MEAN DIMENSION OF GENERATIVE MODELS FOR PROTEIN SEQUENCES

Christoph Feinauer*
Department of Computing Sciences
Bocconi Institute for Data Science and Analytics (BIDSA)
Bocconi University
Milan, Italy
christoph.feinauer@unibocconi.it

Emanuele Borgonovo
Department of Decision Sciences
Bocconi University
Milan, Italy
emanuele.borgonovo@unibocconi.it

December 16, 2022

ABSTRACT

Generative models for protein sequences are important for protein design, mutational effect prediction and structure prediction. In all of these tasks, the introduction of models which include interactions between pairs of positions has had a major impact over the last decade. More recently, many methods going beyond pairwise models have been developed, for example by using neural networks that are in principle able to capture interactions between more than two positions from multiple sequence alignments. However, not much is known about the inter-dependency patterns between positions in these models, and how important higher-order interactions involving more than two positions are for their performance. In this work, we introduce the notion of mean dimension for generative models for protein sequences, which measures the average number of positions involved in interactions when weighted by their contribution to the total variance in log probability of the model. We estimate the mean dimension for different model classes trained on different protein families, relate it to the performance of the models on mutational effect prediction tasks and also trace its evolution during training. The mean dimension is related to the performance of models in biological prediction tasks and can highlight differences between model classes even if their performance in the prediction task is similar. The overall low mean dimension indicates that well-performing models are not necessarily of high complexity and encourages further work in interpreting their performance in biological terms.

1 Introduction

Generative models for protein sequences trained on datasets of homologous sequences are used widely for mutational effect prediction [1, 2, 3], pathogenicity prediction in humans [4], the creation of novel protein sequences [5], and structure prediction [6] and protein interaction prediction [7, 8]. While pairwise models that consider the compatibility between pairs of positions when assigning the probability to a sequence have been used and studied extensively in this field [9, 6, 2, 3, 10, 11], the question of how to include higher-order epistasis into these models is a topical research subject. While it is possible to manually add specific higher-order interactions to pairwise models [12], more recent works explore the use of neural networks, which can in principle capture arbitrary patterns from the underlying data. Architectures that have been proposed in literature are based for example on Variational Autoencoders [13, 1], GANs [14] and Transformer based architectures [15].

Nonetheless, it is often unclear what patterns these more complex models capture in the data and whether higher-order interactions play a key role in their performance. This is an important question, since disentangling biologically relevant effects from biases originating in phylogenetic relationships in the training sequence and data preprocessing pipelines might reveal interesting biological insights and lead to improved models. Some recent work has attempted to address this issue, for example by introducing quantities like the *pairwise saliency* and relating it to the performance in structural

*corresponding author

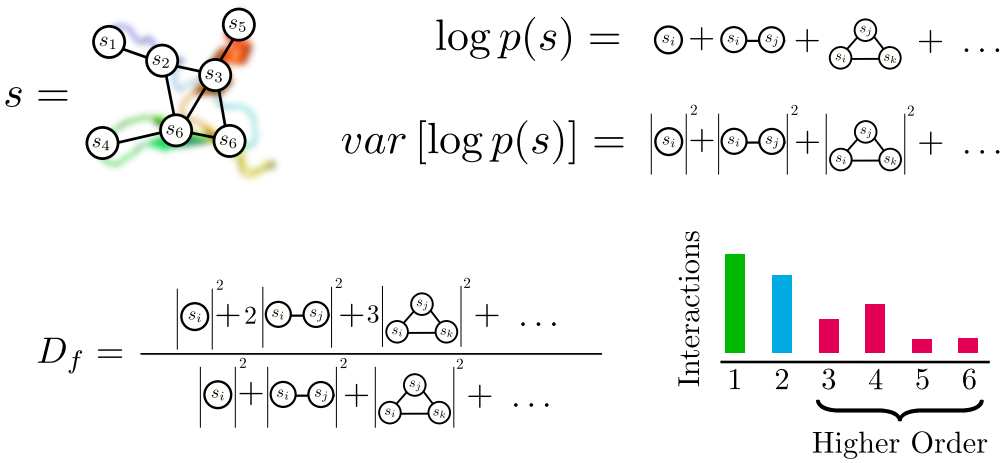

Figure 1: **Overview**: The figure represents a conceptual overview of the ideas used in this paper: The log probability $\log p(s)$ of a sequence s in a model can be expanded into terms of different orders. Under some assumptions on the expansion, the corresponding variance under the uniform distribution can be decomposed into contributions of different orders as well. The mean dimension is then defined as the average of orders under weights that correspond to contributions of orders to the total variance. The contribution of an order to the variance is proportional to the sum of squared interaction coefficients of that order. The mean dimension can therefore be conceptualized as an average over the histogram of squared interaction coefficients of different orders.

and mutational effect prediction [16], by extracting pairwise models from the original neural network models and comparing their performance [17], or by assessing how well-trained models reproduce sequence statistics [18].

In this work, we propose to analyze the *mean dimension* of generative protein sequence models, a concept that originates in statistics [19] and has recently been used for the analysis of neural networks trained for supervised prediction tasks [20, 21]. To extend the definition of mean dimension to probabilistic models $p(s)$ over amino acid sequences s, we define it as a weighted average over interaction orders k, where every order is weighted by its independent contribution c_k to the variance of $\log p(s)$ when s is sampled from the uniform distribution. We show below that, in this setting,

$$var[\log p(s)] = c_1 + c_2 + \ldots + c_N, \tag{1}$$

where N is the length of the amino acid sequences and c_k depends only on terms in $\log p(s)$ that involve k positions in the sequence. The mean dimension D_f can then be defined as

$$D_f = \frac{\sum_{k=1}^{N} k c_k}{var[\log p(s)]}, \tag{2}$$

where every interaction order is weighted by their contribution to the variance (see Fig. 1 for an illustration). For factorized models, where the log probability can be written as a sum of terms involving only a single position, the mean dimension is $D_f = 1$. For pairwise models, where the log probability contains terms depending on individual positions and position pairs, the mean dimension is $1 \leq D_f \leq 2$. For models that are dominated by higher-order interactions, we would expect a higher mean dimension. The mean dimension, therefore, allows us to assess the complexity of a model in terms of its interactions and, as we show in Methods, it can be estimated efficiently.

The mean dimension is a known tool in sensitivity analysis [19], and is also closely related to the *total influence* as defined for Boolean functions [22]. It has, however, to the best of our knowledge, never been applied to generative models for protein sequences. We derive the notion of mean dimension for such models in the framework of *energy-based models* (EBMs) of sequences of categorical variables in Methods. Given the specific framework and input domain, our derivation (see Methods) is different from typical approaches found in literature. To connect the present work to existing literature, we add in the Appendix a short review of the formalism as found in other works.

2 Methods

We assume a probabilistic, generative model that assigns a probability $p(s)$ to sequences of amino acids $s = (s_1, \ldots, s_N)$ of fixed length N. Here, the sequence element s_i corresponds to a single amino acid or an alignment gap, and we will denote by q the number of possible values any s_i can take (typically 20 different amino acids and a gap symbol, so $q = 21$). Such probability distributions underlie many popular generative models for protein sequences, for example Potts Models [9], Variational Autoencoders [1] or ArDCA [23].

2.1 Expansion

In this work, we will not analyse the probability distribution $p(s)$ directly, but a function proportional to the negative log probability $f(s) \sim -\log p(s)$, where we assume that $p(s) > 0$ for all s. The function $f(s)$ therefore maps sequences s of categorical variables of length N to a real value. We assume q to be the number of different categories, where a single category corresponds to an amino acid. Given $f(s)$, the probability distribution $p(s)$ can be recovered by

$$p(s) = \frac{\exp(-f(s))}{\sum_{s'} \exp(-f(s'))}, \tag{3}$$

which is the Boltzmann-Gibbs distribution with $f(s)$ as the energy. Here, the sum is over all possible sequences s'. We do not assume to be able to calculate the normalization factor corresponding to the denominator in Eq. 3, which is often not tractable, for example for Potts models.

A general expansion of f can be written as [17]

$$f(s) = \sum_{J \subseteq \mathcal{N}} f_J(s_J), \tag{4}$$

where we denote by $\mathcal{N} = \{1, \ldots, N\}$ the set of all positions, by J a subset of $\mathcal{N}$, by s_J the amino acids at positions in J and by f_J a function that depends only on the amino acids at positions in J. The sum is therefore over the powerset of $\mathcal{N}$.

While we can always expand a function $f(s)$ in this way, see the Appendix, the expansion in Eq. 4 is in general not unique. For $L \subset J$ we can for example redefine $f_J(s_J) \to f_J(s_J) + f_L(s_L)$ and $f_L(s_L) \to 0$ without changing the values of $f(s)$. Different ways of fixing these degrees of freedom are referred to as different *gauges* [11]. For the following, the choice of the *zero-sum* gauge is convenient, which is a popular choice for example for contact prediction [11]. In pairwise models, this gauge minimizes the Frobenius norm of pairwise interactions and leads to a representation in which as much as possible of the variance of $f(s)$ is explained by simple terms depending only on a single position. In this sense, it aims to find the *simplest* representation of the expansion.

In the context of the expansion in Eq. 4 the zero-sum gauge, also called the *Ising gauge* [24], can be defined as the representation in which the sum of any $f_J(s_J)$ over any of its dependencies is 0 for $J \neq \emptyset$. This condition can be written as

$$\sum_{s_i} f_J(s_J) = 0 \text{ if } i \in J, \tag{5}$$

where we sum over all possible values of s_i with the other dependencies of f_J arbitrary. Every expansion can be written in this representation, which we show in the Appendix. We assume this representation from now on, if not stated otherwise. In this representation, it is easy to see that with $\mathcal{U}$ being the uniform distribution, where every amino acid in s is sampled independently with probability $1/q$,

$$\mathbb{E}_{s\sim\mathcal{U}}\left[f_J(s_J) f_L(s_L)\right] = 0 \text{ if } J \neq L, \tag{6}$$

since the expectation, which contains a sum over all possible sequences, will contain a sum over a dependency of either $f_J(a_J)$ or $f_L(a_L)$ which is not a dependency of the other, resulting in 0 following Eq.5.

Similarly, the expectation of $f(s)$ itself under $\mathcal{U}$ is

$$\mathbb{E}_{s\sim\mathcal{U}}\left[f(s)\right] = \mathbb{E}_{s\sim\mathcal{U}}\left[\sum_{J\subseteq\mathcal{N}} f_J(s_J)\right] = f_\emptyset, \tag{7}$$

since the expectation for all terms except $f_\emptyset$ vanishes.

2.2 Mean Dimension for Generative Protein Sequence Models

We now define the mean dimension and begin by noting that the variance of $f(s)$, when sampling s from the uniform distribution $\mathcal{U}$, can be written as

$$\begin{aligned} var_{s\sim\mathcal{U}}[f(s)] &= \mathbb{E}_{s\sim\mathcal{U}}\left[\sum_{J\subseteq\mathcal{N}} f_J(s_J)\sum_{L\subseteq\mathcal{N}} f_L(s_L)\right] - \mathbb{E}_{s\sim\mathcal{U}}\left[\sum_{J\subseteq\mathcal{N}} f_J(s_J)\right]^2 \\ &= \sum_{J\subseteq\mathcal{N}} \mathbb{E}_{s\sim\mathcal{U}}\left[f_J^2(s_J)\right] + \sum_{L,J\subseteq\mathcal{N},J\neq L} \mathbb{E}_{s\sim\mathcal{U}}\left[f_J(s_J)f_L(s_L)\right] - \left[\sum_{J\subseteq\mathcal{N}} \mathbb{E}_{s\sim\mathcal{U}}\left[f_J(s_J)\right]\right]^2 . \end{aligned}$$

Assuming the zero-sum gauge defined by Eq. 5 and using the relations in Eq. 6 and Eq. 7, this leads to

$$var_{s\sim\mathcal{U}}[f(s)] = \sum_{J\subseteq\mathcal{N}} \mathbb{E}_{s\sim\mathcal{U}}\left[f_J^2(s_J)\right] - f_\emptyset^2 = \sum_{J\subseteq\mathcal{N}/\emptyset} \frac{1}{q^{|J|}} \sum_{s_J} f_J^2(s_J), \tag{8}$$

where $|J|$ is the cardinality of J and the order of the interaction represented by f_J. This can be written as a sum over interaction orders k as

$$var_{s\sim\mathcal{U}}[f(s)] = \sum_{k=1}^{N} c_k, \tag{9}$$

where we defined

$$c_k := \sum_{J:|J|=k} \frac{1}{q^k} \sum_{s_J} f_J^2(s_J), \tag{10}$$

which is the contribution of order k to the variance, a sum over the averaged squared interactions coefficients corresponding to interactions of order k.

We can then define the weight w_k of order k as

$$w_k := \frac{c_k}{\sum_{\hat{k}=1}^{N} c_{\hat{k}}}, \tag{11}$$

which satisfies $w_k \geq 0$ for all k and $\sum_{k=1}^{N} w_k = 1$. Note that the weights are only defined if $var_{s\sim\mathcal{U}}[f(s)] > 0$, which we assume to be true in the following.

The mean dimension of interaction D_f of the function $f(s)$ can then be defined as the average of k with respect to these weights, leading to

$$\boxed{D_f = \sum_{k=1}^{N} k\, w_k = \frac{\sum_{k=1}^{N}\sum_{J:|J|=k} \frac{k}{q^k}\sum_{s_J} f_J^2(s_J)}{var_{s\sim\mathcal{U}}[f(s)]}} \tag{12}$$

which is the average of interaction orders when weighted by their contribution to the variance of $f(s)$ under the uniform distribution $\mathcal{U}$. It is easy to see that $N \geq D_f \geq 1$.

In the next section, we derive an efficient estimator for the mean dimension.

2.3 Estimating the Mean Dimension

We introduce the function $f^{-i}(s_{/i}) := \mathbb{E}_{s_i\sim U}[f(s)]$, which is the function $f(s)$ averaged over s_i while keeping $s_{/i}$ fixed, where $s_{/i}$ is the sequence s without s_i. When replacing $f(s)$ with the expansion in Eq. 4 and using the definition of the zero-sum gauge in Eq. 5, we get

$$f^{-i}(s_{/i}) = \sum_{J:i\notin J} f_J(s_J), \tag{13}$$

and, conversely,

$$f(s) - f^{-i}(s_{/i}) = \sum_{J: i \in J} f_J(s_J). \tag{14}$$

The same computation as for the variance above leads us to

$$I_i := var_{s\sim\mathcal{U}}[f(s) - f^{-i}(s_{/i})] = \sum_{J: i\in J} \frac{1}{q^{|J|}} \sum_{s_J} f_J^2(s_J), \tag{15}$$

where I_i can be interpreted as the part of the variance of $f(s)$ that involves position i, also called the *influence* of i [22]. When summing the expression over i, a term involving the subset of positions J will appear exactly $|J|$ times in the sum.

Therefore we can write

$$\sum_{i=1}^{N} I_i = \sum_{J\neq\emptyset} \frac{|J|}{q^{|J|}} \sum_{s_J} f_J^2(s_J) = \sum_{k=1}^{N} \sum_{J:|J|=k} \frac{k}{q^k} f_J^2(s_J), \tag{16}$$

which is the numerator of the mean dimension in Eq. 12.

We finally arrive at

$$D_f = \frac{\sum_{i=1}^{N} var_{s\sim\mathcal{U}}[f(s) - f^{-i}(s_{/i})]}{var_{s\sim\mathcal{U}}[f(s)]}, \tag{17}$$

which can be approximated using samples from the uniform distribution $\mathcal{U}$. We will present a simple algorithm based on this identity in Methods.

3 Results

In the following we train different models on the MSAs of protein families and estimate their mean dimension, together with their performance on the task of mutational effect prediction. We take the data from [1], which contains 44 MSAs together with data from deep mutational scanning (DMS) experiments in the form of experimentally determined fitness changes following amino acid substitutions with respect to a wild-type protein. Since different experiments report different proxies for fitness, we follow the common approach and report Spearman correlation values of probabilities of mutated sequences in the models and the experimental measurements. We summarize the characteristics of the MSAs in the Appendix.

3.1 Estimating the Mean Dimension

The estimator in Eq. 17 can be directly translated into an algorithm for estimating the mean dimension of a given function f. We present a possible algorithm in Alg. 1, where we run for T iterations, sampling at every iteration a sequence from the uniform distribution for every position i and update based on these samples stream estimators tracking the quantities in Eq. 17. Since we need to calculate averages over all possible amino acids at all positions, which requires q evaluations of f at all N positions, a single iteration corresponds to $N \cdot q$ evaluations of f and the total number of necessary evaluations is $T \cdot N \cdot q$. Since every iteration can be seen as an independent estimate, we used bootstrap samples over the iterations to estimate the standard deviation of the final value. We found $T = 1000$ to be sufficient for reliable estimates for the protein families analyzed in the later sections.

We provide an implementation of this algorithm in the accompanying repository [2]. There we implement several further improvements and variations, for example parallelizing the loops or decoupling the sampling of sequences and the evaluation of f from the variance estimation, which allows for easier bootstrapping. The resulting estimator is, however, equivalent to Alg. 1.

[2] `https://github.com/christophfeinauer/ProteinMeanDimension`

Algorithm 1: Algorithm for Estimating the Mean Dimension D_f

Input:
f: Function
N: Sequence length
M: Number of iterations
q: Number of different amino acids
Result:
D_f: Mean dimension of f
I_i for $i = 1, \dots N$: Influences defined in Eq. 15
VarianceStreamEstimator v ; `// stream estimator for total variance`
VarianceStreamEstimator $I[N]$; ; `// array of` N `stream estimators for` N `variances (influences)`
for $t = 1 \dots T$ **do**
  for $i = 1 \dots N$ **do**
    $r = zeros(q)$; `// array of` q `zeros`
    $s = rand(q,\ length = N)$; `// set` s `to random sequence`
    for $a = 1 \dots q$ **do**
      $s[i] \leftarrow a$;
      $r[a] \leftarrow f(s)$;
    end
    $a \leftarrow rand(q)$; `// set a to a random number between 1 and` q
    $I[i] < (r[a] - mean(r))$; `// update estimate of influence` I_i
    $v < r[a]$; `// update estimate of total variance`
  end
end
return $sum(I)/v,\ I$

3.2 ArDCA

ArDCA [23] is a model for protein sequences that uses the autoregressive decomposition

$$p(s) = \prod_{i=1}^{N} p(s_i|s_{<i}), \tag{18}$$

where $p(s_i|s_{<i})$ is the probability of the amino acid s_i in position i given the part of the sequence that precedes position i, denoted by $s_{<i}$. This probability is then defined as

$$p(s_i|s_{<i}) = \frac{\exp\left(h_i(s_i) + \sum_{j=1}^{i-1} J_{ij}(s_i, s_j)\right)}{z_i(s_{<i})} \text{ with } z_i(s_{<i}) = \sum_{\hat{s}_i} \exp\left(h_i(\hat{s}_i) + \sum_{j=1}^{i-1} J_{ij}(\hat{s}_i, s_j)\right) \tag{19}$$

where $z_i(s_{<i})$ is a normalization factor depending on $s_{<i}$. The terms $J_{ij}(a, b)$ are called *couplings* and depend on two positions, while the terms $h_i(a)$ are called fields and depend on a single position. Due to the autoregressive nature of the model, the probability of a sequence can be calculated directly using the equations above, which is an advantage to Potts models where the normalization constant is intractable and only a function proportional to the (log-)probability can be calculated. Note that due to the dependence of the normalization factors $z_i(s_{<i})$ on the preceding part of the sequence the model is not a pairwise model and can include higher-order interactions.

We use the code provided by the authors for training [23]. Training includes regularization using an $L2$ penalization with a different strength for the J and h terms, called λ_J and λ_h respectively. The authors provide different values optimized for generative modeling and mutational effect prediction. Since we will test the trained models on the latter task, we use the parameters optimized for mutational effect prediction ($\lambda_J = 0.01$ and $\lambda_H = 0.0001$) except where stated otherwise.

We retrained ArDCA on all 44 MSAs and estimated the mean dimension of the resulting models using $f(s) = -\log p(s)$. Surprisingly, we found all of the mean dimensions to be very close to 1, with an average of 1.02 and a maximal value of 1.053, see Table 1 in the Appendix. We suspected this to be due to the regularization constants used, where the couplings J have a regularization constant that is two orders of magnitude larger than the corresponding constant for the fields h.

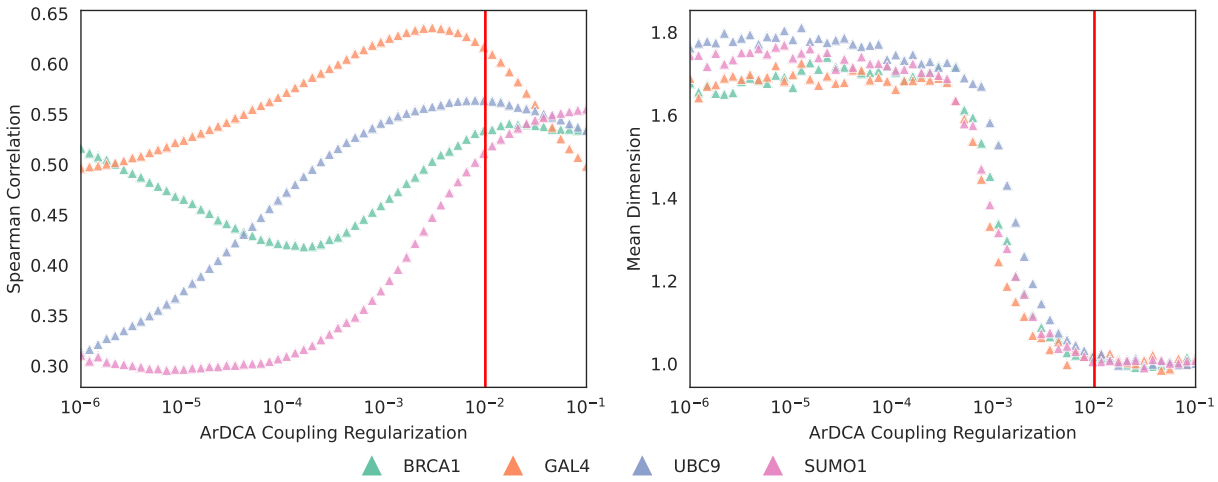

Figure 2: **ArDCA Mean Dimension and Spearman Correlation** *Left:* The Spearman correlation with measured mutational effects for ArDCA models with varying λ_J regularization. *Right*: Estimated mean dimension for the same ArDCA models. The red lines indicate the λ_J regularization optimized for mutational effect prediction.

From Eq. 19 it is clear that in the case $J \approx 0$ the model factorizes and can be written as $\log p(s) \approx \sum_{i=1}^{N} h_i(s_i) - \log Z$, where Z does not depend on the sequence. Such a model would have indeed $D_f \approx 1$.

In order to test this hypothesis we retrained the models for a subset of four MSAs for BRCA1, GAL4, UBC9, SUMO1 with λ_J varying logarithmically from 10^{-6} to 10^{-1}. The resulting mean dimension estimates and the performance in terms of mutational effect prediction can be seen in Fig. 2. Several interesting results emerge: For a large range of λ_J the mean dimension for all models is relatively stable between 1.6 and 1.8, after which it rapidly drops to close to 1. Interestingly, this drop is very similar for all models and happens the roughly the same value of λ_J. Furthermore, the λ_J value at which the mean dimension for all models stabilizes towards 1, at $\lambda_J = 0.01$, is exactly the value that the authors in [23] provided as optimal for mutational effect prediction. It seems therefore that, at least for mutational effect prediction, the best performing models are close to factorized models, where positions contribute almost independently to the log probability. Note, however, that this does not necessarily mean that a factorized model would perform equally to more complex models on the same data. The mean dimension is a measure taking into account *all* of the sequence space. Even if pairwise or higher-order interactions are comparatively small when viewed in this lens, they might still have a significant effect on the order of probabilities around a wild-type sequence.

Next we analyzed the influence I_i of positions as defined in Eq. 15, which measures the variance of f that involves position i. We show the influences in the right part of Fig. 3 for the ArDCA model trained on the GAL4 dataset. The influences vary significantly for different positions. We suspected that the influence of a position might be related to the evolutionary conservation of the position. We therefore calculated for every column in the training MSA the *site entropy* $\mathcal{S}_i$, defined as

$$\mathcal{S}_i = -\sum_{a=1}^{q} f_i(a) \log f_i(a), \tag{20}$$

where $f_i(a)$ is the frequency of amino acid a in the consensus column i in the MSA. The higher the site entropy for a position, the more variable it is. In the left part of Fig. 3 we plot the site entropies of positions against their influences for the same ArDCA model as above. It is apparent that there is a strong *anti*-correlation between these two quantities. This can be explained by noting that a strongly conserved position in the training MSA will most likely lead to a model that assigns very low probabilities to sequences that contain an uncommon amino acid in this position. This, in turn, will lead this position to contribute significantly to the variance in log probability in sequences sampled from the uniform distribution, where all amino acids have equal probabilities in the position. By contrast, if a position is highly variable and has a large site entropy in the MSA, the model will most likely assign similar marginal probabilities to many different amino acids in this position. The position might still exhibit a large influence on the total variance, but this would need to be mediated by pairwise or higher-order interactions. These, however, are small for the ArDCA models used here, as shown above.

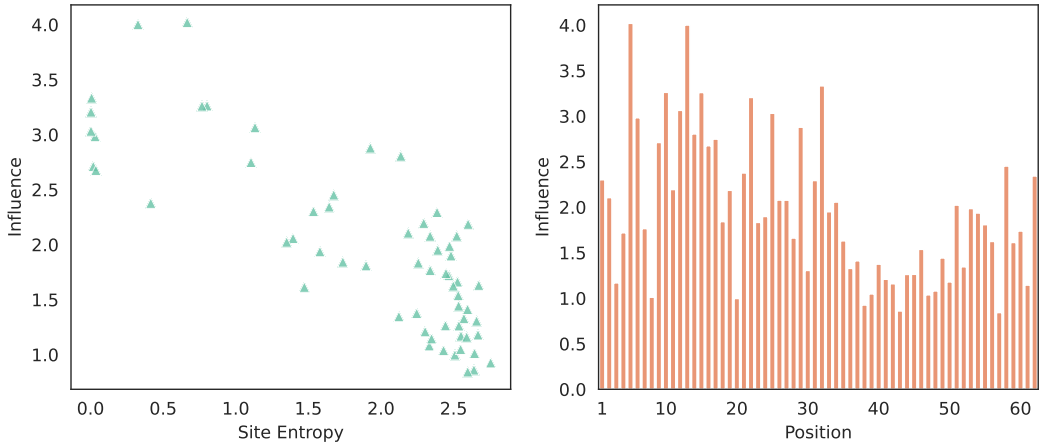

Figure 3: **Site Entropies and Estimated Influences for ArDCA trained on GAL4** *Left:* Site entropy of a position in the training dataset versus the estimated influence of the position in the model as calculated from Eq.15. *Right*: Influence for individual positions along the sequence. Both parts use the same model, trained with $\lambda_J = 0.01$ and $\lambda_h = 0.0001$.

3.3 Variational Autoencoder

Variational autoencoders (VAEs) are popular latent-space models for homologous protein data that have been shown to perform well in tasks like mutational effect prediction [1], the assessment of the pathogenicity of amino acid variants in humans [4], and the exploration of sequence space by analyzing the latent representations [13].

Following the VAE architecture presented in [13], we project the sequence s into a probability distribution over latent representation $z \in \mathcal{R}^d$ by using a probabilistic encoder MLP which takes a one-hot encoded version of s as input and outputs the means and log variances of a multivariate Gaussian with zero off-diagonal correlations. A second MLP, the probabilistic decoder, then transforms the latent representation to the inputs of a *softmax* layer, which defines probabilities of amino acids at all positions. Defining the prior over z, the model can be trained using the ELBO objective [25]. The probability of a sequence s can then be estimated using importance sampling [26]. We use a single-layer MLP with 40 hidden units and a $tanh$ activation function for both the encoder and decoder, and a latent representation with dimension $d = 5$, with an $L2$ penalization of strength 0.01 on all parameters. These values perform well in practice on mutational effect prediction. Following the setting in [13], we train in full batch mode for 10.000 epochs.

For calculating the log probabilities necessary for determining the mean dimension we use 5000 ELBO samples, which also leads to a good performance in mutational effect prediction. However, this means that a single evaluation of f in the estimator in Eq. 17 requires 5000 single evaluations of the neural network. We therefore do not estimate the mean dimension on all 44 MSAs but on the same subset that we used for a more focused analysis in the last section (the MSAs corresponding to BRCA1, GAL4, UBC9 and SUMO1). We calculate the mean dimension of the final models and, in order to get more insight about the training process, during training. We also monitor the Spearman correlation with the experimental data during the same checkpoints. We show both quantities for the four protein families in Fig. 4.

Focusing on the first 250 epochs, the behavior of the mean dimension is similar for all datasets: The initial mean dimension is close to 1 at epoch 0 (corresponding to the models with randomly initialized parameters), but then increases in the first 50 epochs to around 20 for BRCA1, GAL4 and SUMO1 and to about 80 for UBC9. This is interesting since it means that the models acquire strong higher-order components during the first phase of training. However, in the next 50 epochs the mean dimension decreases again to close to the initial values. In this first phase, the Spearman correlation with the experimental mutational data increases rapidly until about epoch 100. In a second phase, from about epoch 100 to 3000, the mean dimension remains relatively low for BRCA1, GAL4 and SUMO1, but shows another peak of height around 30 for UBC9. The Spearman correlation with the experimental mutational data fluctuates in this phase, for example increasing for BRCA1 to a peak and then decreasing again. After about epoch 4000 both the mean dimension and the Spearman correlation stabilizes. For SUM01, the Spearman correlation seems to decrease slightly until the end of training. We note that this MSA is particular as factorized models (so models with no pairwise or higher-order interactions and with mean dimension equal to 1) perform *better* than more complex models in terms of mutational

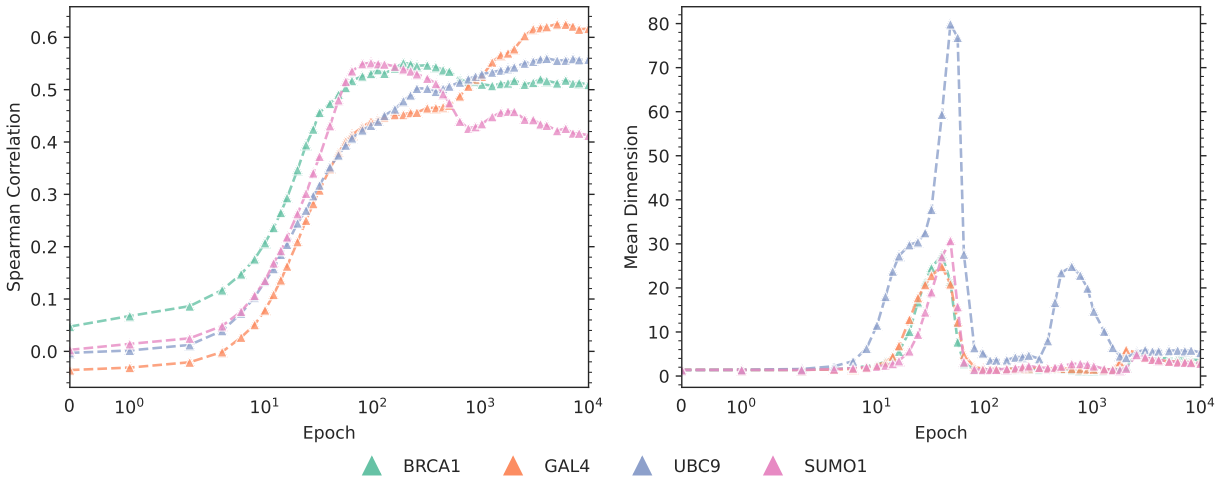

Figure 4: **Spearman Correlation with Experimental Mutational Data and Mean Dimension during Training for VAE models** *Left*: Spearman correlation with the experimental fitness effects for different protein families during training. *Right*: Estimated mean dimension for different protein families during training.

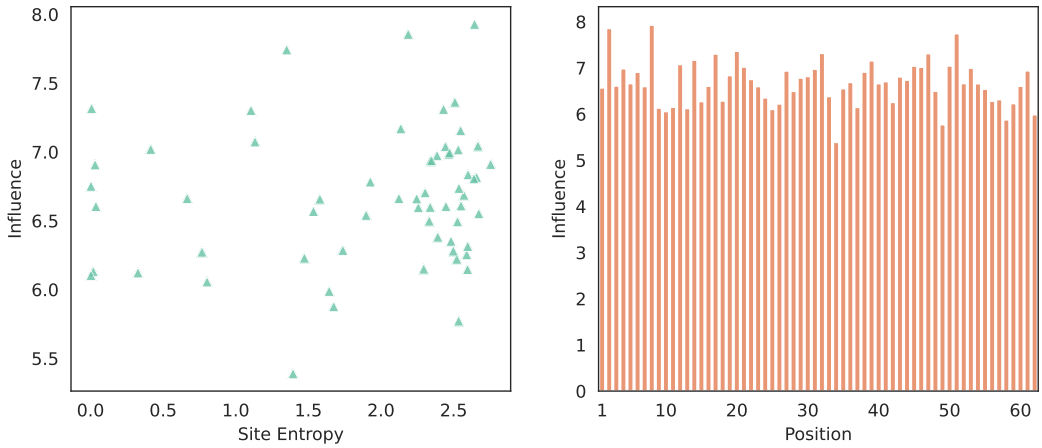

Figure 5: **Site Entropies and Estimated Influences for VAE trained on GAL4** *Left:* Site entropy of a position in the training dataset versus the estimated influence of the position in the model as calculated from Eq.15. *Right*: Influence for individual positions along the sequence.

effect prediction [1]. The final mean dimensions are significantly higher than the corresponding values for ArDCA, ranging from 2.8 for SUMO1 to 5.3 for UBC9, see Table 1 in the Appendix. This seems to indicate that the VAE models indeed implement higher-order interactions that are important for explaining the variance on sequences from the uniform distribution, in contrast to the ArDCA models. Since their performance in terms of mutational effect prediction is not better than for ArDCA models (compare Fig. 2), these higher-order interactions do not seem to be important for this task, however.

We also show the site entropy and the influences in Fig. 5. The situation is quite different than for ArDCA models in that there is no discernible correlation between the influence of a position and its site entropy in the MSA. In addition, the influences vary less among the positions. This is surprising since it indicates that all positions contribute equally to the model variance, independent of their conservation in the training set. In order to analyse this effect close we also calculate the variance in log probability when exchanging amino acids at a given position while keeping the rest of sequence fixed to a wildtype sequence from the training set. More precisely, we calculate the quantities

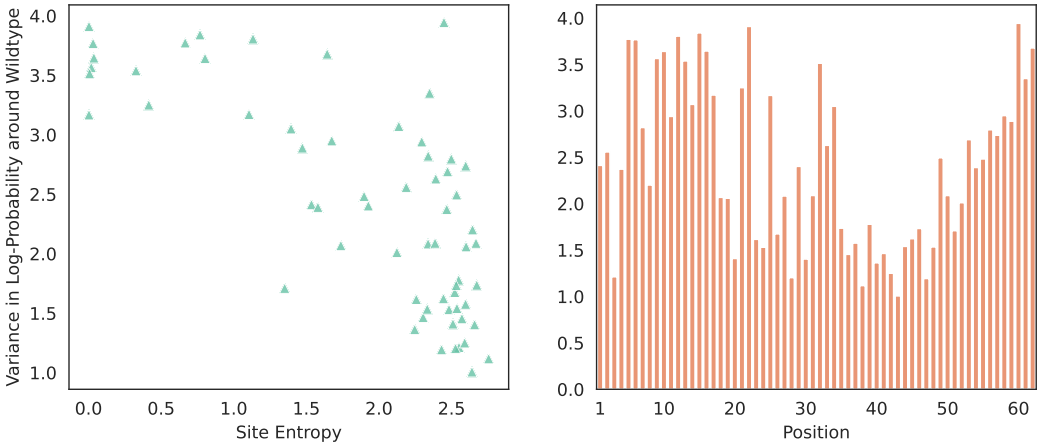

Figure 6: **Site Entropies and Variance in Log Probability around Wildtype for VAE trained on GAL4** *Left:* Site entropy of a position in the training dataset versus the variance in log probability as calculated from Eq. 21. *Right*: Variance in log probability around wild type for individual positions along the sequence.

$$v_i = var_{s_i \sim \mathcal{U}} \log p(s_i, s_{/i} = s^{wt}_{/i}), \tag{21}$$

for all positions i, where $(s_i, s_{/i} = s^{wt}_{/i})$ is a shorthand for the wildtype sequence s^{wt} with s^{wt}_i replaced by s_i and the variance is taken over s_i following the uniform distribution $\mathcal{U}$. Using the yeast sequence for GAL4 we show the quantities and their relation to the site entropy in Fig. 6. The variance in log probability is correlated with the site entropy and the distribution of this quantitiy along the sequence is very similar to the one obtained from ArDCA, see Fig. 3. Together, these results indicate that the influences of the positions are similar to each other when looking at the global distribution, but differentiate for parts of the sequence space closer the training set. We note that this is in line with the idea that VAEs trained on protein sequences behave like mixtures of factorized models, where the emitted factorized model depends on the coarse location in sequence space [16]. Such models might exhibit a large mean dimension due to the mixture, but have a very simple structure locally.

4 Discussion

In this work, we have introduced the mean dimension of interaction for generative protein sequence models, which measures the average order of interaction coefficients in a general expansion of the log probability assigned by the model to protein sequences. We derived a simple estimator of the mean dimension and an algorithm for the computation. We estimated the mean dimension of two model classes, ArDCA and the VAE, trained on MSAs of different protein families. We analyzed changes in mean dimension when modifying the regularization constants in ArDCA and during training for the VAE. The results indicate that the mean dimension of ArDCA models is close to 1 when using parameters optimized for mutational effect prediction and we never observed a mean dimension larger than 2 for this model class in any setting. For the VAE, the mean dimension is higher and we also showed it to be highly variable during training. We also highlighted that the mean dimension is related to the performance of the models in terms of mutational effect prediction.

We believe the mean dimension to be an interesting analysis tool in that it can be efficiently calculated while having a clear interpretation in the context of higher-order interactions, which are often cited as a central advantage of complex models trained on protein sequence data.

The main limitation of the mean dimension presented here is that it is defined via a decomposition of the variance over the whole sequence space. The average dimension of important interactions when assigning the same weight to all possible sequences might, however, be very different from the average dimension of important interactions for example on the training set. A model with a low mean dimension could for example still include subtle higher-order interactions that are important in a specific region in sequence space. This is closely connected to the question of different gauge representations, which makes the concept of interactions itself ambiguous, since their strength depends on the representation used. Fixing a gauge representation, as we did in this work, removes this freedom, but other

choices for the representation are possible and will lead, in general, to different results. A possible avenue could be for example to ask for the representation that leads to the least complex expression on the training set, and then measure the mean dimension in this representation. We intend to explore these questions in further research.

While we used the mean dimension in this work for model comparison and analysis, it would be interesting to probe its potential as an element of the model selection or training procedure itself, for example as an indicator for early stopping or during hyperparameter optimization. In addition, it is possible that the mean dimension could reveal further insights into other phenomena observed in neural networks, for example the double descent phenomenon.

References

[1] Adam J Riesselman, John B Ingraham, and Debora S Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature methods*, 15(10):816–822, 2018.

[2] Christoph Feinauer and Martin Weigt. Context-aware prediction of pathogenicity of missense mutations involved in human disease. *arXiv preprint arXiv:1701.07246*, 2017.

[3] Thomas A Hopf, John B Ingraham, Frank J Poelwijk, Charlotta PI Schärfe, Michael Springer, Chris Sander, and Debora S Marks. Mutation effects predicted from sequence co-variation. *Nature biotechnology*, 35(2):128–135, 2017.

[4] Jonathan Frazer, Pascal Notin, Mafalda Dias, Aidan Gomez, Joseph K Min, Kelly Brock, Yarin Gal, and Debora S Marks. Disease variant prediction with deep generative models of evolutionary data. *Nature*, 599(7883):91–95, 2021.

[5] William P Russ, Matteo Figliuzzi, Christian Stocker, Pierre Barrat-Charlaix, Michael Socolich, Peter Kast, Donald Hilvert, Remi Monasson, Simona Cocco, Martin Weigt, et al. An evolution-based model for designing chorismate mutase enzymes. *Science*, 369(6502):440–445, 2020.

[6] Faruck Morcos, Andrea Pagnani, Bryan Lunt, Arianna Bertolino, Debora S Marks, Chris Sander, Riccardo Zecchina, José N Onuchic, Terence Hwa, and Martin Weigt. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proceedings of the National Academy of Sciences*, 108(49):E1293–E1301, 2011.

[7] Thomas A Hopf, Charlotta PI Schärfe, João PGLM Rodrigues, Anna G Green, Oliver Kohlbacher, Chris Sander, Alexandre MJJ Bonvin, and Debora S Marks. Sequence co-evolution gives 3d contacts and structures of protein complexes. *elife*, 3:e03430, 2014.

[8] Christoph Feinauer, Hendrik Szurmant, Martin Weigt, and Andrea Pagnani. Inter-protein sequence co-evolution predicts known physical interactions in bacterial ribosomes and the trp operon. *PloS one*, 11(2):e0149166, 2016.

[9] Simona Cocco, Christoph Feinauer, Matteo Figliuzzi, Rémi Monasson, and Martin Weigt. Inverse statistical physics of protein sequences: a key issues review. *Reports on Progress in Physics*, 81(3):032601, 2018.

[10] Debora S Marks, Lucy J Colwell, Robert Sheridan, Thomas A Hopf, Andrea Pagnani, Riccardo Zecchina, and Chris Sander. Protein 3d structure computed from evolutionary sequence variation. *PloS one*, 6(12):e28766, 2011.

[11] Magnus Ekeberg, Cecilia Lövkvist, Yueheng Lan, Martin Weigt, and Erik Aurell. Improved contact prediction in proteins: using pseudolikelihoods to infer potts models. *Physical Review E*, 87(1):012707, 2013.

[12] Christoph Feinauer, Marcin J Skwark, Andrea Pagnani, and Erik Aurell. Improving contact prediction along three dimensions. *PLoS computational biology*, 10(10):e1003847, 2014.

[13] Xinqiang Ding, Zhengting Zou, and Charles L Brooks III. Deciphering protein evolution and fitness landscapes with latent space models. *Nature communications*, 10(1):1–13, 2019.

[14] Donatas Repecka, Vykintas Jauniskis, Laurynas Karpus, Elzbieta Rembeza, Irmantas Rokaitis, Jan Zrimec, Simona Poviloniene, Audrius Laurynenas, Sandra Viknander, Wissam Abuajwa, et al. Expanding functional protein sequence spaces using generative adversarial networks. *Nature Machine Intelligence*, 3(4):324–333, 2021.

[15] Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R Eguchi, Po-Ssu Huang, and Richard Socher. Progen: Language modeling for protein generation. *arXiv preprint arXiv:2004.03497*, 2020.

[16] Dylan Marshall, Haobo Wang, Michael Stiffler, Justas Dauparas, Peter Koo, and Sergey Ovchinnikov. The structure-fitness landscape of pairwise relations in generative sequence models. *BioRxiv*, 2020.

[17] Christoph Feinauer, Barthelemy Meynard-Piganeau, and Carlo Lucibello. Interpretable pairwise distillations for generative protein sequence models. *PLOS Computational Biology*, 18(6):e1010219, 2022.

[18] Francisco McGee, Sandro Hauri, Quentin Novinger, Slobodan Vucetic, Ronald M Levy, Vincenzo Carnevale, and Allan Haldane. The generative capacity of probabilistic protein sequence models. *Nature communications*, 12(1):1–14, 2021.

[19] Ruixue Liu and Art B Owen. Estimating mean dimensionality of analysis of variance decompositions. *Journal of the American Statistical Association*, 101(474):712–721, 2006.

[20] Christopher Hoyt and Art B Owen. Efficient estimation of the anova mean dimension, with an application to neural net classification. *SIAM/ASA Journal on Uncertainty Quantification*, 9(2):708–730, 2021.

[21] Roman Hahn, Christoph Feinauer, and Emanuele Borgonovo. The mean dimension of neural networks–what causes the interaction effects? *arXiv preprint arXiv:2207.04890*, 2022.

[22] Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

[23] Jeanne Trinquier, Guido Uguzzoni, Andrea Pagnani, Francesco Zamponi, and Martin Weigt. Efficient generative modeling of protein sequences using simple autoregressive models. *Nature communications*, 12(1):1–11, 2021.

[24] Chen-Yi Gao, Fabio Cecconi, Angelo Vulpiani, Hai-Jun Zhou, and Erik Aurell. Dca for genome-wide epistasis analysis: the statistical genetics perspective. *Physical biology*, 16(2):026002, 2019.

[25] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[26] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[27] Bradley Efron and Charles Stein. The jackknife estimate of variance. *The Annals of Statistics*, pages 586–596, 1981.

[28] Il'ya Meerovich Sobol'. On sensitivity estimation for nonlinear mathematical models. *Matematicheskoe modelirovanie*, 2(1):112–118, 1990.

[29] Herschel Rabitz and Ömer F Aliş. General foundations of high-dimensional model representations. *Journal of Mathematical Chemistry*, 25(2):197–233, 1999.

[30] Stefano Zamuner and Paolo De Los Rios. Interpretable neural networks based classifiers for categorical inputs. *arXiv preprint arXiv:2102.03202*, 2021.

A Appendix

A.1 Dataset and Mean Dimension

Table A1: **MSA Properties and Mean Dimension** The table gives an overview over the MSAs used, their properties and the estimated mean dimension. The columns are: ***MSA***: MSA filename in the supplementary data in [1]; $\boldsymbol{N}$: Sequence length (only consensus columns) ; T: Number of sequences in the MSA; $\boldsymbol{M_{Unique}}$: Number of unique sequences in the MSA; $\boldsymbol{D_{\{ArDCA,VAE\}}}$: Mean dimension estimated for ArDCA and VAE models; $\boldsymbol{\sigma_{\{ArDCA,VAE\}}}$ Standard deviation of mean dimension estimates using 1000 bootstrap samples.

MSA	N	T	T_{Unique}	D_{ArDCA}	σ_{ArDCA}	D_{VAE}	σ_{VAE}
BRCA1_HUMAN_1_b0.5	75	39396	21641	1.02	0.009	3.654	0.025
GAL4_YEAST_1_b0.6	62	22985	15833	1.016	0.009	3.193	0.024
UBC9_HUMAN_1_b0.5	138	32486	25188	1.025	0.004	5.328	0.03
SUMO1_HUMAN_1_b0.35	76	21695	8725	1.006	0.006	2.874	0.02
AMIE_PSEAE_1_b0.3	247	76372	71302	1.029	0.003	-	-
B3VI55_LIPSTSTABLE_1_b0.5	364	12925	12633	1.052	0.003	-	-
B3VI55_LIPST_1_b0.5	364	12925	12632	1.046	0.003	-	-
BF520_env_1_b0.5	657	73441	67379	1.05	0.003	-	-
BG505_env_1_b0.5	653	73877	67665	1.044	0.003	-	-
BG_STRSQ_1_b0.5	441	49471	46830	1.054	0.003	-	-
BLAT_ECOLX_1_b0.5	252	14783	13236	1.027	0.003	-	-
BLAT_ECOLX_hmmerbit[..]	253	8403	7620	1.04	0.003	-	-
BRCA1_HUMAN_BRCT_1_b0.3	186	8391	5341	1.043	0.004	-	-
CALM1_HUMAN_1_b0.5	139	36224	28496	1.016	0.004	-	-
DLG4_RAT_2_b0.45	82	210888	44258	1.022	0.006	-	-
DLG4_RAT_hmmerbit[..]	84	102410	24796	1.026	0.006	-	-
DYR_ECOLI_1_b0.5	155	21261	20290	1.017	0.004	-	-
F7YBW7_MESOW_1_b0.4	89	21281	20462	1.012	0.004	-	-
FYN_HUMAN_1_b0.8	53	36578	11125	1.013	0.008	-	-
HG_FLU_1_b0.5	544	51012	48954	1.009	0.003	-	-
HIS7_YEAST_1_b0.5	192	17441	15469	1.02	0.004	-	-
HSP82_YEAST_1_b0.5	218	23447	18119	1.024	0.004	-	-
IF1_ECOLI_1_b0.5	69	9190	8161	1.001	0.006	-	-
KKA2_KLEPN_1_b0.3	226	29808	27113	1.024	0.003	-	-
MK01_HUMAN_1_b0.6	288	65626	44808	1.03	0.004	-	-
MTH3_HAEAESTABILIZED_1_b0.5	318	26513	24103	1.04	0.003	-	-
P84126_THETH_1_b0.5	236	23742	21623	1.023	0.003	-	-
PABP_YEAST_1_b0.5	79	246405	106653	1.009	0.006	-	-
PABP_YEAST_hmmerbit[..]	82	152041	70780	1.021	0.005	-	-
PA_FLU_1_b0.5	716	19611	19472	1.016	0.004	-	-
POLG_HCVJF_1_b0.5	113	11423	6744	1.011	0.005	-	-
POL_HV1N5-CA_1_b0.5	231	40127	30489	1.017	0.004	-	-
PTEN_HUMAN_1_b0.3	304	8566	4585	1.05	0.003	-	-
PYP_HALHA_1_b0.25	89	277327	201031	1.013	0.005	-	-
RASH_HUMAN_1_b0.45	164	84762	65467	1.018	0.004	-	-
RL401_YEAST_1_b0.5	71	33026	14705	1.006	0.006	-	-
TPK1_HUMAN_1_b0.25	201	9966	9494	1.025	0.004	-	-
TPMT_HUMAN_1_b0.35	177	6688	6291	1.026	0.004	-	-
TRPC_SULSO_1_b0.5	237	23743	21550	1.02	0.003	-	-
TRPC_THEMA_1_b0.5	239	23745	21680	1.026	0.003	-	-
TRY2_RAT_1_b0.5	216	85514	68899	1.02	0.004	-	-
UBE4B_MOUSE_1_b0.45	75	16478	10249	1.027	0.006	-	-
YAP1_HUMAN_1_b0.5	30	86353	17956	0.995	0.012	-	-
parEparD_3	163	2740	2737	1.032	0.004	-	-

A.2 Review of Mean Dimension

The mean (or effective) dimension of a function $g : \mathcal{X} \to \mathbb{R}$, denoted with D_g, is typically defined as the average size of interactions present in a multivariate input-output mapping (we reserve the notation f for the special case of functions defined on categorical inputs and use D_f for their mean dimension). Intuitively, consider an function mapping a d-dimensional input $X = (X_1, X_2, ..., X_d) \in \mathcal{R}^d$ to a real value $Y \in \mathcal{R}$, where we consider the inputs following some cumulative distribution F_X. In machine learning applications, this could be for example the log probability of a class in a neural network trained for image classification. Works such as [27, 28, 29] show that it is possible to expand the input-output mapping in the analogue of a multi-body expansion as the sum of 2^d component functions, of which d are univariate component functions, each depending only on a single X_i, $\binom{d}{2}$ are bivariate component functions, each depending only on a single pair (X_i, X_j), and so on. If the expansion contains only univariate components, then the mapping is additive (it is the sum of univariate functions in each variable) and the mean dimension is equal to 1. Conversely, a value of the mean dimension greater than 1 indicates that interactions are present. Given that the expansion parameters are defined using only the inputs and outputs, without further knowledge about how one is calculated from the other, the mean dimension is an ideal tool for exploring the complexity (or richness) of black-box input-output mappings. Recently, renewed interest in the use of the mean dimension in machine learning was generated [20, 21]. The key idea of these works is to use the mean dimension as a tool for comparing different neural network architectures. For instance, [21] consider fitting several well known convolutional neural network architectures of different sizes to the same image dataset. Their experiments show that the mean dimension assumes values much smaller than the overall problem dimensionality d, and that it correlates with properties like the depth of the neural networks. The same finding is confirmed in the experiments of [20].

From the technical side, the works of [20] and [21] are able to avoid a technical complication: Following directly the definition of mean dimension, its computation would be infeasible in most practical applications since it contains an exponential number of terms. However, [20] and [21] show that, assuming feature independence, it is possible to compute the mean dimension using the expression

$$D_g = \frac{\sum_{i=1}^{d} \mathbb{E}[\Phi_i^2(X_i)]}{2V_g}, \tag{A1}$$

where V_g is an estimate of the variance of the target Y if the inputs follow the distribution F_X, and, for each feature, and the random variables $\Phi_i^2(X_i)$ are defined as $\Phi_i(X_i) = g(X_i, X_{-i}^0) - g(X^0)$, where X^0 follows the distribution F_X and (X_i, X_{-i}^0) represents X^0 with X_i^0 exchanged with a newly sampled X_i. We derive a very similar expression for our specific case in the methods section of the main paper.

Note that the mean dimension introduced in this section depends on the function under consideration and the distribution of inputs. The case where features are not independent under this distribution presents a major challenge for the calculation of the mean dimension, and the identity in Eq. A1 does not hold anymore. The result of replacing the distributions with, e.g., uniform distributions is hard to interpret in general and heuristic approaches have to be introduced, for example the introduction of de-correlating PCA layers into analysed neural networks [21].

Fortunately, for protein sequences the situation is different: As we show in the methods section of the main paper, basing our estimates on the uniform distribution over amino acids in the sequences leads to an estimate for the mean dimension that can be interpreted as the mean dimension defined via the parameters in a specific *gauge representation* of the log probability, where different gauge representations refer to different way of fixing degrees of freedom in the expansion of the log probability. The gauge representation corresponding to our values of the mean dimension is the so-called *zero-sum* or *Ising* gauge, which is widely in use in the field [6, 11, 17].

A.3 Zero-Sum Gauge

In the following we show that we can represent every real-valued function defined over fixed-sized amino acid sequences in the zero-sum gauge. This proof is already outlined in [17], and also mentioned in [30], but we present a more detailed version here since it is central to the present work. Note that some quantities are defined differently from there, which we will point out below.

Formally, we assume $f : \mathcal{Q}^N \to \mathcal{R}$ to be a function which maps a sequence of N categorical variables $(s_i)_{i=1}^N$ to a real number, where the variables take values from the finite set $\mathcal{Q}$ with $q := |Q|$.

We then show that we can write the function as

$$f(s) = \sum_{J \subseteq \mathcal{N}} f_J(s_J), \qquad \text{(4 in Main Text)}$$

where $\mathcal{N}$ is the set of all positions and the $f_J(s_J)$, which are functions depending only on the subset of positions $J \subseteq \mathcal{N}$, satisfy the conditions

$$\sum_{s_i} f_J(s_J) = 0 \text{ if } i \in J,$$

defining the so-called *zero-sum* gauge, which is also sometimes called the *Ising* gauge.

We first notice that we can expand any function f trivially by defining

$$f_{\mathcal{N}}(s_{\mathcal{N}}) := f(s),$$

with $f_J(s_J) := 0$ for $|J| < N$. This representation corresponds to assigning the function values of f on all q^N sequences to the q^N interactions of order N and setting all other interactions to 0. This shows that some representation of the form in Eq. 4 can always be found.

In the following we do not assume $f_J(s_J) = 0$ for $|J| < N$ but that we are given some representation with interactions at arbitrary orders.

Given this representation, we define transformed interactions $\hat{f}_{\mathcal{N}}(s_{\mathcal{N}})$ of order N as

$$\hat{f}_{\mathcal{N}}(s_{\mathcal{N}}) := \sum_{J \subseteq \mathcal{N}} (-1)^{N-|J|} f^J_{\mathcal{N}}(s_J), \qquad \text{(A2)}$$

where we define

$$f^J_{\mathcal{N}}(s_J) := \mathbb{E}_{s_{\mathcal{N}/J} \sim \mathcal{U}} [f(s)]$$

as the function $f(s)$ after averaging out $s_{N/J}$, with $s_{N/J}$ following the uniform distribution $\mathcal{U}$. This leaves a function that depends on the positions in J. Note that $f^{\mathcal{N}}_{\mathcal{N}}(s_{\mathcal{N}}) = f_{\mathcal{N}}(s_{\mathcal{N}})$. We also point out that the notation here is different from [17], where $f^J_{\mathcal{N}}$ is defined as a function depending on all positions *except* the ones in J. For the current purpose, the notation used here is clearer, however.

We now define transformed interactions for lower orders than N as

$$\hat{f}_J(s_J) := f_J(s_J) - (-1)^{N-|J|} f^J_{\mathcal{N}}(s_J)$$

for all $J \subset \mathcal{N}$.

The value of the function $\hat{f}$ for a given sequence s, defined by the transformed interactions, is then

$$\begin{aligned}\hat{f}(s) &= \sum_{J \subseteq \mathcal{N}} \hat{f}_J(s_J) = \sum_{J \subset \mathcal{N}} \left(f_J(s_J) - (-1)^{N-|J|} f^J_{\mathcal{N}}(s_J) \right) + \sum_{J \subseteq \mathcal{N}} (-1)^{N-|J|} f^J_{\mathcal{N}}(s_J) \\ &= \sum_{J \subset \mathcal{N}} f(s_J) + f^{\mathcal{N}}_{\mathcal{N}}(s_J) = \sum_{J \subset \mathcal{N}} f(s_J) + f_{\mathcal{N}}(s_J) = \sum_{J \subseteq \mathcal{N}} f(s_J) = f(s),\end{aligned}$$

so the transformation leaves the function value invariant.

Furthermore, if we sum the transformed interactions of order N over any dependency s_i, we get

$$\begin{aligned}\sum_{s_i} \hat{f}_{\mathcal{N}}(s_{\mathcal{N}}) &:= \sum_{J\subseteq\mathcal{N}} (-1)^{N-|J|} \sum_{s_i} f^J_{\mathcal{N}}(s_J) \\ &= \sum_{J\subseteq\mathcal{N}:i\in J} (-1)^{N-|J|} \sum_{s_i} f^J_{\mathcal{N}}(s_J) + \sum_{J\subseteq\mathcal{N}:i\neq J} (-1)^{N-|J|} \sum_{s_i} f^J_{\mathcal{N}}(s_J) \\ &= q \sum_{J\subseteq\mathcal{N}:i\in J} (-1)^{N-|J|} \mathbb{E}_{s_i\sim\mathcal{U}}\left[f^J_{\mathcal{N}}(s_J)\right] + q \sum_{J\subseteq\mathcal{N}:i\neq J} (-1)^{N-|J|} f^J_{\mathcal{N}}(s_J),\end{aligned}$$

where we separated the sum over J into terms that depend on s_i and terms that do not depend on s_i in the first line and replaced the sum over s_i with an expectation over s_i following the uniform distribution $\mathcal{U}$ in the second line. From the definition in Eq. A2 it can be seen that the term $\mathbb{E}_{s_i\sim\mathcal{U}}\left[f^J_{\mathcal{N}}(s_J)\right]$ is identical to $f^{J/i}_{\mathcal{N}}(s_{J/i})$, since we just add another position to the expectation.

The sum in the first term can therefore be rewritten as

$$\sum_{J\subseteq\mathcal{N}:i\in J} (-1)^{N-|J|} \mathbb{E}_{s_i\sim\mathcal{U}}\left[f^J_{\mathcal{N}}(s_J)\right] = \sum_{J\subseteq\mathcal{N}:i\in J} (-1)^{N-|J|} f^{J/i}_{\mathcal{N}}(s_{J/i}) = \sum_{J\subseteq\mathcal{N}:i\notin J} (-1)^{N-|J|+1} f^J_{\mathcal{N}}(s_J),$$

which leads finally to

$$\sum_{s_i} \hat{f}_{\mathcal{N}}(s_{\mathcal{N}}) = q \sum_{J\subseteq\mathcal{N}:i\notin J} (-1)^{N-|J|+1} f^J_{\mathcal{N}}(s_J) + q \sum_{J\subseteq\mathcal{N}:i\neq J} (-1)^{N-|J|} f^J_{\mathcal{N}}(s_J) = 0,$$

which means that the interactions at order N in the transformed representation satisfy the zero-sum conditions outlined in the main paper.

Fixing the interactions at order N, the problem is reduced to transforming a model with interactions up to order $N-1$ to the zero-sum gauge. This, however, can be achieved by the same procedure starting at order $N-1$. We note that it is not trivial to give an expression for the final parameters at all orders, but following the procedure, starting at order N and continuing until order 1, will certainly transform the original representation into the zero-sum gauge.