 2) minimizing the sum of the log-likelihood loss and the WH loss. The WH loss penalizes the distance between DNN and a function constructed using the top-k WH coefficients recovered in the first subproblem. In order to solve the first subproblem, we designed a careful subsampling of the input sequence space [22] that induces a linear mixing of the WH coefficients such that a greedy belief propagation algorithm (peeling-decoding) over a sparse-graph code recovers the noisy DNN landscape in sublinear sample (*i.e.*, $\mathcal{O}(k \log^2 p)$) and time (*i.e.*, $\mathcal{O}(k \log^3 p)$) complexity in p (with high probability) [13, 22, 24, 25]. We solved the second subproblem using the SGD algorithm. EN-S alternates between these two steps until convergence (see Methods for more detail).

Inferring four canonical functions in bacterial fitness. We collected four canonical bacterial fitness functions, whose combinatorial landscapes have been measured experimentally in previously published works (see Table S2 in Supplementary Materials). Figure 2a shows the sparsity level in epistatic interactions of these bacterial fitness functions. The coefficients for epistatic interactions were found by taking the WH transform of the measured combinatorial landscape (see Methods section for various ways to preprocess the landscapes). The fraction of variance explained is plotted as a function of the top WH coefficients. Sparsity levels can be assessed by the proximity of the resulting curve towards the top-left corner of the plot. For comparison, we plot synthetic fitness functions that have all possible epistatic interactions up to a certain order of interaction in Figure 2a. While the sparsity levels varied across fitness functions, the top-5 WH coefficients consistently explained more than 80% of the variance across all the landscapes.

Figure 2b shows the prediction performance of DNN with EN regularization on the bacterial landscapes compared to various competing models. All the models were trained on the same randomly sampled subset of the sequences from the measured combinatorial landscapes and tested on a subset of unseen sequences (see Supplementary Materials for more details). The prediction accuracy is reported in terms of the coefficient of determination, R^2 (*i.e.*, the fraction of the variance in the test set explained from the sequence). DNN with EN regularization consistently outperformed the baseline models in all the landscapes. In particular, DNN with EN regularization performs significantly better than the EN-regularized variant, even though DNN was optimized (in terms of architecture) for best validation

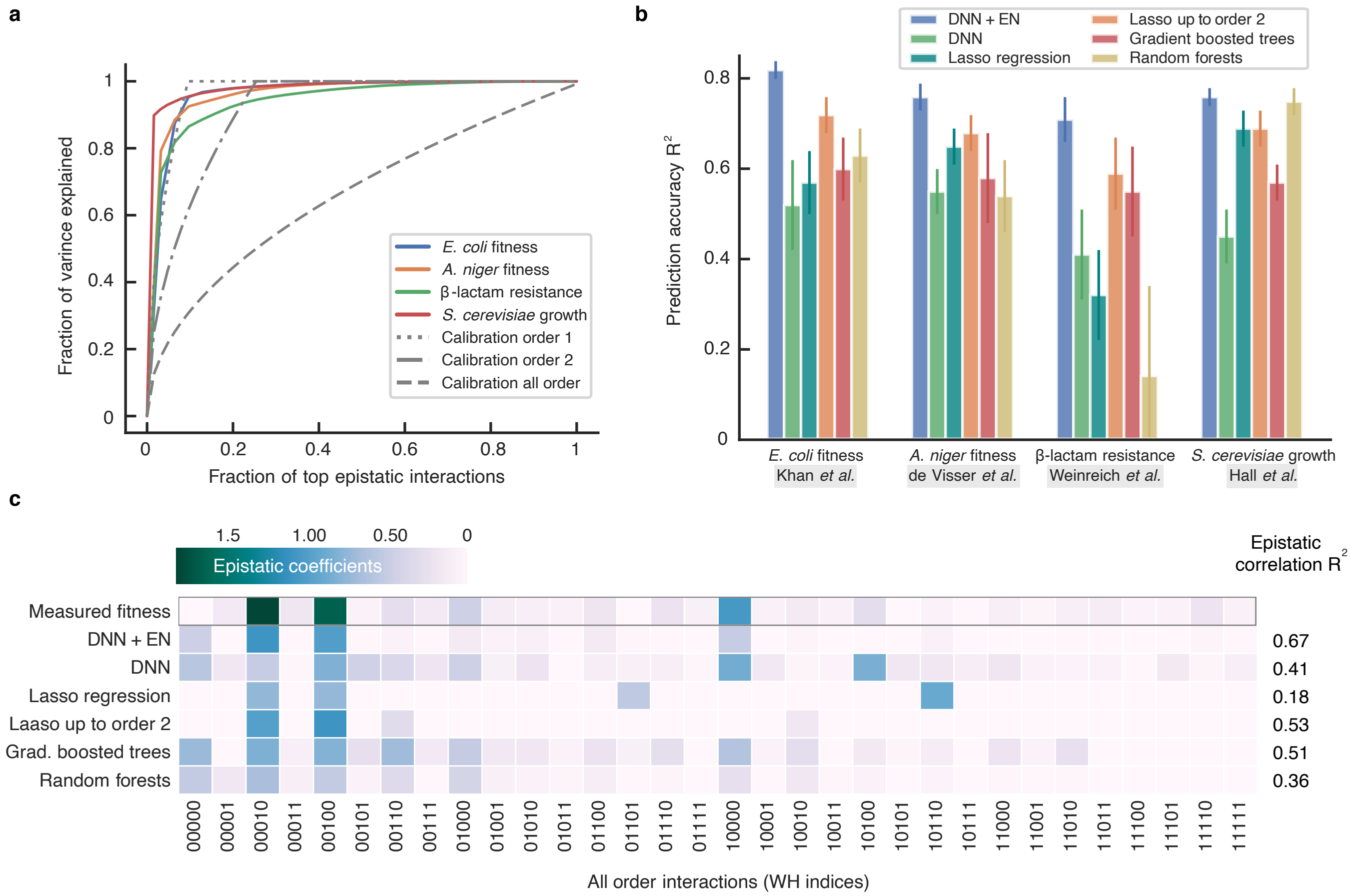

Figure 2: **Predicting bacterial fitness and inferring epistatic interactions in four canonical landscapes.** **a**, Fraction of variance explained by the top WH coefficients showing the sparsity level in the bacterial fitness functions. **b**, Prediction accuracy of deep neural network (DNN) with epistatic net (EN) regularization against competing models in ML. The error bars show the standard error of the mean (SEM) across 5 independent repeats of the experiments with random split of the data into training, validation, and test sets. **c**, Visualization of the epistatic interactions of DNN with and without EN regularization and the baseline models after training on *E. coli* fitness landscape of Khan *et al.* [9]. R^2 values show the correlation of the recovered epistatic interaction with the interactions in the measured combinatorial *E. coli* fitness landscape.

performance in isolation (*i.e.*, without espistatic regularization) and had been subjected to other forms of common sparsity-promoting regularization techniques applied directly to the weights of the DNN (see Methods for more details).

Figure 2c shows the WH transform of the DNN landscape, with and without EN regularization, as well as the landscape of the rest of the competing models, all trained on a training set sampled from *E. coli* fitness landscape of Khan *et al.* [9] (see Figure S3 and S4 for a detailed analysis of the landscapes in spectral domain). In order to find these landscapes, we queried each model for all the combinatorial mutations and took the WH transform using FWHT. In this plot, the epistatic coefficient indexed by

10100, as an example, shows an order 2 interaction between the mutational sites 1 and 3. The rest of the indices can be interpreted similarly. The epistatic interactions in the measured *E. coli* fitness function showed three interactions with higher coefficients and several epistatic interactions with lower coefficients along the WH spectrum. The interactions recovered by DNN with EN regularization closely matches the measured epistatic interactions of the measured *E. coli* fitness function ($\mathrm{R}^2 = 0.67$), a considerable improvement over DNN without EN regularization ($\mathrm{R}^2 = 0.41$). The WH coefficients of gradient boosted trees ($\mathrm{R}^2 = 0.51$) and random forests ($\mathrm{R}^2 = 0.36$) were spread out in the WH spectrum. Lasso regression found two of the three interactions with higher coefficients, however, recovered a spurious third-order interaction which resulted in a low epistatic correlation coefficient ($\mathrm{R}^2 = 0.18$). When restricted to up to order 2 interactions, performance of Lasso improved; it recovered the two interactions with higher coefficients, however, missed the third coefficient and the rest of the small epistatic interactions ($\mathrm{R}^2 = 0.53$).

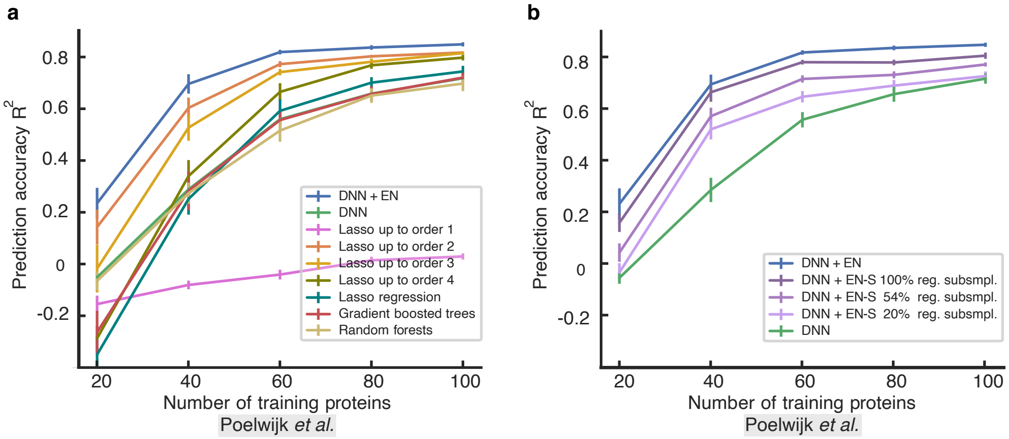

Figure 3: **Inferring the sparse epistatic protein landscape of Poelwijk *et al.* [3].** **a**, Deep neural network (DNN) with epistatic net (EN) regularization outperforms the baselines in terms of prediction accuracy. The performance gap is more pronounced in low-sample regimes. In particular, to achieve the same prediction accuracy, DNN with EN regularization needs up to 3 times less number of samples compared to DNN without EN regularization. **b**, The prediction performance of DNN with EN-S regularization is plotted when EN-S subsamples DNN at progressively smaller fractions of the combinatorial sequence space of proteins, that is, 100% (no subsampling), 54%, and 20%. DNN with EN-S regularization outperforms DNN without the regularization despite restricting EN-S to only sample 20% of the protein sequence space (following sparse-graph codes). Error bars in both plots show the standard error of the mean (SEM) in 20 independent repeats of the experiments with random splits of the data into training, validation, and test sets.

***Entacmaea quadricolor* fluorescent protein.** A comprehensive experimental study has reported all the combinatorial mutants that link two phenotypically distinct variants of the *Entacmaea quadricolor* fluorescent protein [3]. The variants were different in $d = 13$ mutational sites. The study showed the existence of several high-order epistatic interactions between the sites, but also revealed extraordinary sparsity in the interactions. We used this protein landscape to assess EN in regularizing DNN for predicting protein function. We split the $2^{13} = 8192$ labeled proteins randomly into three sets: training, validation, and test. The size of the test set was fixed to 3000 and the validation set size was set equal to the training set size. We varied the training set size from a minimum of $n = 20$ proteins to a maximum of $n = 100$ proteins and measured the accuracy of the models in predicting fitness on the unseen test set in terms of R^2. DNN with EN regularization consistently outperformed the baseline models and the performance gap was more pronounced with smaller training set sizes (Figure 3a). This gap between the DNN with and without EN regularization shows the power of the epistatic sparsity prior in inferring this sparse landscape; in order to achieve the same level of prediction accuracy, DNN with EN regularization required up to 3 times fewer number of training samples, compared to DNN without EN regularization. We note that applying various forms of ℓ_1 and ℓ_2-norm regularization on the weights of different layers of the DNN did not change the performance gap between DNN with and without EN regularization (see Figure S5 in Supplementary Materials). Lasso regression showed a peculiar behavior as we increased the order of allowed interactions. Prediction performance of Lasso improved as the maximum allowed order of interaction increased from 1 to 2, but then degraded with higher-order interactions. We speculate that this is due to the substantial increase in the number of parameters, resulting in Lasso picking up spurious correlation.

The moderate dimension of the fluorescent landscape of *Entacmaea quadricolor* protein enabled us to use the data set to compare the performance of DNN under EN regularization with its scalable version, EN-S. The prediction performance of DNN with EN-S regularization showed a slight drop in accuracy due to the approximations made by the ADMM decoupling (Figure 3b, see Methods). EN-S stayed fairly consistent when we decreased the number of proteins sampled from DNN following the patterns in sparse-graph codes. Using as low as 1678 samples (out of the total of 8192 combinatorial proteins, *i.e.*, 20% subsampling) enabled successful regularization of DNN, resulting in a significant performance gap compared to DNN without EN regularization.

Green fluorescent protein from *Aequorea victoria* (avGFP). The local fitness landscape of the green fluorescent protein from *Aequorea victoria* (avGFP) has been investigated in a comprehensive study [2]. The authors estimated the fluorescence levels of genotypes obtained by random mutagenesis of the avGFP protein sequence at 236 amino acid mutational sites. The final data set included $56,086$ unique nucleotide sequences coding for $51,715$ different protein sequences. Considering the absence or presence of a mutation at a site, created a data set with input sequence size of $d = 236$. Regularization in the resulting $p = 2^{236}$-dimensional space was impossible using EN, illustrating the need for EN-S. We analyzed the EN-S regularization by inspecting the WH spectral representation of DNN once trained on the landscape. Figure S6a shows the first-order WH coefficients of DNN, recovered using EN-S after sampling DNN at $5,074,944$ (out of $2^{236} \approx 10^{71}$) proteins following patterns suggested by sparse-graph codes. We repeated the same procedure with an independent set of subsampled sequences (new sequences from sparse-graph codes with random offset) and visualized the recovered first-order WH coefficients in a scatter plot as a function of the recovered coefficients using the first set of proteins in Figure 4a. When sampled at two different relatively tiny subsets of this massive $p = 2^{236}$-dimensional space, EN-S recovered similar first-order coefficients (with $\mathrm{R}^2 = 0.99$), without assuming any prior

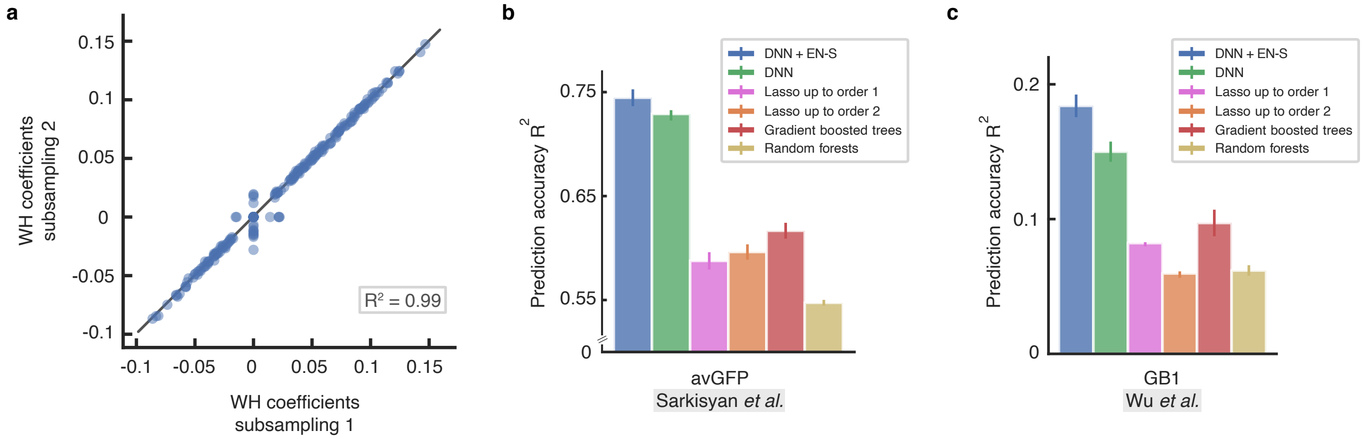

Figure 4: **Robustness of recovered epistatic interactions in the scalable epistatic net (EN-S) and inferring fitness function in two large canonical protein landscapes. a**, The scatter plot of the first-order Walsh-Hadamard (WH) coefficients of DNN trained on Sarkisyan *et al.* landscape [2] recovered by EN-S using two independent sets of $5,074,944$ subsampled proteins (out of 2^{236}) following patterns in sparse-graph codes. The recovered coefficients are highly consistent ($\mathrm{R}^2 = 0.99$). The higher variance of the scatter plot around the center shows the small number (20 out of 236) of coefficients that are deferentially recovered. **b**, DNN with EN-S regularization outperforms the baselines in terms of prediction accuracy in Sarkisyan *et al.* landscape [2]. **c**, DNN with EN-S regularization outperforms the baselines in terms of prediction accuracy in Wu *et al.* landscape [1]. Error bars show the standard error of the mean (SEM) in 3 independent repeats of the experiments with random splits of the data into training, validation, and test sets.

knowledge on the WH coefficients of avGFP being low-order (also see Figure S6). The higher variance of the scatter plot around the center shows the small number of coefficients (30 out of 236) that are deferentially recovered under the two subsamplings. EN-S associated 3.2% and 2.9% of the variation of DNN to higher-order interactions, respectively for the first and second subsampling. We compared the second-order interactions recovered under these subsamplings (Figure S7). Despite the small variation associated with higher-order epistasis, 10% of the recovered second-order interactions were exactly equal, and the rest of the interactions were locally correlated ($\mathrm{R}^2 = 0.60$ block-correlation). Figure 4b shows that the prediction accuracy of DNN with EN-S regularization is higher than the baseline algorithms. The gap between DNN with and without EN-S regularization is smaller compared to the previously described protein landscapes. We speculate that this is due to the nature of the avGFP landscape, where most of the variance in the landscape can be explained by first-order interactions and the rest can be explained by higher-order interactions that are spread throughout the WH spectrum.

Immunoglobulin-binding domain of protein G (GB1). A recent study investigated the fitness landscape of all the $20^4 = 160,000$ variants at four amino acid sites (V39, D40, G41 and V54) in an epistatic region of protein G domain B1, an immunoglobulin-binding protein expressed in Streptococcal bacteria [1]. One-hot encoding of the amino acids resulted in binary sequences of length $d = 80$. As EN does not scale to regularize DNNs trained on this landscape, we relied on EN-S. Figure 4c shows the prediction performance of DNN with EN-S regularization compared to the baseline models that scaled to such dimension. All the models were trained on a random subset of $n = 500$ proteins. EN-S subsampled DNN at $215,040$ proteins in order to perform the sparse epistatic regularization, which was about 10^{18} times smaller than the entire sequence space. Despite such enormous level of undersampling, the DNN regularized with EN-S consistently outperformed the baselines. The performance gap between the DNN with and without EN-S regularization is smaller compared to the same gap in *Entacmaea quadricolor* fluorescent protein landscape. We speculate that this is due to the different nature of the effect of epistatic sparsity in these two landscape: in the *Entacmaea quadricolor* fluorescent protein landscape EN-S promotes sparsity among the $d = 13$ mutational sites, however, in the GB1 landscape the regularization power of EN-S is mostly spent on promoting sparsity among the 20 amino acids in each of the $d = 4$ mutational sites.

3 Conclusion and Discussion

We showed that several of the functional landscapes in biology have common structures (*i.e.*, inductive bias) in their epistatic interactions that manifest as sparsity in the spectral Walsh-Hadamard (WH) domain. Sparse epistatic regularization of deep neural networks (DNNs) is an effective method to improve their prediction accuracy, especially when the number of available training samples is small compared to the vastness of sequence space. To this end, our epistatic net (EN) regularization method combined the advantages offered by the sparsity of biological landscapes with sublinear algorithms in signal processing and coding theory for epistatic regularization of DNNs in the combinatorial space of interactions.

The superior prediction performance of DNNs with EN regularization comes with the additional computational cost of finding the WH transform of the DNN landscape, which increases the computational complexity of the training algorithm by only a linear factor in the product of the length of the sequence and the epistatic sparsity level. While training can be done offline (*e.g.*, on a server) there are avenues for making the algorithm even more efficient such as using the prior knowledge on the maximum order of interaction to constraint the regularization space. Moreover, while this work laid out the algorithmic principles of sparse epistatic regularization in supervised models, unsupervised models, such as Potts model [26], Ising model [27], and Variational Autoencoders (VAEs) [28] can potentially benefit from such regularization scheme as well: it would be tempting to hypothesize that these energy landscapes also have structures that appear as high-order sparse coefficients in WH basis.

Overall, our sparse epistatic regularization method expands the machine learning toolkit for inferring and understanding fitness functions in biology. It helps us to visualize, analyze, and regularize the powerful, however less interpretable black-box models in deep learning in terms of their higher-order interactions in the sequence space. We believe that our work will initiate new research directions towards developing hybrid methodologies that draws power from statistical learning, signal processing, coding theory, and the new generation of physics-inspired models in machine learning.

4 Methods

Notation. Suppose we are given n (experimental) samples $(\mathbf{x}_i, y_i)_{i=1}^{n}$, that is, (sequence, value) pairs from a biological landscape, where $\mathbf{x}_i \in \{-1,+1\}^d$ denotes the binary encoding of d mutational sites in a variant and $y_i \in \mathbb{R}$ is its associated fitness value. We are interested in learning a function $f(\mathbf{x})$ that maps all subsets of mutations to fitness values. In other words, we seek to learn a *set function* $f(\mathbf{x}) : \mathbb{F}^d \to \mathbb{R}$, where $\mathbb{F}^d$ denotes the space of all the binary vectors of length d. A key theorem [29] in mathematics states that any set function (also known as pseudo-Boolean function) $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_d)$ can be represented uniquely by a multi-linear polynomial over the hyper cube $(x_1, x_2, \ldots, x_d) \in \{-1,+1\}^d$:

$$f(x_1, x_2, \ldots, x_d) = \sum_{\mathcal{S} \subseteq [d]} \alpha_{\mathcal{S}} \prod_{i \in \mathcal{S}} x_i, \tag{1}$$

where $\mathcal{S}$ is a subset of $\{1,2,3,\ldots,d\} = [d]$ and $\alpha_{\mathcal{S}} \in \mathbb{R}$ is the WH transform coefficient (or equivalently the epistatic coefficient) associated with the monomial (interaction) $\prod_{i \in \mathcal{S}} x_i$. For example, the pseudo-Boolean function

$$f(x_1, x_2, x_3, x_4, x_5) = 12x_1x_4 - 3x_3 + 6x_1x_2x_5, \tag{2}$$

defined over $d = 5$ mutational sites, has three monomials with orders 2, 1, and 3 and WH coefficients 12, -3, and 6, respectively. The WH transform of this function is sparse with $k = 3$ non-zero coefficients out of a total of $2^5 = 32$ coefficients. Each monomial can be easily explained, for example, the first monomial in the WH transform, that is $12x_1x_4$, indicates that mutation sites 1 and 4 are interacting and the interaction enriches fitness because the sign of the coefficient is positive. On the hand, the second monomial $-3x_3$ shows that a mutation at site 3 depletes fitness. The last monomial $6x_1x_2x_5$ shows a third-order interaction between mutational sites 1, 2, and 5 which also enrich fitness.

If the fitness function is measured (known) for all the combinatorial $p = 2^d$ inputs $\mathbf{x}_i$, then we can use the Fast WH Transform (FWHT) [30] to find the WH coefficients in $\mathcal{O}(p \log p)$ time complexity. The problem is so-called fully determined in such scenario. However, as discussed in the introduction, in inferring fitness functions, we typically face problems where the number of observed samples (sequences) n is much smaller than the total number of possible sequences, that is, $n \ll p = 2^d$; in other words, we are in an under-determined regime. In full generality, we assume that the data is generated according to a noisy nonlinear model

$$y_i = f_{\boldsymbol{\theta}}(\mathbf{x}_i) + \varepsilon_e, \tag{3}$$

where $\boldsymbol{\theta}$ are the parameters of the model, ε_e is a random variable drawn from a Gaussian distribution with zero mean and variance σ_e^2. Under this setting the maximum likelihood estimate is

$$\boldsymbol{\theta}_{MLE} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} (y_i - f_{\boldsymbol{\theta}}(\mathbf{x}_i))^2. \tag{4}$$

We denote a deep neural network (DNN) by $g_{\boldsymbol{\theta}}(\mathbf{x})$, where $\boldsymbol{\theta}$ is a vector of all the weights in DNN. The DNN, $g_{\boldsymbol{\theta}}(\mathbf{x})$, takes in a binary input vector $\mathbf{x}_i$ and predicts the output $\hat{y}_i$. Let $\mathbf{X} \in \mathbb{R}^{p \times d}$ denote a matrix which comprises all the $p = 2^d$ enumeration of the binary sequence $\mathbf{x}_i$ of length d in its rows. We slightly abuse the notation and let $\mathbf{g}_\theta(\mathbf{X}) \in \mathbb{R}^p$ denote the real-valued vector of DNN outputs over all these binary sequences. We call this high-dimensional vector the DNN landscape. In order to find the WH transform of the DNN we can multiply the DNN landscape, $\mathbf{g}_\theta(\mathbf{X})$, by the WH matrix, $\mathbf{H} \in \mathbb{R}^{p \times p}$. The WH matrix $\mathbf{H}$ can be defined using the recursive equation

$$\mathbf{H}^{2^d} = \mathbf{H}^2 \otimes \mathbf{H}^{2^{d-1}}, \tag{5}$$

where $\mathbf{H}^2$ is the 2×2 'mother' matrix defined as $\mathbf{H}^2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ and $\otimes$ denotes the Kronecker product. The WH matrix is a symmetric unitary matrix; in other words, $(1/2^d)\mathbf{HH} = \mathbf{I}$. Each of the 2^d columns of $\mathbf{H}$ corresponds to a monomial $(\prod_{i \in \mathcal{S}} x_i)$ in the pseudo-Boolean representation of set functions and equivalently corresponds to one of the terms in WH transform. In biology literature, this coefficients is known as an epistatic interaction when $|\mathcal{S}| \geq 2$. The WH transform of the DNN can be calculated as $\mathbf{H}\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{X}) \in \mathbb{R}^p$.

Epistatic net (EN). EN regularizes the epistatic interactions in $\mathbf{g}_\theta(\mathbf{X})$ by adding a new WH loss term to the original log-likelihood loss,

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (y_i - g_{\boldsymbol{\theta}}(\mathbf{x}_i))^2 + \alpha \|\mathbf{H}\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{X})\|_0, \tag{6}$$

where $\mathbf{H} \in \mathbb{R}^{p \times p}$ is the WH matrix, the ℓ_0-norm $\|.\|_0$ counts the number of non-zero values in the WH transform of the DNN (*i.e.*, $\mathbf{H}\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{X})$), and α is a scalar which strikes balance between the log-likelihood loss and the regularization term. The scalar α is set using cross-validation. The ℓ_0-norm is a non-convex and non-differentiable term and is not suitable for optimization using the SGD algorithm since the gradient is not well-defined for this term; therefore, following the common practice in convex optimization, we relaxed the ℓ_0-norm and approximated it by a convex and differentiable (except at zero) sparsity promoting ℓ_1-norm in EN. We will discuss in the next section that in the scalable version of EN, it is more efficient to approximately solve the ℓ_0-norm minimization problem using the greedy peeling-decoding algorithm from coding theory, which does not rely on gradient descent optimization.

EN approximately solves the following relaxed optimization problem using the SGD algorithm,

EN

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (y_i - g_{\boldsymbol{\theta}}(\mathbf{x}_i))^2 + \alpha \|\mathbf{H}\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{X})\|_1. \tag{7}$$

Note that despite our convex relaxation, this optimization problem is still non-convex since both the log-likelihood loss and the DNN landscape are non-convex (still differentiable) functions. In general, convergence to the global minimum can not be guaranteed due to non-convexity of DNN, however, in practice we observe that SGD converges smoothly to a useful stationary locally optimal point. To avoid convergence to locally optimal points with poor generalization performance, the DNN can be trained multiple times with several random initialization, however, as we have elaborated in the experimental section, for most of the experiments in this paper random Xavier initialization resulted in excellent generalization using a single initialization (no need for multiple initializations).

Scalable epistatic net (EN-S). For larger sequences (*i.e.*, $d > 25$), the optimization algorithm in EN does not scale well with d. There are two factors that prevents EN from scaling to larger sequences: time and space complexity. We elaborate on these two factors: 1) In order to find the DNN landscape, we need to query the DNN $p = 2^d$ times. Regardless of how fast DNN inference is, the time complexity of this task grows exponentially with d. For example, it would take years to query the DNN with simplest structure on all the binary sequences of length $d = 236$ in the avGFP protein landscape. Furthermore, finding the WH transform of the DNN landscape, even using FWHT with $\mathcal{O}(p \log p)$ computational cost, will not be possible since the computational cost grows exponentially with d. 2) The WH matrix $\mathbf{H}$ is a $p \times p$ matrix and the DNN landscape $g_{\boldsymbol{\theta}}(\mathbf{X})$ is a p-dimensional vector. Regardless of the time required to find those matrices, they need exponential memory to store, which becomes infeasible for even moderate values of d. We need a method that scales sublinear in p (*i.e.*, $\mathcal{O}(\text{polylog } p)$) both in time and space complexity.

Herein, we develop EN-S to approximately solve our optimization problem efficiently. We first perform a change of variables and define the WH transform of the DNN landscape as $\mathbf{u} = \mathbf{H}\mathbf{g}_\theta(\mathbf{X})$ and set it as an explicit constraint in the optimization problem. Following this change of variable, we reformulate the optimization problem in equation (7) as,

$$\min_{\boldsymbol{\theta},\mathbf{u}} \sum_{i=1}^{n} (y_i - g_{\boldsymbol{\theta}}(\mathbf{x}_i))^2 + \alpha\|\mathbf{u}\|_1 \;\; \text{subject to} \;\; \mathbf{u} = \mathbf{H}\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{X}). \tag{8}$$

This change of variable enables us to use an augmented Lagrangian method to decouple the optimization problem in equation (7) into two subproblems: 1) updating the weights of DNN using SGD, and, 2) finding the WH transform of DNN using a fast greedy algorithm based on sparse-graph codes. The alternating direction method of the multipliers (ADMM) is a variant of the augmented Lagrangian methods that uses partial updates for the dual variables and provides a principled framework to decouple the optimization problem above. Following the scaled-dual form of ADMM [23], we decoupled the optimization problem above into two separate minimization problems and a dual update. At iteration t, we first fix $\mathbf{u}_t \in \mathbb{R}^p$ and solve a $\boldsymbol{\theta}$-minimization problem, then fix $\boldsymbol{\theta}_t \in \mathbb{R}^p$ and solve a $\mathbf{u}$-optimization problem, and finally update the dual variable $\boldsymbol{\lambda} \in \mathbb{R}^p$ as follows,

- $\boldsymbol{\theta}$-minimization $\quad \boldsymbol{\theta}^{t+1} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n}(y_i - g_{\boldsymbol{\theta}}(\mathbf{x}_i))^2 + \frac{\rho}{2}\|\mathbf{H}\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{u}^t + \boldsymbol{\lambda}^t\|_2^2$
- $\mathbf{u}$-minimization $\quad \mathbf{u}^{t+1} = \arg\min_{\mathbf{u}} \alpha\|\mathbf{u}\|_1 + \frac{\rho}{2}\|\mathbf{H}\mathbf{g}_{\boldsymbol{\theta}^{t+1}}(\mathbf{X}) - \mathbf{u} + \boldsymbol{\lambda}^t\|_2^2$
- dual update $\quad \boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t + \mathbf{H}\mathbf{g}_{\boldsymbol{\theta}^{t+1}}(\mathbf{X}) - \mathbf{u}^{t+1},$

where $\rho \in \mathbb{R}$ is a hyperparamter set using cross-validation. Note that the time and space scaling issues remain here and will be addressed momentary. Assuming an infinite time and space budget, the $\boldsymbol{\theta}$-minimization problem can be tackled using SGD and the $\mathbf{u}$-minimization problem can be solved by projecting $\mathbf{w}^{t+1} := \mathbf{H}\mathbf{g}_{\boldsymbol{\theta}^{t+1}}(\mathbf{X}) + \boldsymbol{\lambda}^t$ onto the ℓ_1-norm ball of radius ρ/α. This projection can be solved using the soft-thresholding operator in Lasso [31]:

$$\mathbf{u}_i^{t+1} = \begin{cases} \mathbf{w}_i^{t+1} - \rho/2\alpha & \text{if} \quad \mathbf{w}_i^{t+1} > \rho/2\alpha \\ 0 & \text{if} \quad \rho/2\alpha \leq \mathbf{w}_i^{t+1} \leq \rho/2\alpha \\ \mathbf{w}_i^{t+1} + \rho/2\alpha & \text{if} \quad \mathbf{w}_i^{t+1} < \rho/2\alpha. \end{cases} \tag{9}$$

Unfortunately, all the three steps above still have exponential time and space scaling with d. In what follows we will show how to exploit the sparsity of the WH transform of the DNN landscape $\mathbf{u} = \mathbf{H}\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{X})$ to reformulate new minimization steps such that we need to subsample only a logarithmic factor $\mathcal{O}(\text{polylog } p)$ of rows in $\mathbf{H}$ and approximately solve these steps in sublinear time and space complexity in p (*i.e.*, at most polynomial in d). We call this regularization scheme EN-S.

The first step to arrive at the EN-S regularization scheme is to reformulate the optimizations above such that the WH matrix $\mathbf{H}$ appears as a multiplicative term behind the dual variable $\boldsymbol{\lambda}$ and $\mathbf{u}$. This enables us to convert the $\mathbf{u}$-minimization problem from a ℓ_1-norm ball projection to a sparse WH recovery problem with $\mathbf{H}$ as the basis, for which we have fast solvers from signal processing and coding theory. Note that $\|\mathbf{H}\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{u}^t + \boldsymbol{\lambda}^t\|_2^2 = \|\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{H}\mathbf{u}^t + \mathbf{H}\boldsymbol{\lambda}^t\|_2^2$ and $\|\mathbf{H}\mathbf{g}_{\boldsymbol{\theta}^{t+1}}(\mathbf{X}) - \mathbf{u} + \boldsymbol{\lambda}^t\|_2^2 = \|[\mathbf{g}_{\boldsymbol{\theta}^{t+1}}(\mathbf{X}) + \mathbf{H}\boldsymbol{\lambda}^t] - \mathbf{H}\mathbf{u}\|_2^2$ because $\mathbf{H}$ is a unitary matrix. Therefore, we can write the optimization steps above as,

- $\boldsymbol{\theta}$-minimization $\quad \boldsymbol{\theta}^{t+1} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n}(y_i - g_{\boldsymbol{\theta}}(\mathbf{x}_i))^2 + \frac{\varrho}{2}\|\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{H}\mathbf{u}^t + \mathbf{H}\boldsymbol{\lambda}^t\|_2^2$
- $\mathbf{u}$-minimization $\quad \mathbf{u}^{t+1} = \arg\min_{\mathbf{u}} \alpha\|\mathbf{u}\|_1 + \frac{\varrho}{2}\|[\mathbf{g}_{\boldsymbol{\theta}^{t+1}}(\mathbf{X}) + \mathbf{H}\boldsymbol{\lambda}^t] - \mathbf{H}\mathbf{u}\|_2^2$
- dual update $\quad \mathbf{H}\boldsymbol{\lambda}^{t+1} = \mathbf{H}\boldsymbol{\lambda}^t + \mathbf{g}_{\boldsymbol{\theta}^{t+1}}(\mathbf{X}) - \mathbf{H}\mathbf{u}^{t+1}.$

Now, the $\mathbf{u}$-minimization problem is to find the WH transform of $\mathbf{g}_{\boldsymbol{\theta}^{t+1}}(\mathbf{X}) + \mathbf{H}\boldsymbol{\lambda}^t$ with an ℓ_1-norm sparsity prior. In order to solve this $\mathbf{u}$-minimization problem, we resort to the fast sparsity-enabled tools in signal processing and coding theory. This class of greedy algorithms solves the original ℓ_0-norm minimization problem and finds the k-WH sparse landscape (for specific value of k) in a time and space efficient manner ($\mathcal{O}(k \text{ polylog } p)$, *i.e.*, $\mathcal{O}(k \text{ poly } d)$) using sparse-graph codes (see Supplementary Materials for an overview of these methods). To this end, we leverage subsampling of input sequences based on patterns in sparse-graph codes [22]. We denote the rows corresponding to these subsampled sequences as $\mathbf{X}_T$, where $|T| \sim \mathcal{O}(k\log^2 p)$. The subsampling induces a linear mixing of WH coefficients such that a belief propagation algorithm (peeling-decoding) over a sparse-graph code recovers a p-dimensional noisy landscape with k non-zero WH coefficients in sublinear sample (*i.e.*, $\mathcal{O}(k\log^2 p)$) and time complexity (*i.e.*, $\mathcal{O}(k\log^3 p)$) with high probability [13, 22, 24, 25] (see Supplementary Materials for a full discussion). This fully addresses both the time and space scalability issues in solving the $\mathbf{u}$-minimization problem.

In order to resolve the time and space scalability issues in the $\boldsymbol{\theta}$-minimization problem and the dual update we introduce a novel approximation. We follow the subsampling patterns dictated by the sparse-graph codes in solving the $\mathbf{u}$-minimization problem, and restrict both the $\boldsymbol{\theta}$-minimization problem and the dual update to those subsamples as well to arrive at,

- $\boldsymbol{\theta}$-minimization $\quad \boldsymbol{\theta}^{t+1} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n}(y_i - g_{\boldsymbol{\theta}}(\mathbf{x}_i))^2 + \frac{\varrho}{2}\|\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{X}_T) - \mathbf{H}_T\mathbf{u}^t + \boldsymbol{\gamma}^t\|_2^2$
- $\mathbf{u}$-minimization $\quad \mathbf{u}^{t+1} = \arg\min_{\mathbf{u}} \alpha\|\mathbf{u}\|_0 + \frac{\varrho}{2}\|[\mathbf{g}_{\boldsymbol{\theta}^{t+1}}(\mathbf{X}_T) + \boldsymbol{\gamma}^t] - \mathbf{H}_T\mathbf{u}\|_2^2$
- dual update $\quad \boldsymbol{\gamma}^{t+1} = \boldsymbol{\gamma}^t + \mathbf{g}_{\boldsymbol{\theta}^{t+1}}(\mathbf{X}_T) - \mathbf{H}_T\mathbf{u}^{t+1},$

where $\boldsymbol{\gamma}^t := \mathbf{H}_T\boldsymbol{\lambda}^{t+1} \in \mathbb{R}^{|T|}$ and $\mathbf{H}_T$ comprises the rows of $\mathbf{H}$ that are in T. Note that the change of dual variable $\boldsymbol{\gamma}^t = \mathbf{H}_T\boldsymbol{\lambda}^{t+1}$ is only possible because in all the three steps the dual variable $\boldsymbol{\lambda}^{t+1}$ appears in the WH basis. Note that while the columns of the subsampeld WH matrix $\mathbf{H}_T$ still live in a p-dimensional space, this matrix is never instantiated in memory because it only appears as $\mathbf{H}_T\mathbf{u}$, where $\mathbf{u}$ is a k-sparse vector. Therefore, $\mathbf{H}_T\mathbf{u}$ is computed on the fly by only finding the columns of the (row-subsampeld) WH matrix $\mathbf{H}_T$ that corresponds to the non-zero values in $\mathbf{u}$.

The final EN-S method iterates over these three steps to train the DNN until convergence. We indicate the algorithm to solve each step in brackets,

EN-S

- $\boldsymbol{\theta}$-minimization $\quad \boldsymbol{\theta}^{t+1} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (y_i - g_{\boldsymbol{\theta}}(\mathbf{x}_i))^2 + \frac{\rho}{2}\|\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{X}_T) - \mathbf{H}_T\mathbf{u}^t + \boldsymbol{\gamma}^t\|_2^2$ **[SGD]**
- $\mathbf{u}$-minimization $\quad \mathbf{u}^{t+1} = \arg\min_{\mathbf{u}} \alpha\|\mathbf{u}\|_0 + \frac{\rho}{2}\|[\mathbf{g}_{\boldsymbol{\theta}^{t+1}}(\mathbf{X}_T) + \boldsymbol{\gamma}^t] - \mathbf{H}_T\mathbf{u}\|_2^2$ **[Peeling]**
- dual update $\quad \boldsymbol{\gamma}^{t+1} = \boldsymbol{\gamma}^t + \mathbf{g}_{\boldsymbol{\theta}^{t+1}}(\mathbf{X}_T) - \mathbf{H}_T\mathbf{u}^{t+1}.$ **[Directly computed]**

All the three steps above in the EN-S method scale sublinearly with p (*i.e.*, at most polynomial with d) both in terms of time and space complexity.

Experimental setup. The architecture of DNN was selected in isolation (*i.e.*, without any WH regularization). In our architecture search we considered a four-layer fully-connected DNN with batch normalization and leaky ReLU as the activation function. The dimension of the layers were set to $d \times fd$, $fd \times fd$, $fd \times d$, and the dimension of the final layer was $d \times 1$, where f is an expansion factor. We searched for a value of f that resulted in best generalization accuracy on an independent data set—a prediction task on DNA repair landscapes [13] which we did not use for evaluation in this paper. DNN prediction performance was stable around $f = 10$ with highest validation accuracy on the independent data set. We selected $f = 10$ in all our experiments, except for the experiments done on the avGFP landscape [2], where due to the sheer dimensionality of the problem (*i.e.*, $d = 236$), we set $f = 1$ (on limited independent tests with $f = 10$ on the same landscape, we observed no considerable difference in prediction accuracy). The weights of the DNN were always initialized with the Xavier uniform initialization [32]. We used the exact same initialization (random seed) for the baseline DNN with and without EN(-S) regularization to ensure that we solely capture the effect of regularization and not the variations due to initialization of DNN. We used the Adam optimizer in all the steps of the methods requiring SGD and learning rate of 0.001, which resulted in best validation accuracy. We always set $\alpha = 0.1$ in EN. For the DNN with EN(-S) regularization, a learning rate of 0.01 resulted in the best validation accuracy. In EN-S, the hyperparameters α and ρ have to be jointly set since they are dependent. We set $\alpha = 1$ and $\rho = 0.01$ in EN-S although other value pairs could have resulted in the same accuracy. The validation accuracy of DNN was monitored and used for early stopping to avoid over-fitting based on the performance on a hold-out validation set (with a maximum of 1000 epochs). We used the exact same validation set to perform hyperparamter tuning of the baseline algorithms, including the Lasso family, random forest, and gradient boosted trees.

For the family of Lasso regression, we did an extra step to improve the prediction performance. We selected the top most recovered coefficients and did ordinary least squares (OLS) on the reduced problem. This step improves the robustness and the prediction accuracy of Lasso [33]. Therefore, in addition to the standard λ regularization parameter, which strikes a balance between sparsity and the fidelity term (*i.e.*, the mean squared error), we also did hyperparameter tuning for the number of top coefficients in the OLS (note that the regular Lasso is included in our hyperparameter search and appears when all the non-zero coefficients is selected to perform OLS). We did a grid search over the hyperparameter λ and the number of top coefficients in Lasso. For λ we considered 50 values spanning the range $[10^{-7}, 1]$. Overall, this comprised of an exhaustive hyperparameter search to make sure the best performance of Lasso is being captured.

For training gradient boosted trees and random forests baselines, we used packages from sklearn in python. We did hyperparameter tuning for max depth and the number of estimators, using the default values for all other parameters. For max depth, we considered parameters ranging from 1 to the maximum number of mutations in the fitness function ($i.e.$, d), for the number of estimators we considered values in $\{10, 50, 100, 200, 300, 400, 500, 1000, 2000, 3000\}$, and chose the pair that resulted in best validation accuracy. As a general trend, we observed that larger numbers of estimators result in higher validation accuracies before they saturate.

Herein, we report the hyperparameters that resulted in highest validation accuracy, that is, the ones we selected in our experiments. For the avGFP landscape, we set the number of estimators to 300 and max depth to 11 for gradient boosted trees and set the number of estimators to 100 and max depth to 55 for random forests. We set $\lambda = 1 \times 10^{-4}$ for Lasso regression when considering up to first-order interactions and $\lambda = 1 \times 8^{-4}$ when considering up to second-order interactions. For the GB1 landscape, we set the number of estimator to 100 and max depth to 2 for both gradient boosted trees and random forests. We set $\lambda = 7 \times 10^{-3}$ for Lasso regression when considering up to first-order interactions and $\lambda = 2.5 \times 10^{-2}$ when considering up to second-order interactions. For the protein landscape in Figure!3, we set the number of estimators to 3000 and the max depth varied between the values in the sets $\{1, 2, 3, 4\}$ and $\{1, 2, \ldots, 15\}$ across the random repeats of the experiments with different train, test, and validation set, respectively for gradient boosted trees and random forest; the value with the best validation performance was selected for each repeat. For the bacterial landscapes in Figure 2, we set the number of estimators to 300 and the max depth varied between the values in the set $\{1, 2, 3\}$ across the random repeats of the experiments with different train, test, and validation set; the value with the best validation performance was selected for each repeat.

Prepossessing the fitness landscapes. For the landscapes tested in this paper, we followed the Box-Cox power transform method as described in ref. [18] to remove possible global nonlinearities from the landscape. Although the effect of removing such nonlinearities was small in our analysis, global nonlinearities in general can produce high-order epistatic interactions that are not truly needed. Removing these nonlinearlities can reduce noise and increase epsitatic sparsity. Nevertheless, one can completely ignore this preprocessing step and rely on DNN with EN regularisation to capture the global nonlinearites and infer the fitness landscape for prediction purposes.

5 Code Availability

A software for the EN and EN-S regularization algorithms has been developed in Python and is publicly available in our github repository at https://github.com/amirmohan/epistatic-net. All the data sets used in the paper are publicly available in the references cited in this manuscript.

6 Acknowledgment

The authors thank David Brookes and Clara Wong-Fannjiang for insightful discussions.

References

[1] Nicholas C Wu, Lei Dai, C Anders Olson, James O Lloyd-Smith, and Ren Sun. Adaptation in protein fitness landscapes is facilitated by indirect paths. *eLife*, 5:e16965, 2016.

[2] Karen Sarkisyan, Dmitry Bolotin, Margarita Meer, Dinara Usmanova, Alexander Mishin, George Sharonov, Dmitry Ivankov, Nina Bozhanova, Mikhail Baranov, Onuralp Soylemez, et al. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397–401, 2016.

[3] Frank Poelwijk, Michael Socolich, and Rama Ranganathan. Learning the pattern of epistasis linking genotype and phenotype in a protein. *Nature Communications*, 10(1):1–11, 2019.

[4] George Kopsidas, Rachael K Carman, Emma L Stutt, Anna Raicevic, Anthony S Roberts, Mary-Anne V Siomos, Nada Dobric, Luisa Pontes-Braz, and Greg Coia. Rna mutagenesis yields highly diverse mrna libraries for in vitroprotein evolution. *BMC Biotechnology*, 7(1):18, 2007.

[5] Rafael Sanjuán. Mutational fitness effects in rna and single-stranded dna viruses: common patterns revealed by site-directed mutagenesis studies. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1548):1975–1982, 2010.

[6] Matthew A Takata, Steven J Soll, Ann Emery, Daniel Blanco-Melo, Ronald Swanstrom, and Paul D Bieniasz. Global synonymous mutagenesis identifies cis-acting rna elements that regulate HIV-1 splicing and replication. *PLoS Pathogens*, 14(1):e1006824, 2018.

[7] Daniel Weinreich, Nigel Delaney, Mark DePristo, and Daniel Hartl. Darwinian evolution can follow only very few mutational paths to fitter proteins. *Science*, 312(5770):111–114, 2006.

[8] David Hall, Matthew Agan, and Sara Pope. Fitness epistasis among 6 biosynthetic loci in the budding yeast Saccharomyces cerevisiae. *Journal of Heredity*, 101(suppl_1):S75–S84, 2010.

[9] Aisha Khan, Duy Dinh, Dominique Schneider, Richard Lenski, and Tim Cooper. Negative epistasis between beneficial mutations in an evolving bacterial population. *Science*, 332(6034):1193–1196, 2011.

[10] J Arjan GM De Visser and Joachim Krug. Empirical fitness landscapes and the predictability of evolution. *Nature Reviews Genetics*, 15(7):480–490, 2014.

[11] Jennifer Listgarten, Michael Weinstein, Benjamin P Kleinstiver, Alexander A Sousa, J Keith Joung, Jake Crawford, Kevin Gao, Luong Hoang, Melih Elibol, John G Doench, et al. Prediction of off-target activities for the end-to-end design of CRISPR guide RNAs. *Nature Biomedical Engineering*, 2(1):38–47, 2018.

[12] Ryan T Leenay, Amirali Aghazadeh, Joseph Hiatt, David Tse, Theodore L Roth, Ryan Apathy, Eric Shifrut, Judd F Hultquist, Nevan Krogan, Zhenqin Wu, et al. Large dataset enables prediction of repair after CRISPR-Cas9 editing in primary T cells. *Nature Biotechnology*, 37(9):1034–1037, 2019.

[13] Amirali Aghazadeh, Orhan Ocal, and Kannan Ramchandran. CRISPRLand: Interpretable large-scale inference of DNA repair landscape based on a spectral approach. *Bioinformatics*, 36(Supplement_1):i560–i568, 2020.

[14] Zachary Wu, SB Jennifer Kan, Russell D Lewis, Bruce J Wittmann, and Frances H Arnold. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proceedings of the National Academy of Sciences*, 116(18):8852–8858, 2019.

[15] Jason B Wolf, Edmund D Brodie, Michael John Wade, Michael J Wade, et al. *Epistasis and the evolutionary process*. Oxford University Press, USA, 2000.

[16] Heather J Cordell. Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans. *Human Molecular Genetics*, 11(20):2463–2468, 2002.

[17] Zachary R Sailer and Michael J Harms. High-order epistasis shapes evolutionary trajectories. *PLoS Computational Biology*, 13(5):e1005541, 2017.

[18] Zachary Sailer and Michael Harms. Detecting high-order epistasis in nonlinear genotype-phenotype maps. *Genetics*, 205(3):1079–1088, 2017.

[19] Rui Jiang, Wanwan Tang, Xuebing Wu, and Wenhui Fu. A random forest approach to the detection of epistatic interactions in case-control studies. *BMC Bioinformatics*, 10(1):1–12, 2009.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[21] Aditya Ballal, Caroline Laurendon, Melissa Salmon, Maria Vardakou, Jitender Cheema, Marianne Defernez, Paul E O'Maille, and Alexandre V Morozov. Sparse epistatic patterns in the evolution of Terpene synthases. *Molecular Biology and Evolution*, 2020.

[22] Xiao Li, Joseph Bradley, Sameer Pawar, and Kannan Ramchandran. The SPRIGHT algorithm for robust sparse Hadamard transforms. In *2014 IEEE International Symposium on Information Theory*, pages 1857–1861. IEEE, 2014.

[23] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[24] Xiao Li, Joseph Bradley, Sameer Pawar, and Kannan Ramchandran. SPRIGHT: A fast and robust framework for sparse Walsh-Hadamard transform. *arXiv preprint arXiv:1508.06336*, 2015.

[25] Xiao Li and Kannan Ramchandran. An active learning framework using sparse-graph codes for sparse polynomials and graph sketching. In *Advances in Neural Information Processing Systems*, pages 2170–2178, 2015.

[26] Fa-Yueh Wu. The Potts model. *Reviews of Modern Physics*, 54(1):235, 1982.

[27] Barry M McCoy and Tai Tsun Wu. *The two-dimensional Ising model*. Courier Corporation, 2014.

[28] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[29] Endre Boros and Peter Hammer. Pseudo-Boolean optimization. *Discrete Applied Mathematics*, 123(1-3):155–225, 2002.

[30] William T Cochran, James W Cooley, David L Favin, Howard D Helms, Reginald A Kaenel, William W Lang, George C Maling, David E Nelson, Charles M Rader, and Peter D Welch. What is the fast Fourier transform? *Proceedings of the IEEE*, 55(10):1664–1674, 1967.

[31] Mário Figueiredo, Robert Nowak, and Stephen Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, 2007.

[32] Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. In *Advances in Neural Information Processing Systems*, pages 571–581, 2018.

[33] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[34] David Donoho. Compressed Sensing. *IEEE Transaction on Information Theory*, 52(4):1289–1306, 2006.

[35] Joel Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

[36] Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[37] Tom Richardson and Rüdiger Urbanke. *Modern Coding Theory*. Cambridge University Press, 2008. Cambridge Books Online.

[38] Peter Elias. Error-free coding. *Transactions of the IRE Professional Group on Information Theory*, 4(4):29–37, Sep. 1954.

[39] Robin Scheibler, Saeid Haghighatshoar, and Martin Vetterli. A fast Hadamard transform for Signals with Sublinear Sparsity in the Transform Domain. *IEEE Transaction on Information Theory*, 61(4):2115–2132, 2015.

[40] Orhan Ocal, Swanand Kadhe, and Kannan Ramchandran. Low-degree Pseudo-Boolean Function Recovery Using Codes. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 1207–1211. IEEE, 2019.

[41] Thomas Richardson and Rüdiger Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2):599–618, 2001.

[42] Alex Tseng, Avanti Shrikumar, and Anshul Kundaje. Fourier-transform-based attribution priors improve the interpretability and stability of deep learning models for genomics. *bioRxiv*, 2020.

[43] Sameer Pawar and Kannan Ramchandran. FFAST: An algorithm for computing an exactly k-sparse DFT in $\mathcal{O}(k \log k)$ time. *IEEE Transactions on Information Theory*, 64(1):429–450, 2017.

[44] Mahdi Cheraghchi and Piotr Indyk. Nearly optimal deterministic algorithm for sparse Walsh-Hadamard transform. *ACM Transactions on Algorithms*, 13(3):1–36, 2017.

[45] Surojit Biswas, Grigory Khimulya, Ethan Alley, Kevin Esvelt, and George Church. Low-N protein engineering with data-efficient deep learning. *bioRxiv*, 2020.

[46] J Arjan GM de Visser, Su-Chan Park, and Joachim Krug. Exploring the effect of sex on empirical fitness landscapes. *The American Naturalist*, 174(S1):S15–S30, 2009.

7 Supplementary Materials

7.1 Sparse recovery using sparse-graph codes

The problem we are interested in this section is recovering the WH transform (WHT) coefficients (equivalently, the pseudo-Boolean function) when there is sparsity in the WHT domain. Methods proposed in compressed sensing literature can be used to recover a sparse signal (*i.e.*, landscape) in a sample efficient way [34]. However, the algorithms proposed in the literature like Orthogonal Matching Pursuit (OMP) [35] or Lasso [36] requires operations that scale at least linearly with the ambient dimension p. On the other hand, our method requires sublinear computational complexity whenever the degrees of freedom k scales sub-linearly with the ambient dimension p [24]. The key properties of our algorithm are presented in the following theorem.

Theorem 7.1 ([24]). *Let $\alpha \in (0,1)$ be a fixed number. Suppose $p = 2^d$ and assume $k = p^\alpha$. Let $\mathbf{y} \in \mathbb{R}^p$ be a vector and $\mathbf{Y} \in \mathbb{R}^p$ be its WHT. Assume that $\mathbf{Y}$ is k-sparse and its support is selected uniformly at random among all possible $\binom{d}{k}$ subsets of $[d]$ of size k. Then, there is an algorithm with the following properties:*

1. *Sample complexity: Algorithm uses $\mathcal{O}(k \log^2 p)$ samples of $\mathbf{y}$.*
2. *Computational complexity: Total number of operations to successfully decode all nonzero WHT coefficients or declare a decoding failure is $\mathcal{O}(k \log^3 p)$.*
3. *Success probability: Probability of recovering $\mathbf{Y}$ completely approaches 1 as p grows, where the probability is taken over randomness of selecting the support of $\mathbf{Y}$.*

This speedup is achieved by employing a divide-and-conquer strategy where we break the problem of recovering a k-sparse signal into k smaller problems of recovering 1-sparse signal, solve each 1-sparse problem efficiently, and then combine the solutions to each of them to recover the original signal. The recovery algorithm is closely tied to decoding a sparse-graph code through *peeling* using techniques from the literature on Low Density Parity Check (LDPC) codes [37] and product codes [38].

Note that under the assumptions of the theorem, theoretically, order of $k \log(p)$ samples are required for learning the correct model by information theoretic arguments [24]. The algorithm described here, which requires $k \log^2(p)$ samples is off from order optimality by only a logarithmic factor. As a matter of fact, the algorithm can be tweaked to be order optimal [24]. However, that version of the algorithm is not described in this paper as it requires a complex additional step.

The first step of the algorithm is to generate linear mixing of transform domain coefficients based on the following property.

Property 1. *Let* $\mathbf{y}$ *be a* $p = 2^d$ *length vector. Given a shift vector* $\mathbf{q} \in \mathbb{F}_2^d$ *and a full-rank subsampling matrix* $\mathbf{H} \in \mathbb{F}_2^{b \times d}$*, let* $\mathbf{z}$ *be the vector of length* $B = 2^b$ *where* $\mathbf{z}_\mathbf{x} = \mathbf{y}_{\mathbf{xH}+\mathbf{q}}$ *for all* $\mathbf{x} \in \mathbb{F}_2^b$*. Then, the WHT coefficients of* $\mathbf{z}$ *satisfy*

$$\mathbf{Z}_\mathbf{k} = \sqrt{\frac{B}{p}} \sum_{\mathbf{j} \in \mathbb{F}_2^p : \mathbf{j}\mathbf{H}^\top = \mathbf{k}} (-1)^{\langle \mathbf{q}, \mathbf{j} \rangle} \mathbf{Y}_\mathbf{j}, \tag{10}$$

where $\mathbf{Y}_\mathbf{j}$ *is the* $\mathbf{j}^{th}$ *WHT coefficient of* $\mathbf{y}$*.*

The above property states that the WHT coefficients $\mathbf{Y}_\mathbf{k}$ are modulated by $(-1)^{\langle \mathbf{q}, \mathbf{k} \rangle}$ when a shift of $\mathbf{q}$ is applied to the indices of $\mathbf{y}$, and that subsampling of the input signal creates a linear mixing of WHT coefficients.

Using Property 1 we create linear mixing of coefficients by choosing C many subsampling matrices $\mathbf{H}_1, \cdots, \mathbf{H}_C$ where each matrix is $b \times d$ dimensional. Furthermore, we choose for each subsampling $\mathbf{P}_1, \cdots, \mathbf{P}_C$ shift matrices where each of them is $\mathcal{O}(\log^2 p) \times d$ dimensional. The choice of C, the matrices $\mathbf{H}_i$ and the delays $\mathbf{P}_i$ for $i = 1, \cdots, C$ are going to be described in the following sections. Then WHT coefficients are calculated for the shifted-and-subsampled sequences. We give an example below for the linear mixing resulting from subsampling.

Example 1. *Let* $\mathbf{y}$ *be a vector of length* 16*, and let us define* $\mathbf{z}_\mathbf{x}^{(1)} = 2\mathbf{y}_{\mathbf{H}_1\mathbf{x}}$ *and* $\mathbf{z}_\mathbf{x}^{(2)} = 2\mathbf{y}_{\mathbf{H}_2\mathbf{x}}$ *where*

$$\mathbf{H}_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \mathbf{H}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

From property 1, we see that all the WHT coefficients of $\mathbf{y}$ *whose binary index have the same last two digits is hashed to the same bin (underlined in the following equations) for* $\mathbf{z}^{(1)}$*, that is, we have*

$$\begin{aligned}
\mathbf{Z}_{00}^{(1)} &= \mathbf{Y}_{00\underline{00}} + \mathbf{Y}_{01\underline{00}} + \mathbf{Y}_{10\underline{00}} + \mathbf{Y}_{11\underline{00}}, \\
\mathbf{Z}_{01}^{(1)} &= \mathbf{Y}_{00\underline{01}} + \mathbf{Y}_{01\underline{01}} + \mathbf{Y}_{10\underline{01}} + \mathbf{Y}_{11\underline{01}}, \\
\mathbf{Z}_{10}^{(1)} &= \mathbf{Y}_{00\underline{10}} + \mathbf{Y}_{01\underline{10}} + \mathbf{Y}_{10\underline{10}} + \mathbf{Y}_{11\underline{10}}, \\
\mathbf{Z}_{11}^{(1)} &= \mathbf{Y}_{00\underline{11}} + \mathbf{Y}_{01\underline{11}} + \mathbf{Y}_{10\underline{11}} + \mathbf{Y}_{11\underline{11}}.
\end{aligned}$$

Similarly, for $\mathbf{z}^{(2)}$ *we get*

$$
\begin{aligned}
\mathbf{Z}^{(2)}_{00} &= \mathbf{Y}_{\underline{00}00} + \mathbf{Y}_{\underline{00}01} + \mathbf{Y}_{\underline{00}10} + \mathbf{Y}_{\underline{00}11},\\
\mathbf{Z}^{(2)}_{01} &= \mathbf{Y}_{\underline{01}00} + \mathbf{Y}_{\underline{01}01} + \mathbf{Y}_{\underline{01}10} + \mathbf{Y}_{\underline{01}11},\\
\mathbf{Z}^{(2)}_{10} &= \mathbf{Y}_{\underline{10}00} + \mathbf{Y}_{\underline{10}01} + \mathbf{Y}_{\underline{10}10} + \mathbf{Y}_{\underline{10}11},\\
\mathbf{Z}^{(2)}_{11} &= \mathbf{Y}_{\underline{11}00} + \mathbf{Y}_{\underline{11}01} + \mathbf{Y}_{\underline{11}10} + \mathbf{Y}_{\underline{11}11}.
\end{aligned}
$$

Under the assumptions of Theorem 7.1 on sparsity and the support of the non-zero WHT coefficients of the signal, the linear mixing of coefficients take a form where they can be solved for through *peeling*. The following provides an example of such linear mixing.

Example 2. *Let* $\mathbf{y} \in \mathbb{R}^{16}$ *have WHT coefficients equal to*

$$
\mathbf{Y_k} = \begin{cases}
\mathbf{Y}_{0001} & \textit{if } \mathbf{k} = 0001,\\
\mathbf{Y}_{0100} & \textit{if } \mathbf{k} = 0100,\\
\mathbf{Y}_{0101} & \textit{if } \mathbf{k} = 0101,\\
\mathbf{Y}_{1010} & \textit{if } \mathbf{k} = 1010,\\
0 & \textit{otherwise.}
\end{cases}
$$

Under the subsampling used in example 1 the WHT coefficients of the sub-sampled vectors satisfy

$$
\begin{aligned}
\mathbf{Z}^{(1)}_{00} &= \mathbf{Y}_{01\underline{00}}, & \mathbf{Z}^{(2)}_{00} &= \mathbf{Y}_{\underline{00}01},\\
\mathbf{Z}^{(1)}_{01} &= \mathbf{Y}_{00\underline{01}} + \mathbf{Y}_{01\underline{01}}, & \mathbf{Z}^{(2)}_{01} &= \mathbf{Y}_{\underline{01}00} + \mathbf{Y}_{\underline{01}01},\\
\mathbf{Z}^{(1)}_{10} &= \mathbf{Y}_{10\underline{10}}, & \mathbf{Z}^{(2)}_{10} &= \mathbf{Y}_{\underline{10}10},\\
\mathbf{Z}^{(1)}_{11} &= 0, & \mathbf{Z}^{(2)}_{11} &= 0.
\end{aligned}
$$

We give the details of peeling algorithm in reference to this example in the following section.

7.2 Recovery Through Peeling with an Oracle

The relationship between the measurements and the unknown coefficients can be shown as a bipartite graph. The graph related to the linear mixing in Example 2 and the recovery of the non-zero coefficients are illustrated in Figure S1. The unknown coefficients are shown on the left and referred to as *variable nodes*, and the measurements are shown on the right and referred to as *check nodes*. An edge is drawn

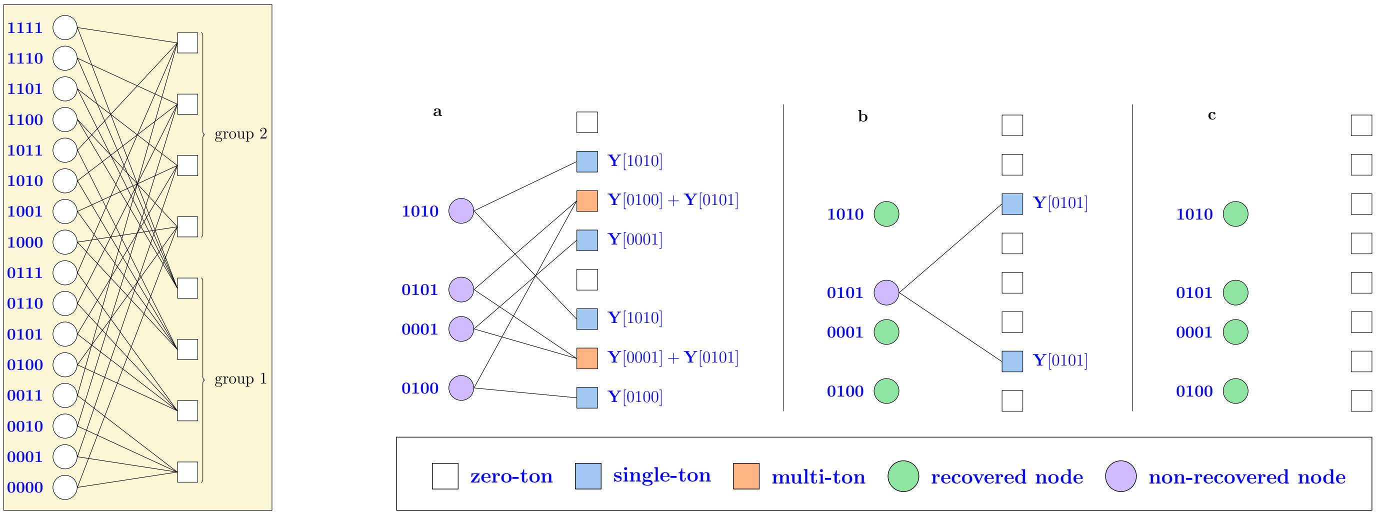

Figure S1: **(Left)** The connections between the variable nodes (WHT coefficients) and the check nodes (measurements) in Example 1. **(Right)** Recovering the unknown coefficients in Example 2. The graph induced by the non-zero coefficients is shown in **a**. In the first round of peeling we recover coefficients at indices 0100, 001 and 1010, and get the graph in **b**. In two rounds of peeling, all the non-zero elements of the signal are recovered as shown in **c**.

between a variable node and a check node if the unknown coefficient related to that variable node contributes to the measurement related to that check node. Each check node can be categorized into the following three types:

1. Zero-ton: a check node is a zero-ton if it has no non-zero coefficients (shaded in white in Figure S1).
2. Single-ton: a check node is a single-ton if it involves only one non-zero coefficient (shaded in blue in Figure S1). Specifically, we refer to the index k and its associated value $\mathbf{Y_k}$ as the index-value pair $(\mathbf{k}, \mathbf{Y_k})$.
3. Multi-ton: a check node is a multi-ton if it contains more than one non-zero coefficient (shaded in orange in Figure S1).

To illustrate the peeling algorithm for recovery, we assume that there exists an "oracle" that informs the decoder exactly which check nodes are single-tons, and provides the index-value pair for that single-ton. In Example 2, in the first round of peeling (shown in Panel (A) in Figure S1), the oracle informs the decoder that the check nodes corresponding to $\mathbf{Z}_{00}^{(1)}$, $\mathbf{Z}_{10}^{(1)}$, $\mathbf{Z}_{00}^{(2)}$, and $\mathbf{Z}_{10}^{(2)}$ are single-tons with index-value pairs $(0100, \mathbf{Y}_{0100})$, $(1010, \mathbf{Y}_{1010})$, $(0001, \mathbf{Y}_{0001})$ and $(1010, \mathbf{Y}_{1010})$ respectively. Then the decoder can subtract their contributions from other check nodes, forming new single-tons. Therefore, with the oracle information, the peeling decoder repeats the following steps:

1. select all the edges in the bipartite graph with right degree 1 (identify single-ton bins);

2. remove (peel off) these edges as well as the corresponding pair of variable and check nodes connected to these edges;
3. remove (peel off) all other edges connected to the variable nodes that have been removed in Step 2.
4. subtract the contributions of the variable nodes from the check nodes whose edges have been removed in Step 3.

Decoding is successful if all the edges are removed from the graph.

In this work we choose the subsampling matrices uniformly at random over $\mathbb{F}^{b\times d}$. Other constructions alongside with their theoretical guarantees can be found at [24, 39]. We chose the random design as it is observed to have superior practical performance in some regimes of interest [39, 40].

Since the proof of the algorithm follows the same steps as in [24], we just provide a sketch here and refer the interested readers to that paper. Since the sparsity is uniformly distributed, each non-zero entry of $\mathbf{Y}$ is connected to a check node chosen uniformly at random in each subsampling group. This results in a left-regular LDPC code construction, and the proof for recovering the support $\mathbf{Y}$ follows the same steps in [24].

Table S1: Thresholds for recovery [24]. M : number of check nodes, k : number of variable nodes (sparsity).

groups	3	4	5	6
M/k	1.2218	1.2949	1.4250	1.5697

In peeling, we recover a variable node (non-zero coefficient of $\mathbf{Y}$) if it is connected to a check node with degree 1, and remove the outgoing edges from that variable node. The *density evolution* is a powerful tool in modern coding theory that tracks the average density of remaining edges in the graph after ℓ rounds of peeling [37]. The density evolution equations for our setting is given by the recursive equation

$$p_\ell = \left(1 - e^{-vp_{\ell-1}/(M/k)}\right)^{v-1}, \tag{11}$$

where $p_0 = 1$, and M is the total number of parity check nodes. This assumes that the depth ℓ neighborhood of the chosen edge is a tree. We can show similarly to [24] that the depth ℓ neighborhood of a randomly chosen edge is a tree with high probability for any fixed ℓ. On average, an arbitrarily large fraction of edges are removed if p_ℓ goes to zero as $\ell \to \infty$. For p_ℓ to go to zero, M/k needs to be greater than a threshold for a fixed v. These thresholds are shown in Table S1. Then, one can use the standard Doob's martingale argument to show that the fraction of non-recovered components concentrates around it's mean [41]. This guarantees recovery of arbitrarily-large fraction of significant components. Then, an expander-graph argument is used to show that peeling continues until all of the coefficients are recovered [24].

7.3 Replacing the oracle

We now show how to replace the *oracle* in the peeling algorithm with a realizable mechanism. This is done by employing $\mathcal{O}(\log^2 p)$ shifts for each subsampling matrix where $\log(p)$ shifts are to recover each digit of the location $\mathbf{k}$, and we take $\mathcal{O}(\log p)$ samples for each location for noise averaging. Let $\mathbf{U}_{\mathbf{H},\mathbf{q}}(\mathbf{k})$ be the $\mathbf{k}^{\text{th}}$ WHT coefficient of the signal obtained by shifting indices of $\mathbf{y}$ by $\mathbf{q}$ and then subsampling by $\mathbf{H}$. From Property 1 we have

$$\mathbf{U}_{\mathbf{H},\mathbf{q}}(\mathbf{k}) := \sqrt{\frac{B}{p}} \sum_{\mathbf{j}:\mathbf{j}\mathbf{H}^\top=\mathbf{k}} (-1)^{\langle \mathbf{j},\mathbf{q}\rangle} \mathbf{Y}_\mathbf{j}. \tag{12}$$

Furthermore, let us define the ratio of a WHT coefficient obtained by using the same subsampling matrix but using two different shifts

$$\mathbf{r}_{\mathbf{A},\mathbf{p},\mathbf{q}}(\mathbf{k}) := \frac{\mathbf{U}_{\mathbf{A},\mathbf{p}+\mathbf{q}}(\mathbf{k})}{\mathbf{U}_{\mathbf{A},\mathbf{p}}(\mathbf{k})}. \tag{13}$$

Assume that for a WHT index $\mathbf{k}$ in equation (12), there is only one index $\mathbf{j}$ such that $\mathbf{A}^\top\mathbf{j} = \mathbf{k}$ and $\mathbf{Y}_\mathbf{j} \neq 0$ (that is, the check node corresponding to it is a single-ton). Then, it follows that $\mathbf{U}_{\mathbf{A},\mathbf{p}}(\mathbf{k}) = \sqrt{\frac{B}{p}}(-1)^{\langle \mathbf{j},\mathbf{p}\rangle}\mathbf{Y}_j$. Using $\mathbf{q} = \mathbf{e}_i \in \mathbb{F}^d$ (the vector with all indices equal to 0 except for the ith index which is equal to 1) in equation (13) yields

$$\mathbf{r}_{\mathbf{A},\mathbf{p},\mathbf{e}_i}(\mathbf{k}) = \frac{(-1)^{\langle \mathbf{j},\mathbf{p}+\mathbf{e}_i\rangle}\mathbf{Y}_\mathbf{j}}{(-1)^{\langle \mathbf{j},\mathbf{p}\mathbf{Y}_\mathbf{j}\rangle}} = (-1)^{\langle \mathbf{j},\mathbf{e}_i\rangle}. \tag{14}$$

Note that this value is in $\{-1,+1\}$ for all p if there is no noise. As the value of $\langle \mathbf{j},\mathbf{e}_i\rangle$ is equal to the i^{th} index of the location $\mathbf{j} \in \mathbb{F}_2^d$, by using shifts $\{\mathbf{e}_i\}_{i=0}^{d-1}$ going through all indices of $\mathbf{j}$ we can recover it. When there is noise, it can be shown that by taking $\mathcal{O}(\log p)$ random shifts, the probability of detecting the location wrongly can be made polynomially small [24].

7.4 Related works

Fourier attribution priors. Sparse epistatic regularization in Epistatic Net (EN) is conceptually related to a recent work describing Fourier-transform-based attribution priors in deep neural networks (DNNs) [42]. It has been observed that, in the context of mapping DNA sequence to transcription factors (TF) binding and chromatin accessibility profiles, penalizing high-frequency components of the

Fourier spectrum, can improve the stability, interpretability, and performance of DNNs. The focus of this work is, however, on the regularization of DNN to promote *sparsity* in Fourier basis. In fact, our results show that other forms of regularization in the spectral domain (*e.g.*, ℓ_2-norm instead of the ℓ_1-norm) are not beneficial for protein function prediction. Also distinct from these works, our regularization has a semisupervised flavor in imposing the ℓ_1-norm loss over the WH transform of the entire DNN landscape, that is, the combinatorial space of proteins which includes the ones that have not been observed in the training set.

Theoretical of sparse recovery. From a theory perspective, our regularization scheme relates to sparse recovery algorithms and compressed sensing. One distinction is that the fitness function in proteins do not exactly follow the exact sparse signal model in compressed sensing with added Gaussian noise. Therefore, the classical compressed sensing bounds would not directly apply to the problem in practice. Approximate guarantees for sparse recovery would be an interesting theoretical direction especially in light of the improvements that we have observed over Lasso in terms of sample complexity with EN regularization. We speculate that DNN has an internal inductive bias in favor of natural fitness functions in biology that enable us to reduce the effective dimensioanlity of the problem and thus improve the sample complexity bounds over Lasso. More theoretical studies in this regard is deferred to future works.

Peeling algorithm. Our work also suggests a new method to generalize the use-case of recent peeling-decoding algorithms [22, 25, 43, 44] for sparse-Fourier (WH) recovery problems to settings where we do not have the luxury to select (*i.e.*, design) the sampling patterns based on codes. In such physically-constrained sampling scenarios, DNN can be trained on the data at hand and serve as a "jump-start" that interpolates the data so that it can be queried at any binary patterns. The SGD algorithm converges to a point in DNN that will induce some aliasing effect over the signal that would be interesting to be studied theoretically. Transfer learning has recently emerged as a powerful technique in training deep neural notworks in low-sample regime. In protein design, it has been shown that [45] one can use the wealth of unsuperivised protein data to find a new representation for proteins. Such representation enables training a neural network using handful of proteins for design purposes. In our paper we do consider any external unsupervised data. However it is an interesting question to investigate how much of the power gained in the new representation could have been explained by the sparsity assumption in WH basis.

7.5 Synthetic sparse fitness landscapes

We assessed the performance of our ℓ_1-norm WH (EN)-regularized DNN algorithm on three sets of easy, medium, and hard data sets, each comprising 12 synthetic fitness landscapes with $n = 13$ mutations. In terms of dimensions, we followed the real-world protein landscape of [3], however, we changed the order and type of interactions and their weightings. We considered sparse protein landscapes with $k = 8$ non-zero WH coefficients. We sampled the interactions randomly (with a uniform distribution) from a subset of WH coefficients with up to 2^{nd}-order interactions for the "easy" data set. We selected one of the interactions randomly and replaced it by a random high-order interaction to make the "medium" data set and selected an additional three random interactions and replaced them by three random high-order interactions to make the "hard" data set. In all cases we set the weights of coefficient to be equal. We split the landscape (of size 8192) into training, validation, and test sets of sizes $n = 40$, 1000, and 1000, respectively. Fig. S2 shows the average accuracy of the algorithms in predicting fitness over 5 repeats of the experiments with random splits of the data into training/validation/test sets. Our ℓ_1-norm WH-regularized DNN consistently outperforms the DNN without WH regularization and the Lasso algorithm. The gap becomes even more distinct in presence of higher-order interactions.

Table S2: Description of the biological landscapes used in this paper is tabulated in terms of genotype, phenotype, number of mutations, input sequence size (d), and the reference to the publication.

	Genotype	**Phenotype**	# sites (d)	**Reference**
D1	Scattered genomic mutations	*E. coli* fitness	5 (5)	Khan *et al.* (2011) [9]
D2	Chromosomes in asexual fungi	*Aspergillus niger* fitness I	5 (5)	de Visser *et al.* (2009) [46]
D3	Protein point mutations	Resistance to β-lactam antibiotic	5 (5)	Weinreich *et al.* (2006) [7]
D4	Alleles in biosynthetic network	*S. cerevisiae* haploid growth rate	6 (6)	Hall *et al.* (2010) [8]
D5	Protein mutations	*Entacmaea quadricolor* fluorescence	13 (13)	Poelwijk *et al.* (2019) [3]
D6	Protein mutations	I-binding domain of protein G (GB1)	4 (80)	Wu *et al.* (2014) [1]
D7	Protein mutations	*Aequorea victoria* green fluorescence	236 (236)	Sarkisyan *et al.* (2016) [2]

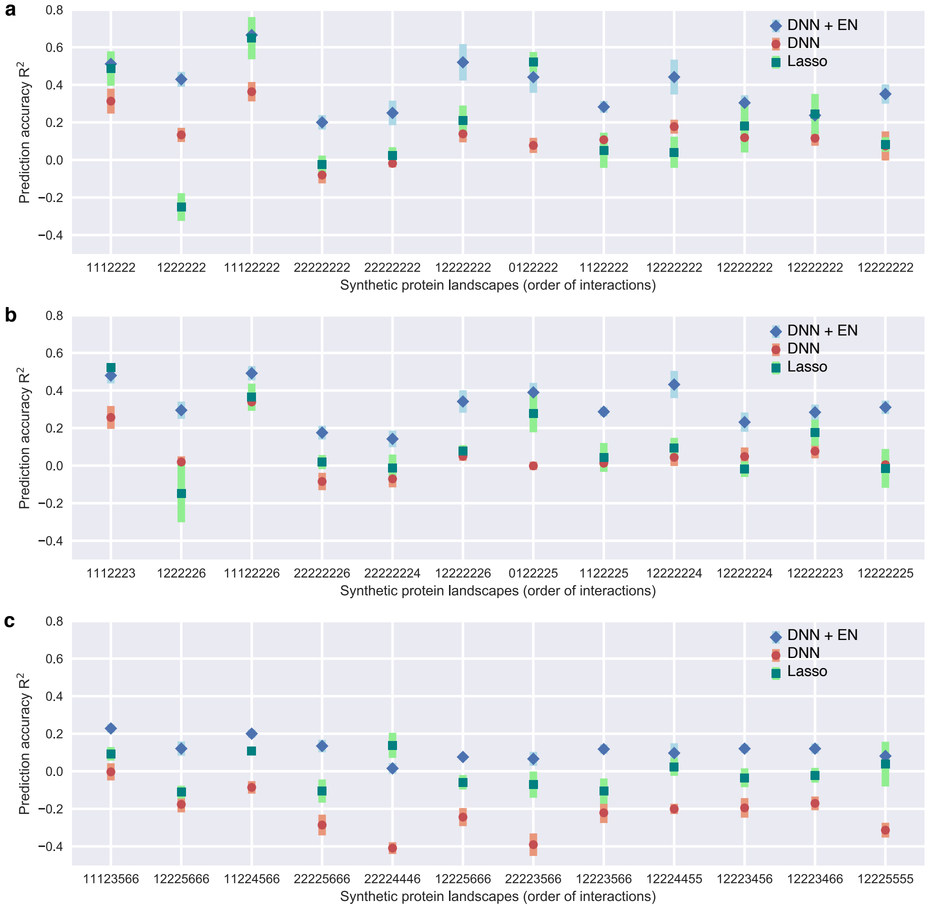

Figure S2: **Function prediction on synthetic landscapes with progressively more complex interactions.** **a**, Twelve landscapes with 8 interactions of up to order 2 are generated in WH basis and split randomly into training, validation, and test sets. Figure shows the prediction accuracy over the test set in 8 landscapes. Deep neural network (DNN) with epistatic net (EN) regularization outperforms DNN without regularization and Lasso regression. **b**, One of the interactions is selected from the landscapes in panel **a** at random and replaced with a high-order interactions. The experiments are repeated over the new more rugged landscapes. The prediction accuracy drops in all the algorithms due to ruggedness, however, DNN with EN regularization outperforms the competing baselines. **c**, An additional three interactions are selected at random and replaced with high-order interactions. DNN with WH-regularization has a consistently better or comparable prediction performance compared to DNN with no WH regularization and the Lasso algorithm in WH basis. All the experiments are repeated 5 times with random splits of the data into training, validation, and test sets. The error bars show the standard error of the mean (SEM).

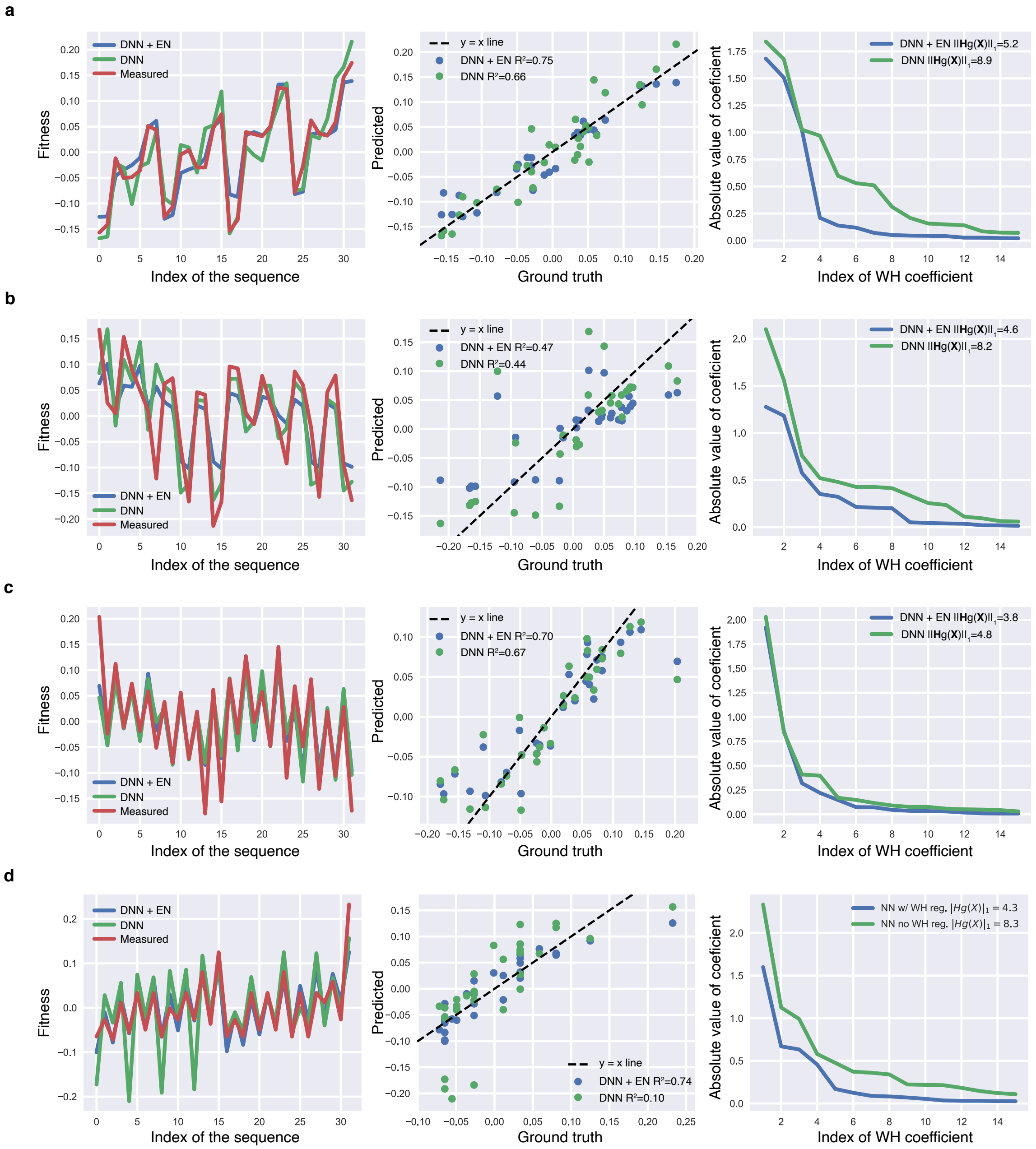

Figure S3: **Epistatic regularization in bacterial functions.** Analyzing the effect of the EN regularization on the WH transform and prediction accuracy of DNN trained on the fitness landscapes of **a**, Khan *et al.* [9], **b** and **c**, de Visser *et al.* [10, 46], **d**, Weinreich *et al.* [7] with $d = 5$ mutational sites.

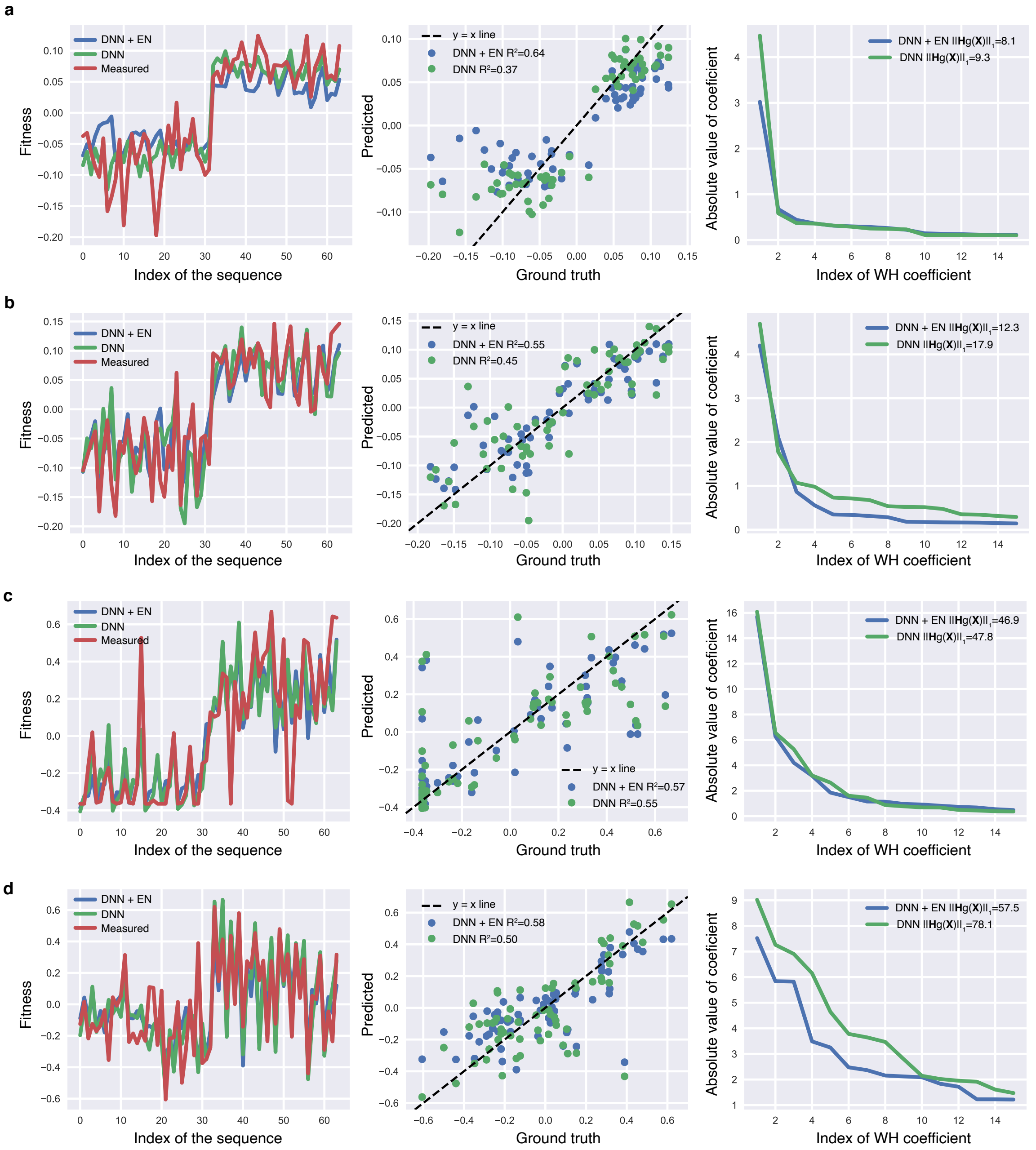

Figure S4: **Epistatic regularization in bacterial functions.** Analyzing the effect of EN regularization on the WH transform and prediction accuracy of DNN trained on the fitness landscapes of Hall *et al.* [8] with $d = 6$ mutational sites: **a**, Haploid growth, **b**, Diploid growth, **c**, Mating efficiency, and **d**, Sporulation.

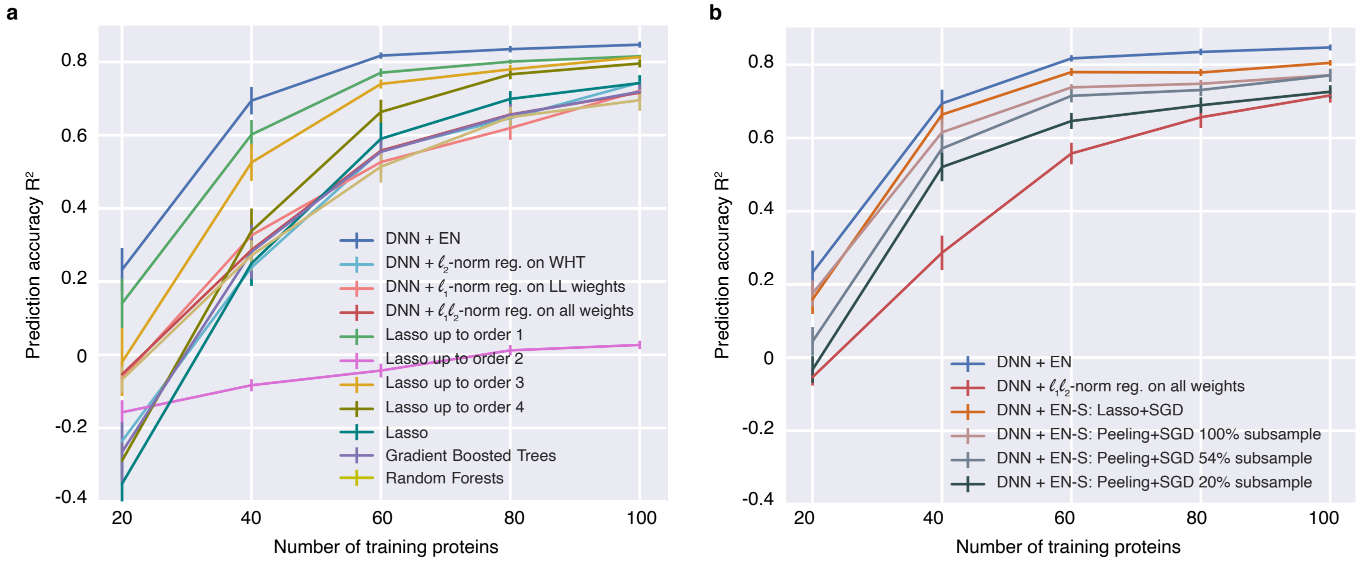

Figure S5: **Comparison of the prediction performance of DNN with the epistatic net (EN) regularization and DNN trained with other forms of sparsity promoting regularizes in the *Entacmaea quadricolor* fluorescense landscape. a**, EN regularization is compared with other forms of regularizations including the ℓ_2-norm regularization on the Walsh-Hadamard (WH) transform of DNN, ℓ_1 and ℓ_2-norm on the weights of DNN directly (elastic net regularization), and ℓ_1-norm on the last layer of DNN. **b**, Comparisons of the scalable EN methods (EN-S) under different setting, including a version which solves the **u**-minimization problem using Lasso (instead of peeling).

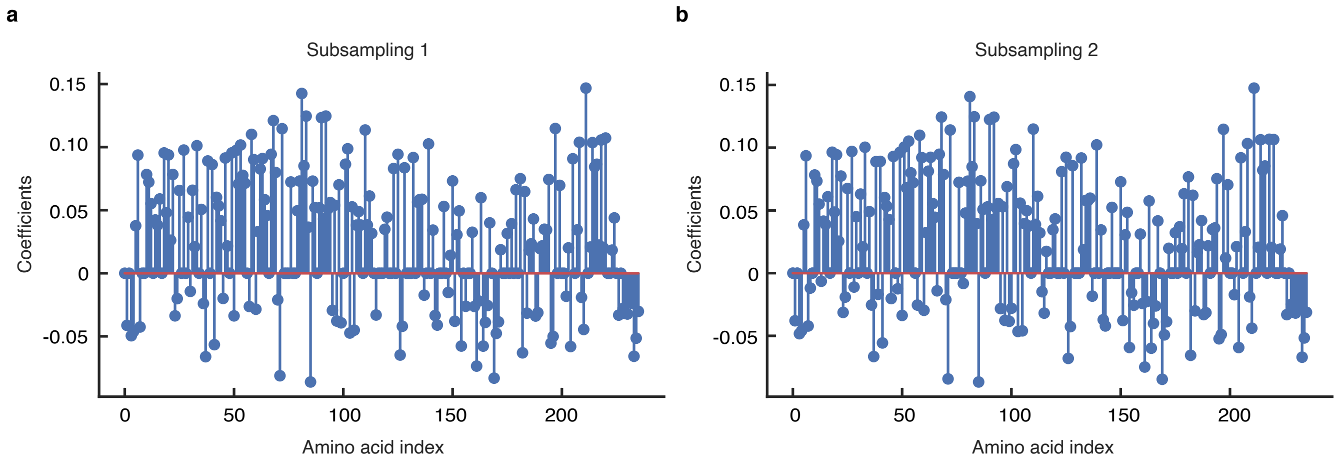

Figure S6: **Finding the epistatic interactions of deep neural networks (DNNs) using our epistatic net (EN) method under two independent subsamplings.** The coefficients of first-order interactions of DNN trained on the avGFP protein landscape recovered by EN-S is plotted under two independent subsamplings. The recovered coefficients are highly correlated ($R^2 = 0.99$) despite the enormous level of undersampling, that is, $5,074,944$ out of a total of 10^{71} sequences.

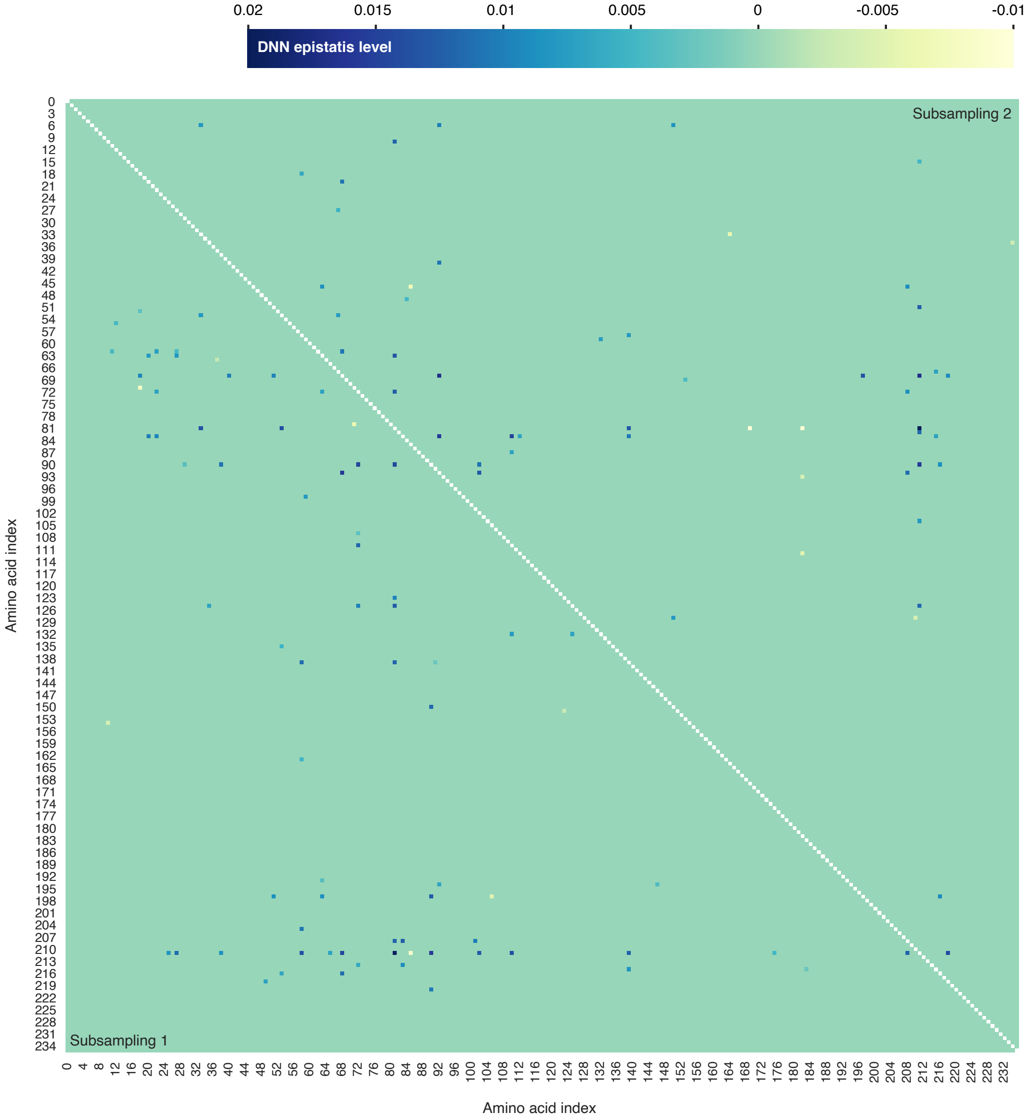

Figure S7: **Finding the epistatic interactions of deep neural networks (DNNs) using our epistatic net (EN) method under two independent subsamplings.** The coefficients of second-order interactions of DNN trained on the avGFP protein landscape recovered by EN-S is plotted under two independent subsamplings. The recovered coefficients are highly (locally-)correlated ($\mathrm{R}^2 = 0.60$) despite the enormous level of undersampling, that is, $5,074,944$ out of a total of 10^{71} sequences. Local correlation is found by evaluating the correlation between all the 3×3 sub-blocks of the interaction matrices.