UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE

MSc in Bioinformatics

Predictive Protein Property Modeling through Gaussian Processes and Encoding Methods

Exploring DES model's kernel and latent information for Protein Design

Richard Michael

Supervised by Prof. Wouter Boomsma, Dr. Simon Bartels, Pengfei Tian Ph.D.

May 2021

Richard Michael
Predictive Protein Property Modeling through Gaussian Processes and Encoding Methods
MSc in Bioinformatics, May 2021
Supervisors: Prof. Wouter Boomsma, Dr. Simon Bartels, Pengfei Tian Ph.D.

University of Copenhagen
Faculty of Science
Masters Degree in Bioinformatics
Bülowsvej 17
1870 Frederiksberg C

Abstract

Efficient protein design relies on properties of protein variants and plays a major role in the development of bio-technical solutions or pharmacological research. The protein-sequence space is vast, and experimental assessments can be expensive and timely. Therefore, efficient computational methods are required to guide experimental research and derive additional insights in protein design. In this work we integrate two state of the art approaches: the Gaussian Process (GP) based mGPfusion method and the sequence-alignment (VAE) based Deep Sequence method [38, 68]. We propose two novel approaches to incorporate the signal of a protein family latent representation into a GP regression: (1) by adding the log-ratio of the evidence lower bound as in-silico data and (2) by computing a covariance function from the VAE's latent representation. We show that mGPfusion is applicable to a range of protein property predictions, apart from thermodynamic stability, while accounting for noise in the data and epistemic uncertainties. We propose a covariance function that uses the information from learned representations, which lets us derive a VAE-protein-specific substitution matrix. The results of this work suggest: that (1) adding a transformed log-likelihood ratio, can improve property predictions, (2) a numerically stable covariance function, can compute log-likelihoods for protein sequences with respect to individual residues from the underlying data and representation. Finally, we evaluate experimentally, how the signal from a latent representations affects GP model predictions.

Contents

List of Figures

List of Algorithms

Abbreviations

CV Cross-validation. 40

ELBO Evidence Lower Bound. 11

EM Expectation Maximization. 12

GPs Gaussian Processes. 6

KLD Kullbach-Leibler-Divergence. 12

MCMC Markov Chain Monte-Carlo. 34

MKL Multiple Kernel Learning. 9

MSA Multiple Sequence Alignment. 10

MSE Mean-Squared Error. 43

NN Neural-Network. 11

NUTS No U-Turn Sampler. 34

SGD Stochastic Gradient Descent. 13

VAE Variational Auto-Encoder. 10

WT Wild Type. 4

Nomenclature

S Substitution Matrix. 5

$\Delta\Delta G$ Δ of change in Gibbs-free energy for protein unfolding. 4

p.s.d. positive semi-definite. 24

i.i.d. independent and identically distributed. 7

1 Introduction

Proteins serve a multitude of biological functions and are relevant to the development of bio-technological solutions, industrial enzymes or in pharmacological research [48]. Determining thermodynamic stability and functional properties of proteins using computational methods is an unsolved challenge. Generally, the function of a protein is defined by its fold, and three-dimensional shape, which is largely determined by the underlying amino-acid sequence [5, ch. 3]. At its most basic, a protein is made up of 20 common amino-acid monomers and contains functional groups. This gives rise to a highly variable range of functional and structural properties. Some structures are rigid, some are highly flexible, some are non-polar and bind to membranes and some bind to each other in dimers, trimers and beyond. The proteins and their functions are selected for in the function that they serve in the respective environment of the organism. Variations in their sequence can affect structure and functional properties of the protein [91, 2].

An abundance of methods to determine a protein's structure, fold and function, have been developed. These include experimental methods, such as *mutagenesis* studies [37, 35, 3], and computational means, such as molecular dynamics simulations [49, 61]. The difficulty of computational methods arises from the fact that protein space is a vast problem domain given the sheer amount of possible combinations of amino-acids, their folds and resulting arrangements [33]. Therefore we have to rely on computational models in order to limit protein candidates. Specifically, the development of efficient computational methods has been intently researched over the previous years with the aim of guiding experimental research [11, 26, 62, 15].

Previous research shows that computational predictions can capture a general predictive trend but fail to make explicit predictions [41]. Current state of the art methods have made significant advances in the prediction of protein properties from sequence variants. We can distinguish the current state of research in the field of protein predictive models broadly into two approaches: unsupervised learning, e.g. the *Deep Sequence* model [68] and supervised learning e.g. the *mGPfusion* method [38].

This work investigates the two state of the art computational methods and provides an overview and background in sections 2.5 and 2.4.3.

Predictions by the Deep Sequence model are among the current state of the art [50], yet they: (a) lack direct applicability in protein design because the model outputs constitute a predictive trend with respect to the latent representation rather than a protein property value, (b) they do not account for the protein's structure explicitly or any experimental observations. Lastly, (c) predictions from this method lack certainty

estimates. On the other hand, the mGPfusion method, is an example of a probabilistic method that provides uncertainty estimates with its predictions [38, 90]. Drawbacks to this method are: (d) in its original form it has been shown to perform only on thermodynamic stability, (e) the method does not utilize information from the protein's family, and previous investigations suggest that (f) multiple kernel learning in this context, may be sub-optimal (see section 2.4.3 and table 7.2).
This work addresses these issues by, (a) building a transformation function conditioned on experimental observations to transform Deep Sequence predictions into in-silico values in the range of protein properties of interest. We incorporate the in-silico data, suggested to capture protein-family information (e) into the mGPfusion workflow which (b) does take into account protein structure information. The predictions made from the Gaussian Process regression (see Section 2.4) do provide (c) uncertainty estimates. The experimental case-studies that we conduct, show that (d) the mGPfusion method is applicable to more than thermodynamic stability, like e.g. growth factors. Lastly, we propose a novel kernel function (f) to quantify relatedness of mutational variants through the latent representation of a Variational Auto-Encoder (see Section 2.6) and verify it experimentally 4.4.

1.1 Research Questions

The aim of this thesis is, to integrate the Deep Sequence approach into mGPfusion and make improvements upon either model. This thesis will address the following questions:

1. Is it possible to outperform a VAE architecture to predict protein properties by combining experimental observation, structure information and the latent representation through Gaussian Processes regression?

2. Is it possible to outperform a state of the art GP-based method by incorporating information based on the VAEs latent representation, trained on Multiple Sequence Alignments?

1.2 Research Hypotheses

With reference to the reported results in [68, 38] my initial hypotheses are:

1. that adding in-silico information derived from a VAE improves performance compared to the mGP workflow,

2. that deriving a covariance function from a latent representation improves performances compared to kernels from substitution matrices,

3. combining experimental observations with VAE derived predictions through Gaussian Process regression increases correlations of predictions with experimental observations.

The methods we propose in the following sections are verified through an experimental case-study. We evaluate the experimental predictions against true experimental measurements through different metrics, namely the correlations between predictions against the true experimental measurements ρ, spearman-r, as well as mean-squared error and its square-root (see supplementary 7.1). Our goal is to find predictions that correlate highly with the true underlying data, while keeping the error low. The case-study is concluded with experimental cross-validation runs per protein, taking into account the performance over individual positions (see Section 4.4).

2 Background

This section reviews the background required for the following methodological work. We introduce metrics and measures by which we can assess the performance for protein design (2.1) as well as methods to quantify relatedness of protein mutational variants (2.2). To understand the previous work (2.3) and build up the methodological proposal, we require an understanding and definitions for Gaussian Processes (2.4) as well as Variational Auto-Encoders (2.6).

2.1 Protein Performance Metrics

We can assess a protein by a variety of properties. These include, for example *thermodynamic stability*, expressed as $\Delta\Delta G$ - the change in Gibbs Free Energy [74][24, p. 895]. This is an essential metric in protein design and a majority of computational models aim to predict thermodynamic stabilities of proteins [66, 57, 70, 12, 17]; including the evaluation of the reference mGPfusion method [38]. Another metric of interest can be the *growth factor* of an organism associated with a functional protein [7]. Previous studies have shown that there is a correlation between the observed frequency of an amino acid residue and the change in Gibbs free energy ($\Delta\Delta G$) [59, 87]. The work by Hopf et al and Riesselman et al predict the performance of differing protein growth factors [32, 68].
The focus of this thesis is to investigate protein properties with respect to *stability* and *organismal fitness*. We extend the mGPfusion model, which has been developed for predictions of thermodynamic stability to the application of growth factors.

2.2 Substitution Matrices

Now that we have established what properties we want to predict, we introduce the measure to assess two sequences and score similarity, provided we have a Wild Type (WT) and variants.
Given that we have two sequences X and X' of length n and m, we e.g. want to quantify their differences, align them, etc. . One way to achieve this is to sum the logged likelihoods of the substitutions of the different elements in the sequence with reference to a common ancestor. For this we use a substitution matrix.
Let us assume that residues in a sequence occur independently and at random with a

frequency q. We denote this for the elements of the sequences, $x \in X$ and $x' \in X'$ [1], as:

$$P(x, x' | \text{ random}) = \prod_{i=1}^{n} q(x_i) \prod_{j=1}^{m} q(x'_j). \tag{2.1}$$

Let us also assume that there is a joint probability for the residues in the sequences to occur together. We refer to this as the *match*-model:

$$P(x, x' | \text{ match}) = \prod_{i=1}^{n} p(x_i, x'_i), \tag{2.2}$$

which has the ratio

$$\frac{P(x, x' | \text{ match})}{P(x, x' | \text{ random})} = \prod_{i=1}^{n} \frac{p(x_i, x'_i)}{q(x_i) q(x'_i)}. \tag{2.3}$$

We want to arrive at an additive scoring system, therefore we introduce the log:

$$\log \left(\frac{p(x, x')}{q(x) q(x')} \right) = s(x, x') \tag{2.4}$$

(for Eq.(2.1) to Eq.(2.5) see [19, pp.14-15]). This is our score $s(x, x')$ which we can apply to the range of letters in our alphabet, which make up our sequence. From this we compute our S:

$$S = \sum_{i=1}^{n} s(x_i, x'_i). \tag{2.5}$$

We have now introduced a method to compute a score from the ratio of the frequencies of residues for a sequence. Some examples of substitution matrices are:

1. various BLOSUM matrices [29], which score based on conserved amino-acid pair blocks from aligned protein sequences,

2. PAM or Point Accepted Mutation matrices, based on the observed mutations of closely related protein sequences [83],

3. matrices based on protein structures,

4. biochemical properties and others [65, 10, 9, 63, 54].

[1] For simplicity the two sequences X and X' are assumed to have the same length, $n = m$.

For later use of these matrices and their applicability in the context of Gaussian Processes we have to adhere to the requirement of a *positive (semi-)definite* matrix, in order to construct a covariance matrix. This means, that the matrix does not have any negative eigenvalues. Though, Jokinen et al. report that they use p.s.d. kernels, BLOSUM62 is also reported not to be p.s.d (see 7.4.1 in [51]), which is accounted for by normalization of S.

2.3 Computational Models for Protein Property Prediction

Now that we have been introduced to the properties which we want to predict and a method to score similarities between sequences, we are interested what approaches exist to predict such properties from sequence and other available information. Ultimately, the approaches to predict protein properties vary. Taking into consideration the *Computational Biology* point of view, one can build a model from *evolutionary information*. Some specific examples are the evolutionary-coupling based structure prediction EVFold [75], and EVmutation [32]. Another approach are models based on *force-fields* and *biophysical* properties.
One of the gold-standards in the industry and an example of a force-field method, which encompasses different approaches is Rosetta. The Rosetta method can be used for protein structure prediction, as well as de-novo protein assembly, for single and multiple mutations and is an often-cited benchmark [78, 77, 70, 17]. Other approaches utilize elements such as: combinatorial linear models like PoPMuSiC [15], Natural Language Processing methods like ProtTrans and ESM [21, 69]. Recent work has suggested that protein information can be captured through deep learning approaches [32, 68]. We can distinguish the methodological approaches also from a Computer Science point of view. Here we can broadly separate supervised from unsupervised learning. In our specific case, we are interested in two models: the mGPfusion method is a supervised method utilizing Gaussian Process regression, whereas the Deep Sequence method relies on unsupervised learning of a probabilistic model from multiple aligned sequences.

2.4 Gaussian Processes

One specific Machine Learning approach for predictive modeling are Gaussian Processes. Gaussian Processes (GPs) have been applied to the realm of protein fitness

arguably first by Romero et al in 2013 [71]. More recently, they were used in the *mGPfusion* method, as presented by Jokinen, et al in 2018 [38].
GPs are a non-parametric, probabilistic method in Machine Learning. We can define a GP, given a mean function m and a covariance function k [2] . A Gaussian Process is a random stochastic process, which produces a *predictive posterior* for the unknown regression function f (see [88, p.13]). We can obtain these function values, for $x, x' \in \mathcal{X}$ as :

$$m(x) = \mathbb{E}[f(x)], \tag{2.6}$$
$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))], \tag{2.7}$$
$$f(x) \sim \mathcal{GP}(m(x), k(x, x')). \tag{2.8}$$

2.4.1 Gaussian Process Regression

As we obtain function values from our GP we can use it to regress an unknown function and find a predictive posterior. Let our training data be X and our unobserved test data X_*. Then our noiseless joint distribution from which we get the resulting expected training values f and test predictions f_* is:

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{2.9}$$

(see [88, p. 16]). For simplicity we assume a *zero-mean* function.
One application case for Gaussian Processes is that it enables us to account for inherent noise in our data. Let the noise terms $\zeta_1, \zeta_2, ..., \zeta_n$ be from a Normal distribution and *i.i.d.* such that $\zeta_i \sim \mathcal{N}(0, \sigma^2)$ for $i = 1, ..., n$. This is then $\text{cov}(y) = K(X,X) + \sigma_n^2 I$:

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right). \tag{2.10}$$

From this we can derive the final predictive distribution for f_*:

$$f_*|X, y, X_* \sim \mathcal{N}(\bar{f}_*, \text{cov}(f_*)), \tag{2.11}$$
$$f_* := \mathbb{E}[f_*|X, y, X_*] = K(X_*, X)[K(X,X) + \sigma_n^2 I]^{-1} y, \tag{2.12}$$
$$\text{cov}(f_*) = K(X_*, X_*) - K(X_*, X)[K(X,X) + \sigma_n^2 I]^{-1} K(X, X_*) \tag{2.13}$$

[2] The covariance function will also be referred to as *covariance kernel* or *kernel function*.

(see [88, p. 16], [40, pp.15-18]). The Gaussian distribution also applies to our observed values $y \sim \mathcal{N}(0, K + \sigma_n^2 I)$, so that we can compute the log marginal likelihood $p(y|X)$ in closed-form as:

$$\log p(y|X) = -\frac{1}{2}\mathbf{y}^T(K + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log|K + \sigma_n^2 I| - \frac{n}{2}\log 2\pi. \tag{2.14}$$

For reference see [88, p.19].

2.4.2 Kernels and Covariance Functions

In order to quantify the relatedness of our data we require a covariance function. The properties of the covariance function influence the nature of the function f. For the covariance functions in our GP we require positive definite kernels as defined in [40, p.7]. We can differentiate between different kernel properties, such as *stationary kernels* [88, p.82] with varying degrees of differentiability, *non-stationary kernels* [88, p.90] or *non-vectorial input* kernels. Furthermore, we can build new kernels from existing positive definite kernels by:

1. sums of kernels $k(x, x') = k_1(x, x') + k_2(x, x')$,
2. product of kernels $k(x, x') = k_1(x, x') \times k_2(x, x')$,
3. product spaces with $z = \begin{pmatrix} x \\ y \end{pmatrix}$, s.t. $k(z, z') = k_1(x, x') + k_2(y, y')$ or $k_n(x, x') = k_1(x, x') \times k_2(x, x')$,
4. vertical rescaling $k(x, x') = a(x)k_1(x, x')a(x')$,
5. warping and embedding $k(x, x') = k_1(u(x), u(x'))$,
6. additions $k(x, x') = k_1(x, x') + c, \forall c \in \mathbb{R}^+$,
7. product of functions $k(x, x') = f(x) \cdot f(x')$ for any $f : \mathcal{X} \longrightarrow \mathbb{R}$

(Def. 19.2-.6 [4, p. 419] and Th.2.20 [30, p. 36]). We keep this in mind for our VAE-specific substitution matrix equivalent, specifically the sum, product and rescaling of kernels.

2.4.3 Predictive GPs for Protein Variants

We have introduced GPs, some of their properties and requirements which are applicable to our method development. A question of interest is, how this method can be applied to predict protein properties.
The baseline for our benchmark and reference to state of the art methods is mGPfusion, which uses a combination of different substitution matrices through Multiple Kernel Learning (MKL). Specifically, MKL encompasses 21 substitution matrices, using the weighted sum as a covariance function which is later optimized. It is defined as:

$$K_\phi = \sum_{m=1}^{21} w_m K_m,$$

where K_m is the covariance kernel for proteins computed from the normalized substitution matrices (see Eq. 9 [38]). Together with conditioning of in-silico mutations on experimental results we compute predictions with GP regression. Results found in [38, p. i277], suggest that a selection of certain substitution matrices outperform others. Furthermore, using mGPfusion without the quasi-newton optimization gives us comparable performance, while allowing for better run-time and memory requirements of the method. [3] Therefore we propose the following changes. Firstly, we will use the latent representation of the *encoder* from the protein family (see section 2.6) to replace the Rosetta *in-silico* input source. Secondly, we propose to replace the *MKL* with a covariance function that computes the substitution likelihood from learned latent embeddings. The previous work and Gaussian Process regression suggests, that if we can obtain a covariance function that holds implicitly the relations of the protein-space within-itself we obtain better likelihood computations and have potentially better results from the model. We rely on the assumption that a VAE captures higher order information (e.g. phylogenetic relations) as constraints in the lower-dimensional latent representation. There are different approaches to capture and express this information, which will be explained in sections 3.4, and 5.4.2.

2.5 The Deep Sequence Method

The Deep Sequence model is another Machine Learning based approach to predict performance of protein variants. It relies on the sequence input and the resulting predictions correlate with protein properties based on the latent representation of the

[3] A selection of baseline results on the original dataset is available in the supplementary table 7.2 for comparison.

model. To train the model we require a Multiple Sequence Alignment (MSA).
A MSA is a generated alignment of a set of related sequences such as protein sequences. The alignment is based on evolutionary traits and accounts for mutations, deletions or insertions of the sequences toward one-another through algorithmic scoring, taking into account substitution matrices [82]. Given protein family sequences from a MSA we fit a Variational Auto-Encoder (VAE), which we introduce in the following section 2.6. From different initialization and priors an ensemble of VAEs computes a log-likelihood ratio, which correlates with underlying protein experimental observations as presented in [68]. The VAE learns a lower-dimensional latent representations, that can give insights about the protein specific or protein-family properties. Our work does not use the Deep Sequence method directly, but relies on the reported insights to use a VAE trained on protein family data. We build a model of lower complexity and use the properties of the VAE as input for predictive models. We thereby aim to find an improvement over the use of substitution matrices to gain information of protein properties from primary structure and secondary structure.

2.6 The Variational Autoencoder

Previous work has shown that specific deep-learning approaches like a (Variational) Auto-Encoders learn lower dimensional representations of underlying distributions, given a training data-set [1]. Over the past years VAEs have been applied to set new state of the art methods such as in chemical design or drug design [23, 13]. Specifically, when constructing generative auto-encoding models on protein family data, previous studies show that protein properties can be captured, such as phylogenetic relationships [68, 18, 16]. Specifically Riesselman et al train an ensemble of VAEs in an unsupervised way and utilize them as a predictive method for different protein properties such as thermodynamic stabilities, and growth-factors [68]. This section serves to introduce the VAE and its background.
Let us first define our data of size N where x is a sequence of length L, as $x_{1...N} \equiv x_{(i)}^{N} \equiv x_1, x_2, ..., x_N = \mathcal{D}$ to be *i.i.d.*.

2.6.1 VAE Definition and Latent Approximation

The general goal is to find a model $p_\theta(x)$ that approximates the true underlying data $p^*(x)$, sufficiently well. This model is then based on a conditional latent variable z. As a general function approximator we use *Neural Networks*, also referred to as *Multi-layer Perceptrons* [55, pp. 565-566]. We define the VAE as a composition of an encoder

Neural-Network (NN) e and the decoder neural network g as proposed in [44] and [25, pp.499-523]. For a VAE the encoder learns the parameterization of the distribution underlying the latent variable z of dimensions d. Such that $z \in \mathbb{R}^d$ expresses the mean μ and standard deviation σ of a Gaussian distribution. The decoder learns to reconstruct the input from the latent representation. We have the prior distribution $p(z)$ for which we find an approximation q. We will refer to the parameters of the encoder as ϕ and the decoder θ. The goal is to find:

$$\mathcal{N}(z; \mu, \sigma) = q_\phi(z|x) \approx p_\theta(z|x). \tag{2.15}$$

For reference see [43, pp.15-16].

Instead of $p(x)$ we evaluate $p(x, z)$. This means we get the joint likelihood of the data and the latent random variable z. The marginalization of the joint probability is therefore: $p(x) = \int p(x, z)dz \Leftrightarrow \int p(x|z)p(z)dz$. With an infinite latent continuous z we would obtain a model as a mixture of Gaussians with differing means [43, p.12], such that:

$$\text{prior: } p(z) \sim \mathcal{N}_z(0, I),$$
$$\text{posterior: } p(x, z, \phi) \sim \mathcal{N}_x(e(z, \phi), \sigma^2 I).$$

Where ϕ are the parameters for our non-linear encoding function e (NN). To compute the posterior over the observed data x given the hidden z we use Bayes rule:

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}. \tag{2.16}$$

The marginal likelihood of the model can be very complex and is generally not tractable. Therefore in this equation $p(x)$ cannot be evaluated, and there exists no closed form expression.
We sample z^* from prior $p(z)$ such that we can compute μ and likelihood of $p(x|z)$ from $e(z^*, \phi)$. We compute:

$$p(x) = \int p(x, z|\phi)dz = \int p(x|z, \phi)p(z)dz = \int \mathcal{N}_x(e(z, \phi), \sigma^2 I)\mathcal{N}_z(0, I)dz. \tag{2.17}$$

Again there is no closed form solution for this integral over the latent random variable z. Therefore we approximate the model by maximizing the log-likelihood of a bound. Specifically, we optimize the Evidence Lower Bound (ELBO), which we introduce in the following section 2.6.2 with a derivation.

Given that we have a bound to optimize on we could perform Expectation Maximization (EM) [4, p.260] - iteratively choosing ϕ such that the approximation q is equal to posterior p and changing θ to maximize the bound.
The problem is that we cannot solve the posterior expression due to the non-linear latent random variable. This makes the EM approach not feasible and instead we have to rely on an approximation through $q(z|\phi)$.
We aim to find a Gaussian distribution that is closest to minimizing Kullbach-Leibler-Divergence (KLD) [4]. Minimizing the KL-divergence is equivalent to maximizing the log-likelihood of the probabilistic model [43, p. 10]. For optimal choice we condition the approximation on the observed data:

$$q(z|\theta, x) \sim \mathcal{N}_z(g_\mu(x|\theta), g_\sigma(x|\theta)). \tag{2.18}$$

Again, the $g(x, \theta)$ is a NN with parameters θ to predict mean and variance. From this we can compute the ELBO as,:

$$\text{ELBO}(\theta, \phi) = \int q(z|x, \phi) \log p(x|z, \theta) dz - KL(q(z|x, \phi), p(z)). \tag{2.19}$$

The first term can be rewritten as an expected value, from summing over samples z^* obtained from e: $\mathbb{E}[f(z)] \approx \frac{1}{N} \sum_n^N e(z_n^*)$.

$$\Rightarrow \text{ELBO}(\theta, \phi) \approx \log p(x|z^*, \theta) - KL(q(z|x, \phi), p(z)). \tag{2.20}$$

further $q(z|x, \theta) \sim \mathcal{N}_z(\mu, \sigma)$. To make things more tractable we can compute the KL divergence in closed form:

$$KL(q(z|x, \theta), p(z)) = \frac{1}{2}(\text{Tr}(\sigma) + \mu^T \mu - D - \log(\det(\sigma))). \tag{2.21}$$

Thus we compute the ELBO by (i) estimating μ and covariance σ of posterior from q and (ii) draw sample z^*.
This sampling procedure is not differentiable. The latent variable z does not allow for back-propagation and subsequently we cannot optimize with respect to our objective function. In order to get a fully differentiable network and perform optimization we have to reparameterize, such that $z^* = \mu + \sigma^{1/2}\epsilon$, with the random variable $\epsilon \sim \mathcal{N}_\epsilon(0, I)$ (see [44, p.3] and [25, pp.686-687]). Now we have the means to set-up and optimize a VAE using variational approximation through q to optimize a lower bound on the true posterior p.

[4] The KL divergence quantifies the distance between two distributions p, q and is defined as: $\text{KL}(q, p) \equiv \langle \log q(x) - \log p(x) \rangle_{q(x)} \geq 0$ [46] [4, p.170].

Further, we rely on the stochastic estimate of the optimization (e.g. SGD or ADAM [42]). Stochastic Gradient Descent (SGD) optimizes on randomly drawn mini-batches $\mathcal{M} \subset \mathcal{D}$ from our data $\mathcal{D}$, such that we get an estimator for our maximum likelihood:

$$\frac{1}{N_{\mathcal{D}}} \log p_\theta(\mathcal{D}) \simeq \frac{1}{N_{\mathcal{M}}} \sum_{x \in \mathcal{M}} \log p_\theta(x) \tag{2.22}$$

(see [43, p. 11] and SGD as presented in [56]).

2.6.2 The Evidence Lower Bound

To optimize the VAE we introduce the loss, which we get from a lower bound.:
Given our data $\{x_i\}^N$ we maximize the log likelihood with respect to θ. Under the assumption that σ^2 is known, we learn θ.

$$\hat{\theta} = \text{argmax}_\theta \sum_i^I \log p(x_i|\theta) = \text{argmax}_\theta \sum_i^I \log \int p(x_i, z_i|\theta) dz_i. \tag{2.23}$$

We find a lower bound on the log likelihood given θ dependent on the parameters ϕ.
We can use Jensen's Inequality (see Def. (2.6.3)) to derive the lower bound:

$$\Rightarrow \log \left(\int q(z) \frac{p(x, z|\theta)}{q(z)} dz \right) \geq \int q(z) \log \frac{p(x, z|\theta)}{q(z)} dz, \tag{2.24}$$

$$\Rightarrow \text{ELBO}(\theta, \phi) = \int q(z|\phi) \log \frac{p(x, z|\theta)}{q(z|\phi)} dz. \tag{2.25}$$

We maximize the lower bound with respect to θ and ϕ.
We know that the bound is tight given a fixed θ, if ϕ is such that the ELBO and the likelihood are equivalent given an observation: $q(\phi) \simeq p(z|x)$. Keeping this in mind we can express the ELBO through the conditional probability:

$$\begin{aligned}
\text{ELBO}(\theta, \phi) &= \int q(z|\phi) \log \frac{p(x, z|\theta)}{q(z|\phi)} dz && (2.26)\\
&= \int q(z|\phi) \log \frac{p(z|x, \theta) p(x|\theta)}{q(z|\phi)} dz && (2.27)\\
&= \int q(z|\phi) \log p(x|\theta) + \int q(z|\phi) \log \frac{p(z|x, \theta)}{q(z|\phi)} dz && (2.28)\\
&= \log p(x|\theta) + \int q(z|\phi) \log \frac{p(z|x, \theta)}{q(z|\phi)} dz && (2.29)\\
&= \log p(x|\theta) - KL[q(z|\phi)||p(z|x, \theta)]. && (2.30)
\end{aligned}$$

The KL divergence is zero iff $q(z|\phi) = p(z|x, \theta)$. The introduction of the ELBO was done with reference to [25, p.693] (see (2.5)-(2.12) in [43, p.18], [44, p.6]). The ELBO allows us a joint optimization of both θ and ϕ through Stochastic Gradient Descent.

2.6.3 Def.: Jensen's Inequality

We rely on Jensen's inequality to derive the ELBO. Let the expected value of a point passed through a concave function f be at least as large as the expected values of that concave function at that point,:

$$f(\mathbf{E}[y]) \geq \mathbf{E}(f(y)) \tag{2.31}$$

(see (4) in [36] [4, p.666]). We can apply this to the $\log$ function, such that in our specific case:

$$\log(\mathbf{E}[y]) \geq \mathbf{E}(\log(y)) \Rightarrow \log(\int p(y)dy) \geq \int p(y)\log(y)dy. \tag{2.32}$$

The likelihood corresponds to our expected value.

3 Methods

The previous chapter provided the basis for the mGP workflow. We use experimental data for either thermodynamic stability or growth factors depending on the proteins. A detailed overview of the data-set can be found in section 4.1. We aim to incorporate information from the protein family data through sequence information and therefore add a MSA per protein as input, from which we fit a VAE. We propose to add the log-likelihood ratio (section 3.1) and transform it using Bayesian Regression (section 3.2.3), to add it as in-silico data. This section introduces how we incorporate the log-likelihood values as well as the required transformation function, which makes up the mGP+ΔELBO workflow. As an additional, separate method, we propose to add the log-likelihoods of the VAE into the covariance kernel directly. This constitutes the mGP+DES-kernel workflow and will be described in section 3.4.

3.1 VAE-derived Likelihoods as Input

Given that we have a VAE that has learned a protein family MSA, we use the likelihoods for a given sample as in-silico input downstream. We make the assumption that the model has learned higher order functional constraints from the sequences. This approach is described by Riesselman et al as the *log-ratio heuristic* [68, p. 817]. We use specifically the change in log-probability with respect to the WT from the latent representation for each protein and derive *in-silico* values y^S.:

$$y^S = \Delta\text{ELBO} = \log p(x) - \log p(\text{WT}). \tag{3.1}$$

Previous work has shown, that this approach can work reliably across different models [32].

3.2 Hyperparameters and Model Selection

Throughout the workflow different hyperparameters are assigned and optimized: hyperparameters for the VAE are the encoder ϕ and decoder θ, the parameters for the transformation function, including noise parameters for experimental observations σ_E, in-silico data which also accounts for encoder output σ_S and their scalar t. Lastly, there

are the weights $\mathbf{w}$ associated with the kernel learning.
We define the vector of hyperparameters as $\Phi = (\phi, \theta, \sigma_E, \sigma_S, t, \mathbf{w})$. We optimize these hyperparameters separately, such that we (1) optimize hyperparameters for our VAE, then (2) we train the Bayesian Regression scaler as a separate model and (3) we perform kernel learning by optimizing the negative log-likelihood.

3.2.1 Models and Training

VAE Specifications

Three VAEs were trained with 55 latent dimensions, the encoder a feed-forward NN layer with 1700 neurons and ReLU activation using one-hot encoded sequences as input. The decoder is a feed-forward NN with 1200 neurons, as well as a dropout layer and ReLU activation. Final activation is a log-softmax on the classification output corresponding to the label-encoding of the sequence. Training was conducted over 200 epochs across mini-batches of size 128 sequences weighted samples from protein respective MSA data-set. For optimization the Adam optimizer with a learning-rate of 0.000027 was used, dropout was set to 0.065 and a weight decay of 0.007 . See supplementary table 7.3 for an overview.
The difference between the *Deep Sequence* model and the VAE architecture used in this thesis is, that we use neither sparsity layers nor linear mappings C or different prior initialization. We limit ourselves to a standard normal prior. The likelihoods which we compute come from one VAE per protein and not an ensemble of VAEs.

3.2.2 Sequence Weighting

The composition of the MSA for each protein can be biased due to the generation of the MSAs. Some sequences may be over-represented, which impacts the training of the VAEs. In order to account for that and improve training performance, sequence weighting was conducted as proposed by Riesselman et al [68]. We find a weight π of the sequence x and a variant x' via the Hamming distance D_H:

$$\pi = \left(\sum_{t}^{N} I[D_H(x, x') < \theta_{ID}] \right)^{-1} . \tag{3.2}$$

See [68, p.823]. Here θ_{ID} is the percentage of allowed divergence and in our case we use 80% sequence identity. The computed weight is then used in the sampling for

the mini-batches when conducting inference. Specifically we sample a sequence with probability $p_S = \frac{\pi_S}{\sum_t \pi_t}$ (see supplementary [68]).

3.2.3 Bayesian Regression Scaling

The original mGPfusion method utilizes Rosetta simulations. When we use Rosetta in-silico input we have *REU* values for which we have to find a transformation function to obtain e.g. $\Delta\Delta$G values. Comparable to this are the values which we obtain from sampling the latent representation. The values are in the range of the latent representation values. Therefore we have to find a transformation function from the latent representation values to experimental observations. We do this by training a *Bayesian Regression model* on the intersection of experimental variants and samples:

$$\tilde{y} = g(y^S|\theta_j) = a_j \exp(c_j y^S) + b_j y^S + d_j. \tag{3.3}$$

For reference see [38, p. i276]. We use the same hyperparameters as proposed in the mGPfusion method (see Supplementary 1 [38]). For inference we use MCMC sampling with a NUTS sampler [31], to fit the posterior:

$$p(\theta_j|y_E, y_S) \propto \prod_{i:x_i \in X_E \cap X_S} \mathcal{N}(y_i^E|g(y_i^S|\theta_j), \sigma_n^2) p(\theta_j), \tag{3.4}$$

with initial $\sigma_n^2 = 0.5$. We sample the parameters from:

$$a \sim \text{Gamma}(2, 1.5), \tag{3.5}$$

$$\frac{b}{2} \sim \text{Beta}(1.3, 2), \tag{3.6}$$

$$3.33c \sim \text{Beta}(2, 5), \tag{3.7}$$

$$d \sim \mathcal{N}(-a, \sigma_s^2). \tag{3.8}$$

See Eq. (3)-(8) [38]. Where $\sigma_S \sim \mathcal{N}(50, 0.007)$ is the *in-silico* noise. The sampling routine are $N = 10000$ MCMC samples with 500 warm-up steps. We obtain the transformation after sampling as:

$$\tilde{y}_i^S = \frac{1}{N} \sum_s^N g(y_i^S|\theta_j^{(s)}), \tag{3.9}$$

$$\sigma_T^2(i) = \frac{1}{N} \sum_s^N (g(y_i^S|\theta_j^{(s)}) - \tilde{y}_i^S)^2. \tag{3.10}$$

See [38, p.i276].

3.3 Optimizing the VAE Architecture

We formulate the configuration of hyperparameters for the VAE as an optimization problem to achieve an optimal architecture given the β-lactamase and the NZ specific data-set. The parameters are *number of neurons in hidden-layers* for encoder ϕ and decoder θ, *latent dimension* (z), *learning-rate* l, *weight-decay* d with training on an MSA over a fixed period (epochs). We run an optimization routine to minimize the objective function given our sequences $x \in X$ with the experimental observations y_i for sampled log-likelihoods $q_\theta(x|z) = y_*$:

$$\mathcal{L}_{\text{VAE}} = -|\rho(y, y_*)|. \tag{3.11}$$

.

3.4 The Making of a Kernel

We now combine the substitution matrices, with the VAE into one method. A latent model, such as the VAE, captures information from underlying data e.g. phylogenetic, evolutionary information [16]. We use this information to derive a covariance function that quantifies the similarity of two sequences, given the generative model.

3.4.1 Deriving a Substitution Matrix from an Embedding

Like the traditional substitution matrices quantify the occurrence of two amino acids together, we use the likelihood of the occurrence of the residues given the VAE. We propose:

$$S(x_i, y_i) = \log \frac{p(x_i \leftrightarrow y_i)}{p(x_i)p(y_i)}, \tag{3.12}$$

where $\leftrightarrow$ is the occurrence of the amino-acid sequence x at position i in the presence of sequence y at position i. This is normalized by $p(x)$; the encoder-likelihood of the occurrence of the sequence residues at that position. We compute the substitution score for a sequence of length L by summing over the individual residues. We cannot integrate over all of the latent space. In order to make the computations feasible, we take a sample of size n from a random normal distribution $\mathcal{N}(0, I)$, with which

we evaluate encoder and decoder likelihoods. When iterating over the sequences we compute the value for S as such:

$$\Rightarrow S(x_i, y_i) \approx \log \left(\frac{1}{p(x_{\neg i})} \frac{1}{p(y_{\neg i})} \frac{1}{n^2} \sum_{z_x \sim q(\cdot|x)}^{n} \sum_{z_y \sim q(\cdot|y)}^{n} \frac{p(x_i|z_x)p(x_{\neg i}|z_x)p(z_x)p(y_i|z_y)p(y_{\neg i}|z_y)p(z_y)}{p(x_i)p(y_i)q(z_x|x)q(z_y|y)} \right). \tag{3.13}$$

When formulating the covariance function we are interested in the value at position i, not the whole sequence. However the above formulation will make the computed values comparable to substitution matrices like BLOSUM. The probability of the elements in the sequence x not being of type i is:

$$p(x_{\neg i}) \approx \sum_{z,a} p(x_i = a, x_{\neg i}|z).$$

A derivation for S as well as $p(x_{\neg i})$ can be found in the subsequent section 3.4.2. Within this proposed computation we conduct importance sampling to weight regions of importance (see [58, ch. 9.1]). We aim to compute values from regions where we have data and likelihood assessments are sensible. *Importance sampling* introduces a *likelihood ratio*, which consists of the importance distribution from our encoder q and the nominal distribution from our decoder p [58, p. 4]. We incorporate the likelihood ratio for every latent sample that we assess.

3.4.2 The Encoder-Residue Corollary

Now that we have posed the initial formulation of the substitution matrix equivalent, we continue with its derivation:

$$S(x_i, y_i) = \log \frac{p(x_i \leftrightarrow y_i)}{p(x_i)p(y_i)}, \tag{3.14}$$

$$\Leftrightarrow S(x_i, y_i) = \log \frac{p(x_i|x_{\neg i})p(y_i|y_{\neg i})}{p(x_i)p(y_i)}. \tag{3.15}$$

We are interested in the local interaction of the residues given our learned VAE representation. Therefore we investigate the local interaction for a an element in the sequence x at position i in particular. We define the local interactions as

$$p(x_i|x_{\neg i}) = \int_{\mathcal{X}} p(x_i|z)p(z|x_{\neg i})dz. \tag{3.16}$$

The input for our encoder is discrete, even though the latent variable z is continuous [1]. We can approximate z by sampling from our encoder by inputting our sequence x; $z_x \sim q(\cdot|x)$ with n samples and we evaluate if the sequence at position i has the amino-acid value a:

$$p(X_i = a|X_{\neg i} = x_{\neg i}) \approx \frac{1}{n} \sum_{z_x \sim q(\cdot|x)}^{n} p(X_i = a|z_x) \frac{p(z_x|X_{\neg i})}{q(z_x|x)}. \tag{3.17}$$

As $p(z|x_{\neg i})$ cannot be computed, we reformulate the term by applying Bayes Rule:

$$p(X_i = a|X_{\neg i} = x_{\neg i}) \approx \frac{1}{n} \sum_{z_x \sim q(\cdot|x)}^{n} p(X_i = a|z_x) \frac{p(X_{\neg i} = x_{\neg i}|z_x)p(z_x)}{p(X_{\neg i} = x_{\neg i})q(z_x|x)}. \tag{3.18}$$

Since $p(x_{\neg i})$ is independent of z we can take it out. We keep the term to normalize with it and as normalization ensures proportionality we take out $\frac{1}{p(x_{\neg i})}$:

$$p(X_i = a|X_{\neg i} = x_{\neg i}) \approx \frac{1}{p(X_{\neg i} = x_{\neg i})} \frac{1}{n} \sum_{z_x \sim q(\cdot|x)}^{n} \frac{p(X_i = a|z_x)p(X_{\neg i} = x_{\neg i}|z_x)p(z_x)}{q(z_x|x)} \tag{3.19}$$

$$= \frac{1}{p(X_{\neg i} = x_{\neg i})} \frac{1}{n} \sum_{z_x \sim q(\cdot|x)}^{n} \frac{p(X_i = a, X_{\neg i} = x_{\neg i}|z_x)p(z_x)}{q(z_x|x)} \propto \frac{1}{n} \sum_{z_x \sim q(\cdot|x)}^{n} \frac{p(x|z_x)p(z_x)}{q(z_x|x)}. \tag{3.20}$$

Before we put everything together we take a look at the likelihood of a sequence element X_i, given that the rest of the sequence has the values in the given configuration:

$$\tilde{p}(X_i|X_{\neg i} = x_{\neg i}) \approx \frac{1}{n} \sum_{z_x \sim q(\cdot|x)}^{n} \frac{p(X_i, X_{\neg i} = x_{\neg i}|z_x)p(z_x)}{q(z_x|X = x)}. \tag{3.21}$$

Now with respect to (3.14), we obtain:

$$\Rightarrow S(x_i, y_i) \approx \log \left(\frac{1}{p(x_{\neg i})} \frac{1}{p(y_{\neg i})} \frac{1}{n^2} \sum_{z_x \sim q(\cdot|x)}^{n} \sum_{z_y \sim q(\cdot|y)}^{n} \frac{p(x_i|z_x)p(x_{\neg i}|z_x)p(z_x)p(y_i|z_y)p(y_{\neg i}|z_y)p(z_y)}{p(x_i)p(y_i)q(z_x|x)q(z_y|y)} \right) \tag{3.22}$$

$$= \log \left(\frac{1}{p(x_{\neg i})} \frac{1}{p(y_{\neg i})} \frac{1}{n^2} \sum_{z_x \sim q(\cdot|x)}^{n} \sum_{z_y \sim q(\cdot|y)}^{n} \frac{p(x|z_x)p(z_x)p(y|z_y)p(z_y)}{p(x_i)p(y_i)q(z_x|x)q(z_y|y)} \right). \tag{3.23}$$

[1] We denote z_x as the latent variable evaluated with the sequence x.

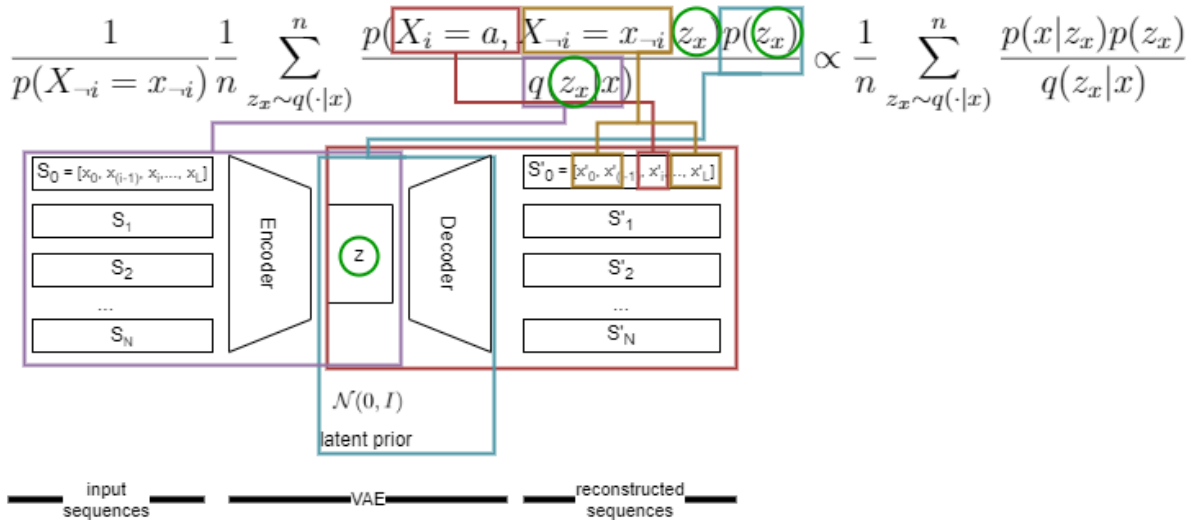


Figure 3.1: Contribution of the VAE-architecture in the proposed kernel function. We take n samples from the latent representation. The joint probability is evaluated for the element at position i with residue a (red box) such that we evaluate the categorical likelihoods output from the decoder. This goes together with the rest of the sequence $x_{\neg i}$ (orange box) being in its specific configuration, with respect to the decoder categorical likelihood output. As part of the importance sampling we evaluate the encoder with the drawn samples z, given a standard-normal prior (blue box) and the likelihood of the encoder observing the latent samples given the latent representation that is computed from the sequence input x (purple box).

3.4.3 Definition Normalizing Constant $p_{x\neg i}$

In equation (3.22) we require the normalization by the likelihood of the sequence not at position i. We introduce this via the sum over the residues, such that:

$$1 = \sum_a p(x_i = a|x_{\neg i}) \approx \sum_a \frac{1}{p(x_{\neg i})}\frac{1}{n}\sum_{z\sim q(\cdot|x)}^{n}\frac{p(x_i=a,x_{\neg i}|z)p(z)}{q(z|x)} \tag{3.24}$$

$$= \frac{1}{p(x_{\neg i})}\frac{1}{n}\sum_{z\sim q(\cdot|x)}^{n}\sum_a \frac{p(x_i=a,x_{\neg i}|z)p(z)}{q(z|x)} \tag{3.25}$$

$$\Rightarrow p(x_{\neg i}) \approx \frac{1}{n}\sum_{z,a}\frac{p(x_i=a,x_{\neg i}|z)p(z)}{q(z|x)}. \tag{3.26}$$

3.4.4 Introducing Numerical Stability

When computing the likelihoods we note that the term $p(X_{\neg i} = x_{\neg i}|z)$ becomes very small especially with longer sequences. This drives the values computed with the proposed kernel function against zero. One potential way to improve the numerical properties of our term is to take apart the normalization with $q(z|X = x)$ into its

constituents. Then we apply parts of the vector to their respective parts in the sequence. For the sake of brevity we will write $q(z|X = x)$ as q. We take apart q into its constituents as follows:

$$q = (q^{\frac{1}{L}})^L = (\exp(\log(q))^{\frac{1}{L}})^L = \left(\exp(\frac{\log(q)}{L})\right)^L = \prod_{i=1}^{L} \exp\left(\frac{\log(q)}{L}\right). \tag{3.27}$$

We introduce this to $p(x_i|x_{\neg i})$. For brevity we leave out the normalization with the latent samples n:

$$p(X_i = a|X_{\neg i} = x_{\neg i}) \propto p(X_i = a|z)\frac{p(X_{\neg i} = x_{\neg i}|z)p(z)}{\prod_{i=1}^{L} \exp(\frac{\log(q)}{L})}. \tag{3.28}$$

We apply the $\log$ [2].:

$$\Rightarrow \log\left(p(X_i = a|z)\frac{p(X_{\neg i} = x_{\neg i}|z)p(z)}{\prod_{i=1}^{L} \exp(\frac{\log(q)}{L})}\right) \tag{3.29}$$

$$= \log(p(X_i = a|z)) + \log(p(X_{\neg i} = x_{\neg i}|z)) + \log(p(z)) - \sum_{i=1}^{L} \frac{\log(q)}{L}. \tag{3.30}$$

In order to subtract our normalizing q from the likelihood at $X_{\neg i}$ we reformulate the log-likelihood with respects to every element in the sequence except for i, using j, such that $j \neq i$:

$$\log(p(X_{\neg i} = x_{\neg i}|z)) = \log \prod_{\substack{j=0 \\ j\neq i}}^{L} p(X_j = x_j|z) = \sum_{\substack{j=0 \\ j\neq i}}^{L} \log p(X_j = x_j|z). \tag{3.31}$$

Now we can subtract q from the likelihood p. We make use of the fact that $\sum_{i=0}^{L} \log(q(z|X_i = x_i)) = \sum_{\substack{j=0 \\ j\neq i}}^{L} \log(q(z|X_j = x_j)) + \log q(z|X_i = x_i)$:

$$\Rightarrow \log(p(X_i = a|z)) + \log(p(z)) + \sum_{\substack{j=0 \\ j\neq i}}^{L} \log(p(X_j = x_j|z)) - \sum_{i=1}^{L} \frac{\log(q(z|X_i = x_i))}{L} \tag{3.32}$$

$$= \log(p(X_i = a|z)) + \log(p(z)) + \sum_{\substack{j=0 \\ j\neq i}}^{L} \left(\log(p(X_j = x_j|z)) - \frac{\log(q(z|X_j = x_j))}{L}\right) \tag{3.33}$$

$$- \frac{\log q(z|X_i = x_i)}{L}.$$

[2] We simplify our equation by using the equivalence of the log of a product as $\log \prod f(x) = \sum \log f(x)$.

We do the same procedure with $p(z)$ and arrive at :

$$\Rightarrow \log(p(X_i = a|z)) + \sum_{\substack{j=0 \\ j\neq i}}^{L} \log(p(X_j = x_j|z)) - \sum_{i=1}^{L} \frac{\log(q(z|X_i = x_i)) + \log(p(z|X_i = x_i))}{L} \tag{3.34}$$

$$= \log(p(X_i = a|z)) + \sum_{\substack{j=0 \\ j\neq i}}^{L} \left(\log(p(X_j = x_j|z)) + \frac{\log(p(z|X_j = x_j)) - \log(q(z|X_j = x_j))}{L} \right) \tag{3.35}$$

$$+ \frac{\log p(z|X_i = x_i) - \log q(z|X_i = x_i)}{L}.$$

Lastly, we have to account for the length of the sequence. We therefore introduce a constant that is proportional to the length of the sequence. If we assume that we sample from uniformly distributed residues we get a likelihood of $p = \frac{1}{20^L}$ per residue in a sequence of length L. Now we introduce a constant of the nature $c = \frac{20^L}{20^L}$ into Eq. 3.24:

$$p(X_i = a|X_{\neg i} = x_{\neg i}) \approx \frac{1}{20^L} \frac{1}{p(X_{\neg i} = x_{\neg i})} \frac{1}{n} \sum_{z_x \sim q(\cdot|x)}^{n} \frac{p(X_i = a|z_x)p(X_{\neg i} = x_{\neg i}|z_x)p(z_x)20^L}{q(z_x|x)}. \tag{3.36}$$

We regard specifically the term $\log(p(X_{\neg i} = x_{\neg i}|z))$:

$$\Rightarrow \log(p(X_i = a|z)) + \sum_{\substack{j=0 \\ j\neq i}}^{L} \left(\log(p(X_j = x_j|z)) + \frac{\log(p(z|X_j = x_j)) - \log(q(z|X_j = x_j)) + \log(20^L)}{L} \right) \tag{3.37}$$

$$+ \frac{\log p(z|X_i = x_i) - \log q(z|X_i = x_i) + \log 20^L}{L}.$$

This is the final, numerically stable computation to obtain the values from the covariance function.

3.4.5 S-Matrix Normalization

The substitution values, which we compute from our proposed covariance function, can be negative, corresponding to values of low likelihood - as they are in log-space. As a consequence it is possible that our kernel function is not positive definite, does

not constitute a covariance matrix and is therefore not applicable to our GP regression. There are different ways to account for negative values and ensure a p.s.d. kernel.

1. normalize S values s in a range of $s \in [0, 1]$,

2. apply $\exp(S)$,

3. in case we have no negative values, but still no p.s.d. matrix, we can compute the eigenvalues of k and incrementally add the smallest eigenvalue, until p.s.d. is achieved.

We use the normalization, which is suggested by Jokinen et al. (see Eq. 8 [38]). Given our original values S_0 we scale by min and max:

$$S = \frac{S - \min(S_0) + 1}{\max(S_0) - \min(S_0) + 1}. \tag{3.38}$$

3.4.6 The VAE-based Covariance Function

We use the substitution value from 3.22 together with the rules of building new kernel functions (see 2.4.2) to define our covariance function [3]:

$$k(x, y) = \sum_{p=1}^{M} \left(S(x_p, y_p) \sum_{l \in \text{nbps}} S(x_l, y_l) \right). \tag{3.39}$$

We compute the value S for all mutational variants M, while incorporating the sum over the values of the neighborhood [4] residues (see top-row graph kernel in Fig. 4.1). As the computed S values can become large, even in log-space we normalize the covariance function values:

$$\hat{k}(x, y) = \frac{k(x, y)}{\sqrt{k(x, x)k(y, y)}}. \tag{3.40}$$

[3] For reference see Eq. 8 in [38].

[4] The neighborhood is defined as nbps residues in a 5Åradius.

3.4.7 The Proposed Algorithm

From our initial proposal and the subsequent considerations, we formulate the algorithm of our proposed method as follows.:

Algorithm 1: DES-KERNEL

Input: VAE, Sequence: x, Sequence: y, n samples: S=$\{s_{0...n} \sim \mathcal{N}(0,1)\}$, index $0 \leq i \leq L$

Output: Kernel value of x and y interaction at position i: $k(x_i, y_i)$

$z_x \leftarrow \texttt{VAE.encoder}(x)$
$z_y \leftarrow \texttt{VAE.encoder}(y)$
$Cat_x \leftarrow \texttt{Cat}(\texttt{VAE.}(z_x.\mu).\exp())$
$Cat_y \leftarrow \texttt{Cat}(\texttt{VAE.}(z_y.\mu).\exp())$
$p_x \leftarrow Cat_x.\texttt{logprob}(x)$
$p_y \leftarrow Cat_y.\texttt{logprob}(y)$
$\vec{p}_{x_i x_{\neg_i}} \leftarrow []$
$\vec{p}_{y_i y_{\neg_i}} \leftarrow []$
for $s \in S$ **do**
    $p_{x_i x_{\neg_i}}.\texttt{append}(\texttt{likelihood}(\texttt{VAE}, x, z_x, i, s))$
    $p_{y_i y_{\neg_i}}.\texttt{append}(\texttt{likelihood}(\texttt{VAE}, y, z_y, i, s))$
$p_{x_{\neg i}} \leftarrow \sum_a \frac{1}{n} \sum_n \vec{p}_{x_i x_{\neg_i}}$
$p_{y_{\neg i}} \leftarrow \sum_a \frac{1}{n} \sum_n \vec{p}_{y_i y_{\neg_i}}$
$\hat{p}_{x_i x_{\neg i}} \leftarrow \frac{1}{p_{x_{\neg_i}}} \frac{\frac{1}{n}\sum_n (p_{x_i x_{\neg i}})[i]}{p_x[i]}$
return $\log(\hat{p}_{x_i x_{\neg i}} \times \hat{p}_{y_i y_{\neg i}})$

Algorithm 2: LIKELIHOOD

Input: VAE, x, z_x, i, s

Output: Normalized likelihood: $\vec{p}_{x_i x_{\neg i}}$

$z \leftarrow z_\mu + s \times z_\sigma$
$p_z \leftarrow \exp(\sum_z \mathcal{N}(\vec{0}_z, \vec{1}_z).\texttt{logprob}(z))$
$q_{zx} \leftarrow \exp(\sum_z \mathcal{N}(\vec{0}_z, \vec{1}_z).\texttt{logprob}(z))$
$Cat_p \leftarrow \texttt{Cat}(\texttt{VAE.decoder}(z).\exp()).p$
$\vec{p}_{x_{\neg i} z} \leftarrow []$
for $j, s \in \texttt{numerate}(\text{sequence})$ **do**
    if $j == i$ **then** continue
    else $p_{x_{\neg_i} z}.\texttt{append}(Cat_p[j, s])$
$\vec{p}_{x_i x_{\neg i}} \leftarrow \prod \vec{p}_{x_{\neg i} z}$
$\hat{\vec{p}}_{x_i x_{\neg i}} \leftarrow \frac{(Cat_p \times \vec{p}_{x_i x_{\neg i}}) \times p_z}{q_{zx}}$
return $\hat{\vec{p}}_{x_i x_{\neg i}}$

4 Results

The workflow encompassing mGP and the proposed additions is displayed in Figure 4.1. For each protein we use the structural information[1], from which we compute the coordinates of the residues and derive contact maps in a 5Å distance around each residue of the protein. We incorporate experimental data from Protabank, as described in the following section (4.1) [84]. The *mGP+*ΔELBO includes scaled in-silico data and is described in the later section (4.3.1). The variants are scored with respect to differing residues in the sequence by either the weighted sum of covariance matrices (*mGP* or *mGP+*ΔELBO) or by the previously proposed covariance function. We predict unseen protein variant properties using GP regression and benchmark the individual workflows, the results of which can be found in section (4.4).

[1]This is derived from the associated PDB data.

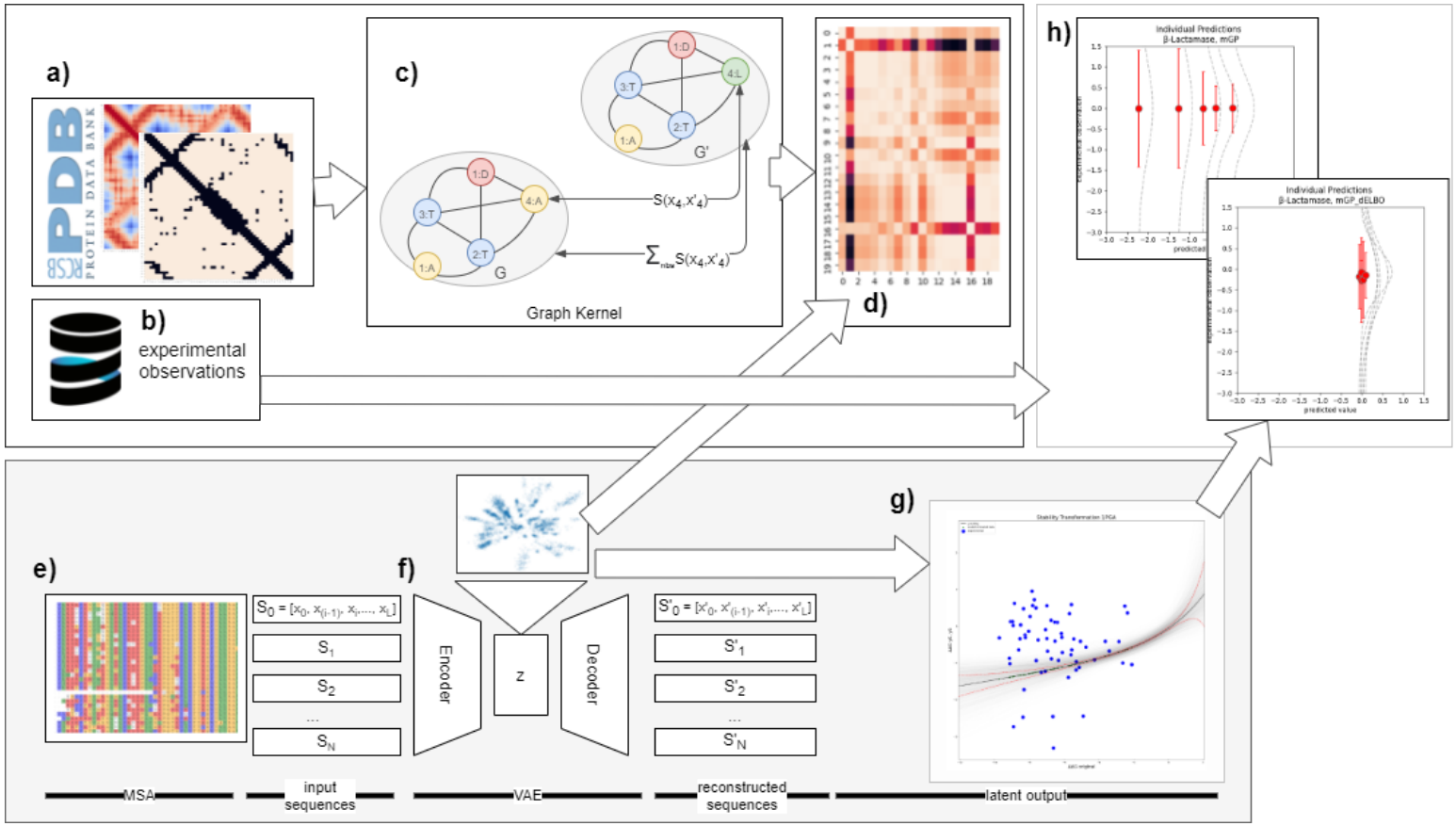


Figure 4.1: The workflow for mGP and proposed methods.

The protein experimental observations come from Protabank (b), while structure information are extracted from PDB files of the Protein Databank (a). From the structure information a contact-map for each residue with its neighbors (in a 5Å radius) is derived. A covariance kernel is computed from one or multiple substitution matrices (d), incorporating the changes in sequences variants (c, Graph Kernel) and neighboring residue information. Further, a MSA is used for individual proteins (e) and related family sequences to train a VAE (f) (bottom row). The learned latent representation is used to derive a substitution kernel (d) and compute log-likelihood ratio (ΔELBO) of sequences. The ΔELBO data is scaled using Bayesian Regression (g) conditioned on experimental observations. The covariance function is optimized, for the mGP workflow, which is the weighted sum of individual substitution matrices. The experimental observations, the optional in-silico data and optimized covariance kernel are input for a predictive Gaussian-Process regression model (h). This results in three workflows for investigation: *mGP* (a, b, c, d, h), *mGP*+ΔELBO in-silico data (a, b, c, d, e, f, g, h) and mGP using a latent representation based covariance function instead of the weighted sum over substitution matrices.

4.1 Data

We are going to evaluate the mGP and mGPfusion method empirically. To give a baseline, the mGPfusion method has been assessed on a subset of the original data - specifically 9 proteins taken from Protherm [47] as presented in [38]. The supplementary table 7.2 gives a starting point and reference values for context.
To experimentally validate the methods and proposal we are going to conduct a case study on three proteins. An overview of the relevant data-sets for the case-study on the selected proteins, can be found in Fig. 4.2. We are going to use different information available for each protein, namely: the *primary* - amino-acid sequence information, *secondary* - sequence structure, and *tertiary* - fold of protein, available from pdb information and basis for computed contact-maps. Different amounts of data are available for each protein, specifically:

1. the growth factor under ampicillin stress for β-Lactamase, (n=4788) single-site mutational variants [80],

2. the thermodynamic stability for a large number of single-site mutants (n=3143) of Protein-G [57],

3. the growth rate of yeast cultures for Ubiquitin (n=1263) [72].

An abundance of family information is available for β-Lactamase and Ubiquitin, whereas Protein-G does not have many native WT sequences and only a small set of sequences available for a MSA, see Fig. 4.2. The SSL data were taken from experimental observations, with Ubiquitin the least available single-site mutational variants and β-Lactamase the most. To put the method in a comparable context, experimental data published on β-lactamase[2] were used for comparison [80]. From this data we can compare the results with the Deep Sequence generative model [68]. In order to assess performance on a well-performing data-set which has been assessed in the mGPfusion publication and with a sufficiently large experimental basis, experiments were also run with Protein-G[3]. [4] [57]. Another protein for comparison was Ubiquitin [5], we selected this, because it showed low performance (spearman-r lower 0.5) in the Deep Sequence work in comparison to other assessed proteins. Therefore this data

[2]PDB-ID: 1FQG
[3]Please note that throughout this thesis we use *Protein-G* to refer to the *B1 Immunoglobulin binding domain of the streptococcal organism.*
[4]PDB-ID: 1PGA
[5]PDB-ID: 1UBQ

serves as a low-performance benchmark with the experimental data [72]. Lastly an internal data-set from Novozymes was used, specifically a Hexosaminidase. Since this data is confidential, intellectual property, all results of that work are contained in an undisclosed appendix.

To train the protein-family VAEs MSA from the individual proteins were used and an overview can be found in Fig. 4.2. The alignments were generated as described in the *Online Methods* section in [32] - see supplementary 7.3 for details. We denote a MSA as a matrix X with n sequences of length L, the sequences were label-encoded $x \in \mathbb{Z}^{n \times L}$ as well as one-hot encoded. β-lactamase and Ubiquitin were taken as available from the work by Hopf et al and Riesselman et al [32, 68]. The available β-lactamase MSA consisted of n=8403 sequences. Two Ubiquitin MSAs [6] were used, such that the combined MSA consisted of n=11405 sequences. The protein-G MSA had to be generated using the tools available from the MPI bioinformatics toolkit [93], since Protein-G wild-type sequences are not as abundant, the HHblits search parameters had to be relaxed. The E-val cutoff was set to 0.1, with n=5 iterations, and 10% min probability in the hitlist against UniRef30_2020_06. The resulting PGA-MSA consisted of n=1133 sequences and is of lesser quality than the other MSAs. This is not just because of the lower sequence count, but also due to the amount of gaps in the MSA and the higher chance of unrelated sequences with respect to the PGA wild-type.

Overall this data gives us a differentiated basis to conduct our experiments with different quality and amount of data, potentially impacting the mGP workflow and the proposed additions.

[6] The Ubiquitin is human UBC and ISG15 specifically.

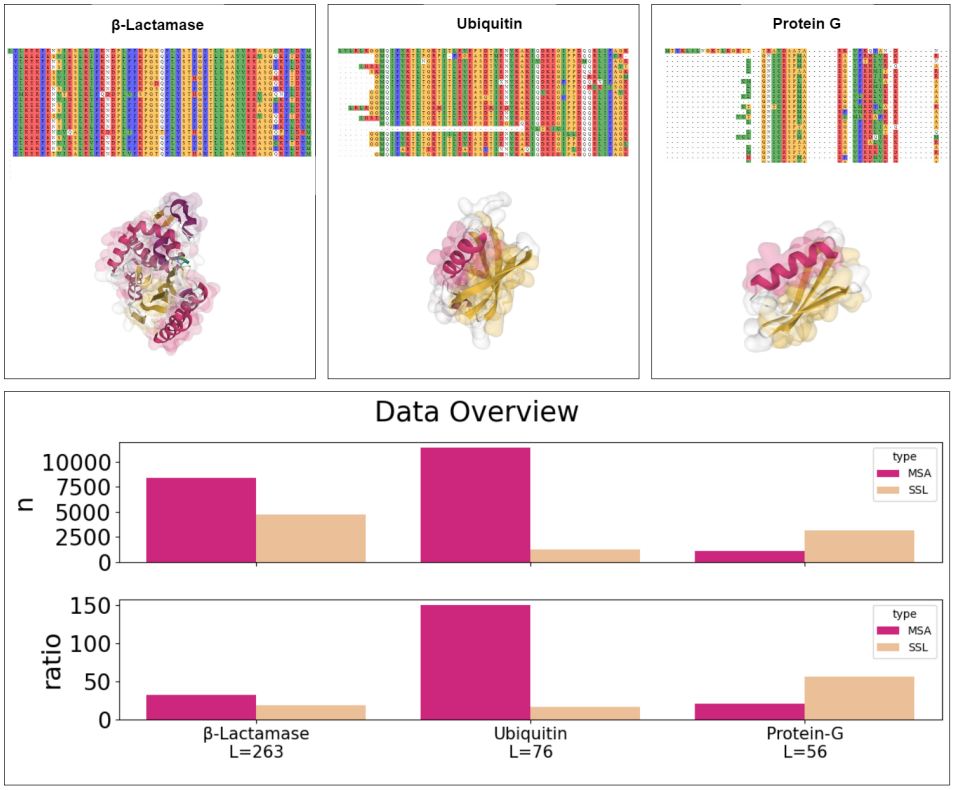


Figure 4.2: Available protein data.
We have primary structure from Multiple Sequence Alignments (top-row, with the conservation of residues color-coded). We obtain the secondary structure information from pdb files (below MSAs, with yellow=sheets, pink=α-helices, white=coils) from which the contacts within neighborhoods around the atoms of the residues are calculated (surface marked as blur around residues 5Å radius). The Proteins have a tertiary structure, their shape in three dimensional space. The bottom row is the count of available data-sets (with an overview, top-barplot) for protein-family data from the MSAs (pink) used for downstream VAE training and experimental variants (SSL, orange) for predictions and evaluation. The count can be taken relative, to the length of the sequence and is therefore also given as the ratio $\frac{n_{\text{Sequences}}}{L_{\text{Protein}}}$ is (the bottom row).

4.2 VAE Representations as Clusters

We are going to evaluate the VAE's learned lower dimensional representation. This can provide insights into the quality of the latent embedding of the model and vice versa into the underlying MSA sequences. The basis for our investigation is a VAE per protein for subsequent experimental analysis. We train a VAE on the Multiple Sequence Alignment of the respective family[7]. To display the lower-dimensional latent representation a VAE with z=2, was trained. The μ of the two-dimensional latent dimension was used and the results can be seen in Fig. 4.3.

[7]The specific architecture can be found in the Appendix table 7.3

From this resulting lower dimensional representation we see that β-Lactamase shows the most differentiated, star-shaped embedding, in latent space. Ubiquitin too shows differentiation in the lower dimensional representation in the form of clusters. However the clusters display less differentiation and a large cluster in the range of $2 \leq x < 4, -2 \leq y < 4$.
Protein-G shows the least differentiation in latent space, with one major cluster. This is to be expected due to the least amount of available sequences. Some points in Fig. 4.3 are outliers, which can come from unrelated sequences in our MSA.
We have verified that VAEs capture lower lower dimensional information from initial sequences and the properties of the MSA has an impact on the resulting impact

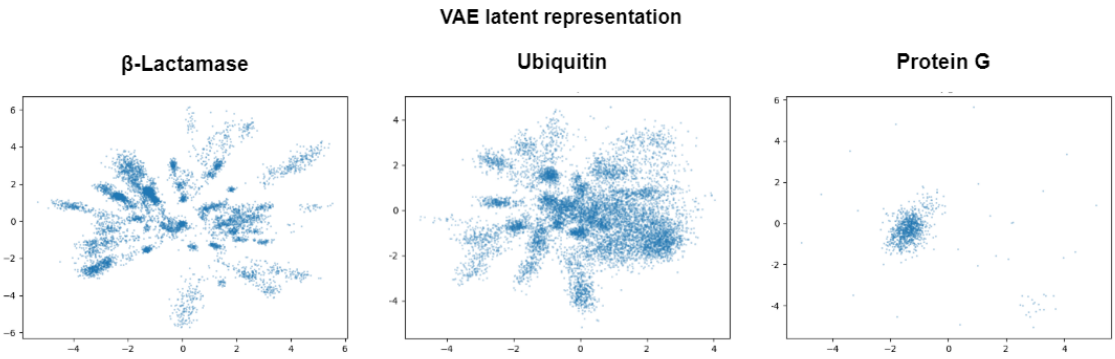


Figure 4.3: VAE two dimensional latent representation.
Each blue dot represents the latent value of an encoded sequence from the experimental data-set. β-Lactamase (left), forms a star shape in the latent representation of the sequences. Ubiquitin (middle), shows clusters in a less differentiated embedding. PGA (right), has the least sequences in the MSA and one cluster is formed in the latent representation.

4.3 Experimental Case-studies

We now evaluate the results for the three workflows that we have introduced earlier - see Fig.4.1. Firstly, we obtain the results for the mGP workflow (1), which uses only the experimental observations and the weighted sum over substitution matrices (MKL). The results of the initial mGP method give an overview of the general method performance and can be found in supplementary 7.2. We obtain and evaluate the ΔELBO values (2) of the experimental sequences, which we scale and use as in-silico information and incorporate into the GP regression. We apply and analyze the novel covariance function (3) with respect to the experimental data. Lastly, for the short Protein-G and Ubiquitin we also combine workflow (2) and (3). The benchmarks for all methods are placed at the end of this chapter 4.4.

4.3.1 Including the $\triangle$ELBO

Evaluating Log-Likelihoods

At this point, we have fitted lower dimensional latent representations from MSAs per protein. Now we are going to include the captured information as substitute in-silico data for our predictions. In order to effectively do that, we have to evaluate the obtained likelihoods against experimental observations.
From the learned VAE we derive the log-ratio as the difference between the mutational variant and wild-type in log-space. An overview of the underlying distribution and the correlation can be found in Fig. 4.4. The results are displayed in Fig. 4.4 and show that the log-likelihood ratio captures protein properties. The figure shows the spearman correlation (red-line) of the ΔELBO against experimental observations. The distribution of the experimental as well as log-ratio values for each protein are displayed atop and right next to the scatter-plots. We see that β-Lactamase has the highest correlation of $r = 0.636$. [8] When accounting for all sequences in the experimental observations our correlation is $r = 0.62$. We observe that Ubiquitin correlates *over all experimental sequences* with $r = 0.26$ and Protein-G has the lowest of $r \approx 0.028$, which constitutes almost no correlation.
We have shown that VAEs capture information from the protein sequences, because the latent representation correlates with experimental observations. This is the basis for our in-silico inputs and to evaluate likelihoods in a covariance function.

[8] However this applies only to the a sub-set of the β-Lactamase experimental sequences, as presented in [68].

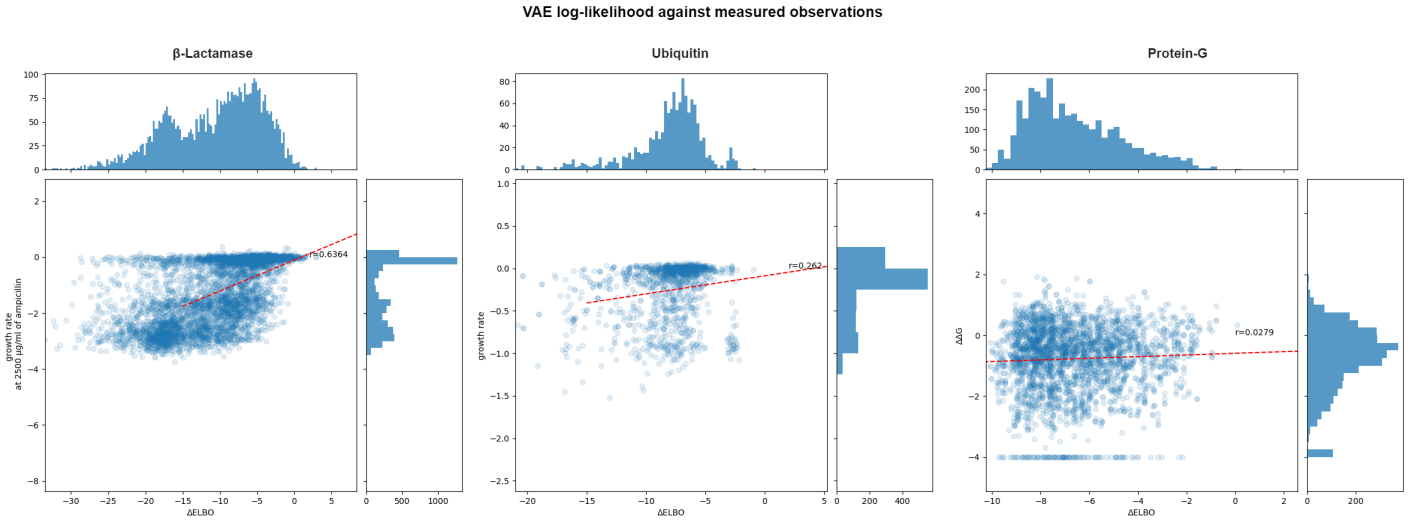


Figure 4.4: ΔELBO correlates with experimental measurements.
The blue dots in the scatter-plot show the log-likelihood ratio (x-axis) against experimental observations (y-axis). The correlation is displayed as a red, dashed line per protein. β-Lactamase (left) against the measured growth factor (spearman $r \approx 0.636$), (middle) Ubiquitin against yeast growth rate ($r \approx 0.26$), Protein-G measured change in thermodynamic stability ($r \approx 0.028$). Histograms show the respective distributions over the the values of the latent samples and experimental observations.

Applying Bayesian Regression

We are going to use Bayesian Regression scaling on the ΔELBO values conditioned on only few experimental observations. The log-likelihood ratio values were obtained by encoding each sequence in the experimental data-set and initially these values are in the range of the encoding latent representation. We require them to be in the range of the experimental values to suffice as in-silico values.
To fit a Bayesian Regression model, we sampled 10% of the experimental observations uniformly. These subsampled experimental observations used to train the transformation were excluded from later downstream tasks, as to not bias the later cross-validation. To fit the posterior exponential function (Eq. (3.3)), inference was conducted with the experimental subset as ground-truth y_E values to scale the in-silico substitutes from the latent representation. We perform inference using a Markov Chain Monte-Carlo (MCMC) algorithm for 10.000 steps plus 500 warm-up steps was conducted using a No U-Turn Sampler (NUTS) sampler [53, 31, 6]. We transform the ΔELBO values to the range of experimental observations, as can be seen in Fig. 4.5. There the sampled transformation functions can be seen as grey lines in the background. We use this regression function also to obtain the variance for each data-point. This is displayed as red bands (Fig. 4.5), which are tighter in the regions of experimental observations and larger in peripheral regions.
This operation preserves the correlation of the ELBO values towards the observed measurements. The experimental measurements for both β-Lactamase and Ubiquitin are growth factors, therefore the values are largely smaller or equal to zero, whereas the binding domain of Protein-G are $\Delta\Delta$G values and distributed around zero, as evident from Fig.4.5. The fitted functions are non-linear, however the transformation for Ubiquitin and Protein-G appears nearly linear for the majority of the data.
Now that we have applied Bayesian Regression and fitted a transformation function per protein, we have transformed the values from the log-likelihood ratios to in-silico proposals, which can now be added to the mGP workflow.

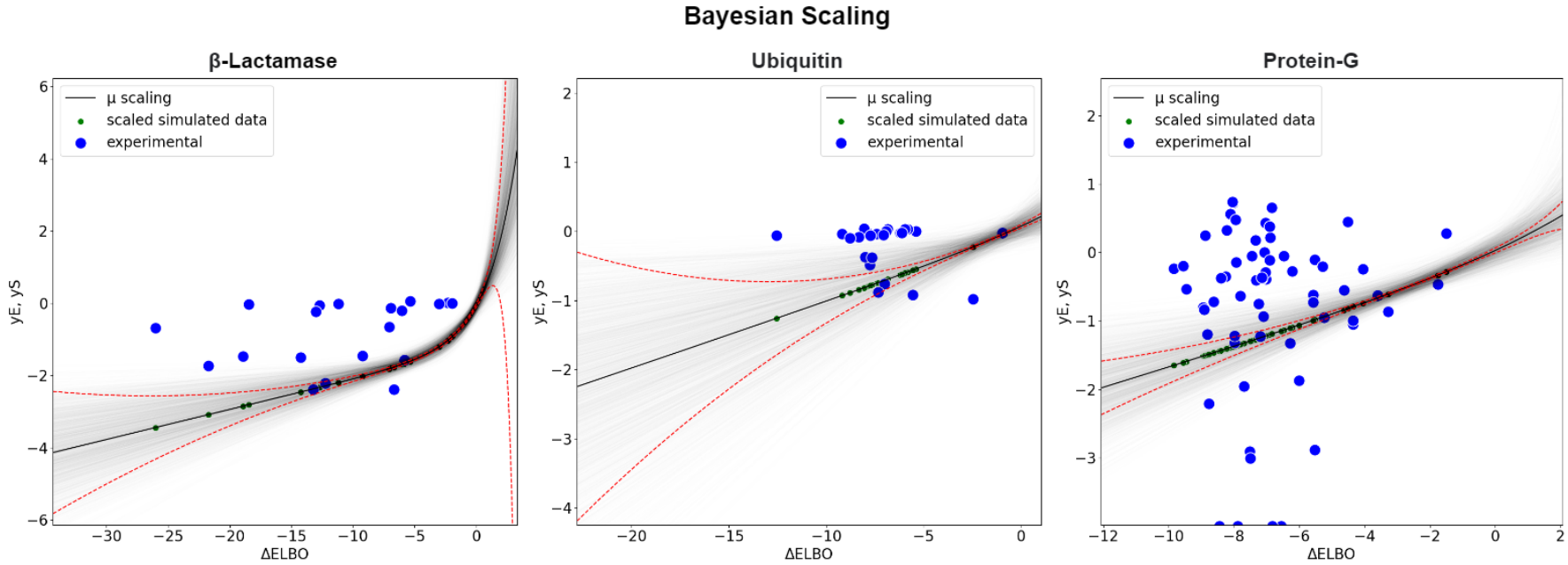


Figure 4.5: Bayesian Regression of ΔELBO values.
The ΔELBO values (x-axis) were conditioned on a subset (10%) of the experimental observations (blue-dots, y-axis), that map onto the transformation function (green dots on black line). The uncertainty (σ) is captured as red bands. Samples of the inference are plotted, (grey) in the background. This was conducted for all proteins: βLactamase (left), Ubiquitin (middle), Protein-G (PGA, right).

4.3.2 Evaluating the Covariance Function and DES-Kernel

A different means to include information from the learned latent representation is through the covariance kernel. The covariance matrix gives us an insight into the pairwise relatedness of variants. We are going to evaluate the proposed function with the experimental variant data.
During the development of the method, a naive iterative kernel was built and to directly implement with reference to Eq. (2.10). Additionally, a kernel with logged operations was built and tested to assess numerical stability of the implementation with respect to the naive kernel - see Eq. (3.37). The experiments were conducted with a vectorized kernel implementation (see supplementary 7.1). The vectorized implementation was tested for nearly-exact results against the naive kernel on short sequences[9]. After numerically stabilizing the covariance function, results for larger proteins could be obtained and the covariance function and has been applied to the proteins of interest, with the MSA variants as well the experimental variant observations. Furthermore, we confirmed that the weighted sum over all residues at a position is 1 (in reference to Eq. (3.24)).
In order to assess the numerical properties of the covariance function and to conduct Gaussian Process regression the covariance matrix for all mutational variants has been computed. All our experimental variants are single-site mutations.
For a subset of 20 variants per protein the computed kernel values are displayed in Fig. 4.6. The **covariance function values do not result in a definite kernel**. In the figure 4.6, brighter values are associated with higher likelihoods. We can see that the computed matrices are not p.s.d., given that the diagonal values, equal to one, are smaller than some of the off-diagonal values. This is the case for all observed proteins. That means that the computed gram-matrices for each protein are not positive semi-definite and subsequently not a covariance matrix and not applicable to a Gaussian Process The computed values are distributed in a range close around 1.[10] We would have expected the values distributed between zero and one, where one is the diagonal value on the diagonal indicating that the pairwise value for variants with themselves are highest.
We have investigated the covariance function values and found that we did not achieve a p.s.d. covariance matrix, which suggests that we need to make changes.

[9] In this case the reported accuracy is a difference of $\leq 10^{-4}$.
[10] The distribution of resulting covariance function values over the whole matrix for the individual proteins, see Appendix 7.4.

We can enforce positive semi-definiteness and symmetry to obtain a p.s.d. kernel. At this point we could conclude our analysis into the proposed covariance function. We have shown that the kernel is not positive (semi-)definite and therefore not applicable. The differences between the likelihoods are small, which can be explained partly, because the data are single-point mutation variants.
The subsequent steps are taken extra in order to investigate, if covariance function values capture properties of the latent representation, given that we enforce definiteness and symmetry. Therefore to continue the analysis the difference between the highest off-diagonal value the diagonal was taken plus a constant ϵ and added to the diagonals, in order to make the covariance matrix psd. For β-Lactamase $\epsilon_{\text{BLAT}} = 0.1$, for Ubiquitin $\epsilon_{\text{UBQ}} = 0.204$, and for Protein-G $\epsilon_{\text{PGA}} = 1.4$.
The resulting covariance matrix was used in the evaluation of the Gaussian Process regression.

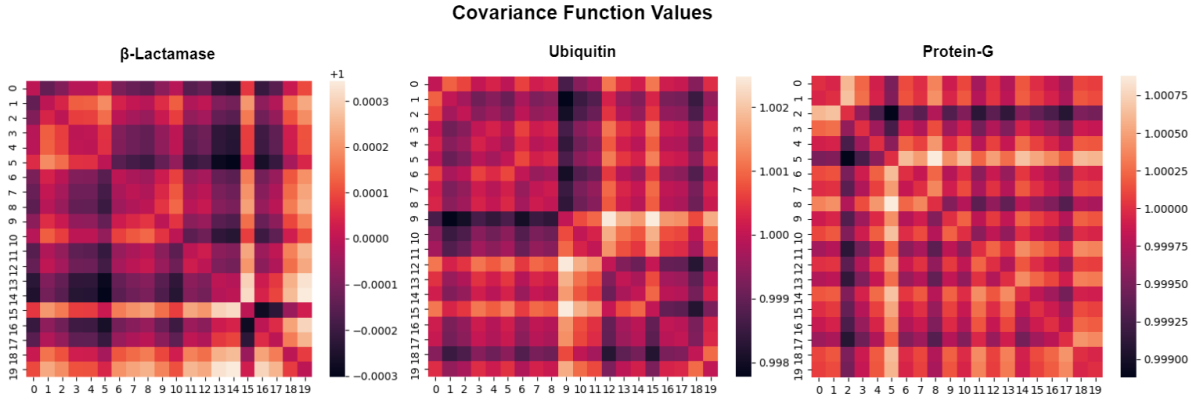


Figure 4.6: Covariance function values for experimental variants.
The selected subset are the first 20 variant sequences of the experimental (single-mutational variant) data-set for each protein. β-Lactamase variants (left), (middle) Ubiquitin variants, and Protein-G variants (right). The covariance function values have been computed for each and we see that some off-diagonal values are larger than the diagonal values for each of the matrices.

4.3.3 Deriving a Substitution Matrix Equivalent

For the mGP and mGP+ΔELBO we use substitution matrices to compute scores from log-likelihoods. The proposed covariance function formulation allows us to compute such likelihoods from our embedding. This can provide additional insights into the VAE and underlying sequence information. Therefore we are going to derive a substitution matrix equivalent, given a trained VAE. A subset of the sequences was used, analogue to the covariance function computation[11]. Instead of evaluating the likelihood of an

[11]The subset of the experimental data are the first $n = 1000$ sequences.

individual residue at a given position, we compute the mean per residue over all samples over the sequence, so that:

$$S(X) = \frac{1}{L}\sum_{i=1}^{L} \log\left(\frac{1}{p(x_{\neg i})}\frac{p(x_i|x_{\neg i})}{p(x_i)}\right). \tag{4.1}$$

[12] As seen in Fig. 4.7 we obtain a substitution matrix equivalent for β-Lactamase (left), Ubiquitin (middle) and Protein-G (right). This matrix is position, specific and not as general in its application as e.g. a BLOSUM or PAM matrix. The matrices were also computed for the Multiple Sequence Alignments used for training of the VAE (see Appendix Fig.7.3). In the comparison of the computed matrices from single-site variants to the MSA, the biggest differences in values are evident for Protein-G. Generally, we observe a block-like structure in their make-up, which is comparable to the matrix that arises from the outer product of the log-likelihoods of a standard-normal, sampled vector (see Appendix Fig. 7.2). From a comparison, with Fig. 7.1, we notice differences to classical substitution matrices. For the matrix derived from the latent representation of β-Lactamase we notice low log-likelihoods for the diagonal-elements of e.g. C and W, which make for some of the highest log-likelihood values in the PAM and BLOSUM matrix (Fig.7.1). Generally the diagonal value, which are the highest observed log-likelihoods in the substitution matrices are not distinguishable in the covariance function derived matrix for either of the proteins. What the embedding derived S-matrices have in common with the original substitution matrices is the higher likelihood bands for the S and T columns, as well as V. The likelihood for a gap is highest for Protein-G, this can be explained by the underlying MSA used for training, with the highest relative gap-count. The resulting matrices give us ground for later discussion (see section 5.3.1).
After having computed covariance function values, we have shown that we can compute something *like* a substitution matrix. However, the obtained matrices do not share the same properties as classical substitutions like BLOSUM.

[12] For brevity the stabilizing constant adjustments have been omitted.

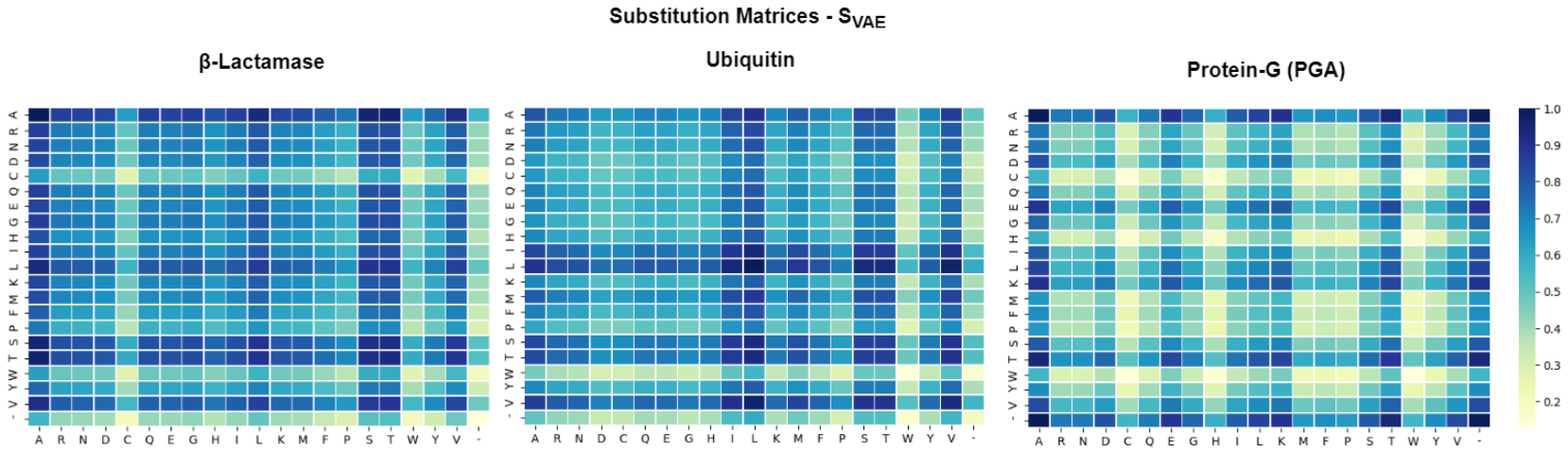


Figure 4.7: Position-specific substitution matrix.
The S values over a subset of the experimental data ($n = 1000$) has been computed with the mean value over all sequences, for each encoded residue. Darker colors correspond to higher log-likelihoods e.g. columns for A,S,T (left, middle) of a pair of residues interacting with each other. A higher-value diagonal is not distinguishable.

4.4 Benchmark Results

To conclude the analysis of the experimental results we compare the overall performance of the methods for each protein. In order to get a reliable estimate we use test-train splits of the data based on the positions for each protein, as position-level Cross-validation (CV).
As experimental evidence we collect Ubiquitin and Protein-G runs over all available experimental mutational variants for each of the methods. For β-Lactamase a subselection of the first 1000 experimental variants was taken, to reduce the overall run-time. We evaluate each method through position-level cross-validation. This means that as many CV steps were taken, as number of positions exist in a protein sequence.
For each position at a time all mutational variants are taken as test-data; while the rest of the data, namely all variants not having a mutation at that position, were considered for training data. Each CV run per protein was conducted with a 25%, 50% or 100% sub-sample, taken uniformly from all mutational variants in the training data. When incorporating in-silico data, the uniform sample was conducted over the experimental data plus the in-silico simulations. Under this set-up we can compute the spearman r and MSE for each position.
The overall results for each method are displayed in Fig. 4.8: The boxplot shows the distribution of spearman r and the mean-squared error based on the results from each individual position.
Figure 4.8 shows the mean and quartiles of the measured correlation and mean-squared error over all positions from the CV run per protein. For additional investigation, irrespective of positional results, the ρ, spearman r and root-mean squared error was computed for all generated predictions against all ground truth experimental observations, see supplementary Fig. 7.6.
We observe that the use of the in-silico data (added scaled ΔELBO), as well as the proposed augmented kernel lead to less distributed correlation coefficients and MSE. However we see from the position-wise individual results (see 7.7) that the values for the introduced kernel have generally been constant, which can result in more compact MSE values. The results for the proposed kernel show next to no correlation for the majority of the positions. For Ubiquitin and β-Lactamase we observe an improvement on the mean of the correlations, which we do not observe for Protein-G. We get a mGP mean of > 0.4 and < 0.5, with a higher correlation when adding the in-silico information for β-Lactamase. It is also the case, that we observe a significant improvement of the correlations for Ubiquitin, including the in-silico data.
The latent embedding of β-Lactamase had the highest reported correlation, therefore we would have assumed the highest performance when incorporating in-silico val-

ues. There is in general an improvement in correlations of positions evident - this is underlined by the position-wise analysis in supplementary Fig. 7.7. Combining the DES-kernel and in-silico data does not yield an improvement; neither for Ubiquitin nor β-Lactamase.
A tabular overview over the prediction metrics per position of the positional cross-validation can be found in supplementary tables 7.5 and 7.4.
We have assessed the performance of the individual methods through position-level cross-validation. We have observed an improvement from including the in-silico values, however not for the protein with the uncorrelated latent embedding (PGA).

Lastly, we want to investigate the performance and properties of *individual* predictions against the experimental values. One feature that distinguish probabilistic models like Gaussian Processes from other methods are uncertainty estimates per prediction. We are going to utilize this to get an estimate of the predicted values. For individual investigation three randomly selected β-Lactamase variants are displayed in Fig.4.9. We can see the predicted growth factor (x-axis) against the measured value (y-axis) from mGP (left), mGP+ΔELBO (middle) and mGP+DES-kernel (right). From mGP to mGP+ΔELBO, we observe slight adjustments on the y-axis, which is a trend that is confirmed for the overall predictions (see Appendix Fig. 7.7). From the computed covariances, we obtain the standard-deviation, which is a suggested uncertainty that decreases when we add in-silico information. Per prediction a grey curve describes the uncertainty per estimate, as we rely on Normal distributions as an underlying assumption. Given that we have an overall improvement in the correlations and a decrease in the MSE we expected a decrease in uncertainty, when incorporating in-silico ΔELBO information. The use of the DES-kernel, leads to a slight reduction in the uncertainty and a shift on the y-axis. This does not compensate for the uncorrelated results.
In conclusion, we have seen that the performance trend, for the overall results, when incorporating transformed in-silico data, leads to a slight improvement on the predictions and a reduction in the standard-deviation per prediction.

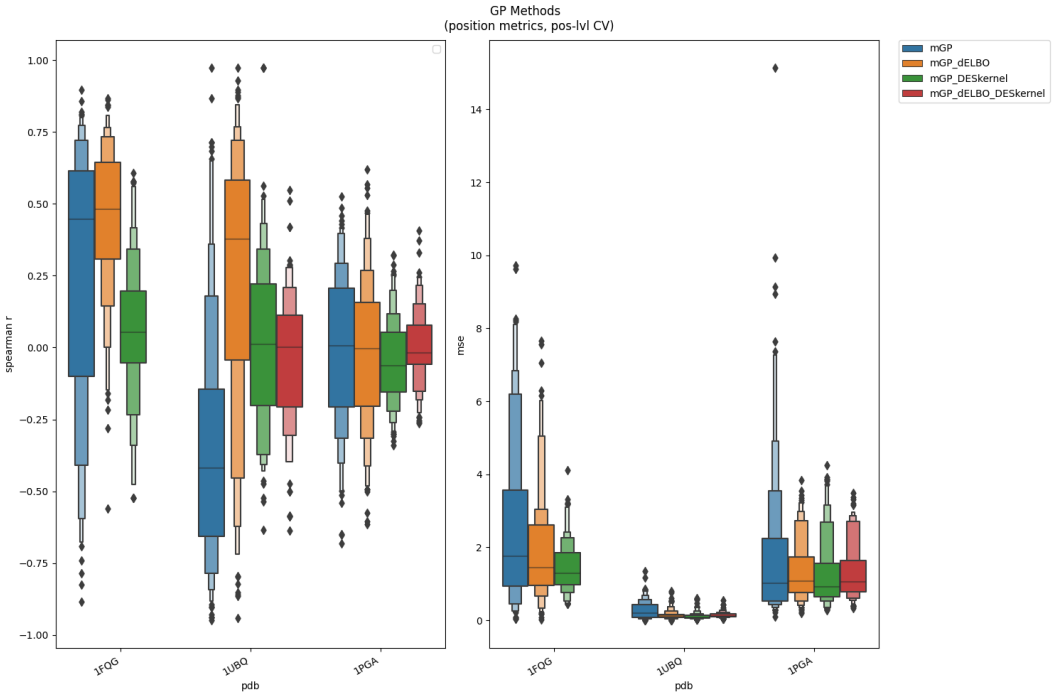


Figure 4.8: Performance over positional results for position level CV.
Results of spearman r correlation (left plot) as well as mean-squared-error (right plot) from each position for each protein (x-axis: 1FQG, 1UBQ, 1PGA). For mGP (blue), mGP with ΔELBO (orange) runs added, mGP with DES-kernel (green) and the combination of both ΔELBO and DES-kernel (red). Metrics have been computed for each run-configuration (25%, 50%, 100% training data), while the boxplot accounts for all results per protein. See appendix for an overview of the individual results 7.5.

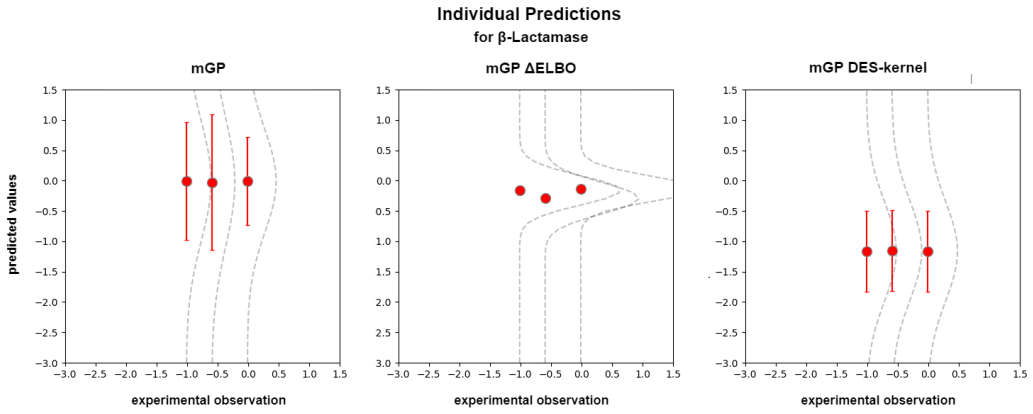


Figure 4.9: Selected individual predictions from position-level CV.
Selection of three β-Lactamase measurements (y-axis) with their respective predictions (x-axis) and predictive covariance (red-band) and a grey, Gaussian band as the resulting distribution around each prediction. We compare mGP (left) with mGP added ΔELBO (middle) and mGP with the DES-kernel (right).

5 Discussion

5.1 Limitations of the Conducted Case-Studies

The obtained results are a limited experimental validation for the proposed concepts. More experimental results have to be collected, from different quality latent embeddings, before definitive statements can be made. The obtained results suggest, that adding the transformed ΔELBO values do improve predictions, both for the correlation of the predicted values, as well as the trend inMean-Squared Error (MSE). However, improvements could not be achieved for the investigated variants of Protein-G, which was anticipated, as this protein held almost no correlation of the latent representation log-ratio against experimental observations. The correlation of β-Lactamase were not as strong as initially assumed. The latent representation had a correlation coefficient greater than 0.6, however the mean over the cross-validations did not tend to 0.6 - even though the overall performance per position displayed better correlations (see Fig. 7.5). In the final assessments, differences arise when assessing the performance of the mGPfusion method, based on cross-validation and the performance of the VAE based approach; due to the nature of cross-validation.

5.1.1 Cross-Validations for Proteins

During experimental conduct cross-validation has been performed, to give an assessment of the method: we iterate over all data-splits as to have one portion of the data for training and the rest for testing, such that after the CV run, all potential parts of the data have been in the training data-set or in the testing data-set at one point in time. The CV protocol presented in [38] was used for comparability of the results. This allows for a generalizable assessment, compared to just observing the distinct positions or single splits. However, this experimental conduct does not apply to the Deep Sequence method. The values e.g. log-likelihoods were directly computed from the model and no hold-out dataset or cross-validation has been conducted. Though, during training of the model we perform a split into training and validation data for fitting the VAE, we do not assess the performance of the VAE on the final distinct test-set. Evaluating the latent variable model in this context is not directly comparable to the mGPfusion method, which has to be taken into consideration when assessing the results.
The performance reported through this conduct is lower than with mutational-level CV,

as is supported by the results of mGPfusion [38]. In this context *mutation level* means Leave-One-Out cross-validation where we leave out one mutational variant at a time [28, p. 243]. Another viable form of CV is to randomly select whole segments of the protein for exclusion and evaluate against the rest. Mutation level CV has not been conducted for this work, and it should be noted that mutation level CV performs at least as good and in the majority of cases better than position level cross-validation. This can be explained, because we eliminate only one mutational variant at a time. This potentially exposes the prediction for a position that is currently in testing to take into account values from the other variants that are available at that position. When we eliminate all variants from one position at a time, as we do in position level CV we have a harder problem to solve.
Performing efficient and informative cross-validation on protein data is an open issue in Bioinformatics and Machine Learning in Life Science [45, 39].

5.1.2 Impact of Assessed Metrics

In our case-studies and with reference to previous work we use the (root-)mean squared error, a correlation measure ρ and the spearman r correlation coefficient. We compute the correlation of the predictions to the underlying true experimental observations. Furthermore we measure how much the predictions differ from the experimental measures by the mean squared error. This estimate specifically might hide information differences - e.g. we have constant predictions which will result in lower MSE values than an equally bad model that results in volatile predictions. Using both methods to assess the methods gives us a clearer picture. However either metric by itself can be misleading.
The results with respect to the (R)MSE appear to increase with an increase in available training data. However, we would expect the opposite to happen, such that when we increase the data-set we get a decrease in error for our predictions. One way to investigate this behavior is sampling from the prior distributions, making predictions and assessing the with an increase in data-set size. If this behavior is consistent, it could suggest that sub-sampling from the same training data leads to overly optimistic estimate for smaller sample sizes. As we evaluate the discrepancy between the predictions and the true underlying data, with more data from the same distribution we get an increase of in-sample error [28, p. 228]. Further investigation into the error estimate is required.

5.1.3 Why the sum of log-likelihoods is not a good assessment with respect to the structure of the protein

In our workflow we integrate prior information through different means. One of the ways we integrate information that we have of our problem domain, while reducing computational complexity is through the structure information, obtainable from pdb files. From the coordinates we derive neighborhoods and from neighborhoods we derive contacts in order to assess the residues with the residues that it is in closer contact with. We assess the covariance of variants by the weighted sum over the sequence and a variant sequence also by incorporating the sum over the neighbors. I argue that this is potentially not a good way to integrate structure information.
One of the shortcomings is that in most cases, the structure of a protein to a variant does not account for structural changes. Substitutions of bigger impact mutations or multiple position substitutions within a variant have the potential to disrupt the structure of a protein [22]. In our model we do not account for such changes to the structure in from our mutational variants.
I argue that there need not be an explicit structure formulation necessary for the computation of covariance values and likelihood assessments of protein variants. One extending hypothesis is that a well-aligned MSA captures structural information in itself, implicitly. For example, current state of the art models can make structural predictions while having only learned from sequences [69]. By fitting deep learning models we can learn structural information implicitly from the variant sequences.
We require the structural information if we conduct something like a position-specific substitution likelihood for our kernel. The results show that this is not optimal and incorporating the structural information in a different way, liberates us from the restriction that mutational variants impact contacts and structure in the protein, which we cannot sufficiently model from the contacts derived from a pdb file. Utilizing information captured from learned representations can be element for further investigations and has the potential to improve covariance kernel computations.

5.2 Review on the VAE Work

Throughout this work, we have taken a look at two different state of the art models and their evaluation under a variety of conditions. For example the Deep Sequence approach takes into account multiple design choices to optimize the VAE results, which were not accounted for in this work. An ablation study has been conducted by the authors of the Deep Sequence method on the architecture choices associated for the

VAE (see Supplementary 5 [68]). This can be viewed critically because the optimal model is derived empirically. Machine Learning theory indicates that sparsity layers and mappings can improve on (probabilistic) model performances [55, pp.386-387]. The predictions by the Deep Sequence have been made with an ensemble of VAEs. In total the VAE implementation for this experimental result lacked different prior initialisation, sparsity layers, and linear mappings with respect to the Deep Sequence model. The underlying motivation for the model parameter choices offers the potential to be explored in depth apart from its experimental justification.
It has to be added that, not all experimental measurements were reported in the predictions made from the VAE ensemble. Specifically positions, which showed too many gaps in the underlying training MSA have been excluded, (see supplementary tables in 3 [68] and [32]). In the benchmark results of this work all experimental observations for all positions were reported.

5.2.1 VAE Latent Information

Assessing the potential impact that a prior distribution has, leads us directly to the learned latent representation of the VAE. The results suggest, that given a higher correlated latent embedding can give rise to a better in-silico substitute. We discuss the hypothesis that more diverse MSAs can lead to better predictions of the model. Further investigation is required into, how the highly correlated log-likelihood of our example β-Lactamase do not yield improvements in the same order of magnitude when incorporated as in-silico information. One starting point is to investigate if this also applies to other proteins, with diverse MSAs, which have high correlations, such as β-Glucosidase or YAP 1 (see Fig. 3 in [68]). This is directly related to the potential research question, what relationship exists between the correlation of the log-likelihood ratio and the experimental values with a decrease in correlations and the impact on the improvement factor towards mGP.
The method presented in this thesis relied on using an optimized encoder-decoder architecture to capture good correlations for e.g. β-Lactamase. The performance of the VAE presented in this work is therefore not optimal. The built models were optimized with respect to β-Lactamase. The presented architecture has not been optimized for Ubiquitin or Protein-G. One can find an optimal architecture for each protein by running an optimization routine to iteratively fit different VAEs on a subset of different experimental data, which would result in different but optimized VAE hyper-parameters per protein.
Lastly, the Deep Sequence method found significant improvements using *sparse-layers* and *linear-mappings* with their Variational Auto-Encoders [68].

5.2.2 Assessing the Effect of Different Priors

Previous work has shown that different prior distribution for VAEs affect model properties and applicability in the domain [14, 68]. Already Riesselman's work has shown that different priors and initialization can lead to better results than using a standard normal initialization.
Hyperspherical VAEs, which are based on a von-Mises distribution instead of a standard-Normal distribution have been hypothesized to model protein sequences adequately. Additionally the use of MSAs as a basis for training data can be viewed critically. Using a MSA as basis for training has the potential to introduce fallacies in itself, to the learned latent representation, which require mitigation [86].
Different prior distribution properties or derived MSA distributions have not been assessed. The investigation into the applicability of different prior distributions and the impact on the model-performance, offer a basis for continued work on the subject.

5.2.3 Changes in ELBO are Uninformative

The crux of the reported ΔELBO values is that they are not directly applicable in protein design. Even though we obtain ΔELBO values which are highly correlated with the true measurements of protein properties, we cannot use the values directly and extra steps are required to transform the values to an applicable range. This might pose a hurdle in the practical work with the models. In this work we have shown that Bayesian Regression is one means to do this. One can use a small subset of the experimental observations to fit a transformation function. Once a transformation function has been obtained, one can work with the transformed ΔELBO values. I argue that Bayesian Regression is a good method to perform this. We can rely on small subsets and use MCMC inference to compute the posterior. Further we get uncertainty estimates with our σ of the transformation function. This can be used downstream in the GP regression noise term.
One can argue that Variational Inference may be applicable to fit the regression, however MCMC can better guarantee exactness - a topic which has been explored in Salimans, Kingma et al [73]. Overall, Bayesian Regression is a suitable tool for transformation of in-silico values.

5.3 Kernel Function Likelihoods and their Interpretation

The initially proposed covariance function does not give rise to a positive semi-definite kernel. This requires further analysis. Specifically the resulting log-likelihoods for the substitution of a variant with itself is less likely than the likelihood of observing it with respect to another variant, as is expressed in Fig. 4.6. This counters our intuition about the intended computation. We would have assumed the likelihood of a variant with itself to be larger or equal to the most likely other interaction. This extends to the derived substitution matrices as well. As evident from Fig. 4.7 the log-likelihoods on the diagonal are in parts lower than values on the off-diagonal. An interpretation of this observation is that the likelihood of observing a residue being substituted is more likely than it staying the same.

As an example, let us consider two sequences X, X' of length 250 which vary in one residue at the same position i. This case is comparable to the assessment for the variants of β-Lactamase. When scoring the elements x_j, x'_j such that $j \neq i$, with the proposed function, we may find an A, and so we get a log-likelihood from $S(A, A') = \log(\frac{p(A \leftrightarrow A')}{p(A)p(A')}$. In this example, the encoder is the same for both sequences, the latent z_x is nearly the same as $z_{x'}$. In order to get a large log-likelihood value we would require $p(A \leftrightarrow A)$ to be sufficiently large. However, we expect $p(x_j = A|z_x)p(x_{\neg j})$ is nearly identical to the term observed for x', also $p(A)$ is nearly identical to $p(A')$ and so are the values obtained from q. All in all, we do not get a large log-likelihood, even though we just queried for a $A \leftrightarrow A$ substitution. This makes the proposed covariance function problematic for the evaluation of single-site mutational variants.

A potential mitigation would be to add to the diagonal in order to explicitly express our belief in the conservation of this residue or to find an alternative treatment of a residue substitution with itself. In the experimental conduct we make this explicit by the fact that we add a constant to the diagonal.

There exist different means to make the covariance matrix positive semi-definite. Instead of min-max normalization of the S-values, one could apply the exponential function, as to ensure positive values. However this would convert the log-likelihoods to likelihood values.

5.3.1 Why We don't get a BLOSUM Matrix

In the experimental runs, we have computed the log-likelihood values from our covariance functions to assess the likelihood of a substitution at a given position. Not assessing a specific amino-acid configuration at a position allows us to compute a

more general overview of the log-likelihoods given the covariance function, e.g. a substitution matrix. We notice substantial differences between the substitution-like matrices that we compute and the original substitution matrices like BLOSUM, PAM and others - compare Fig. 4.7 and Fig.7.1. One major difference is the diagonal. Our covariance function suggests that the likelihood of observing a substitution from one residue towards a residue of the same type, given the likelihood of the sequence - is lower or the same as the substitution to another residue. The original substitution matrix suggests that observing the likelihood of a substitution of the residue with itself is more likely than to another residue - in the majority of the cases. Making a same-residue substitution more explicit, potentially improves the covariance function and subsequently the substitution matrix equivalent. We always evaluate the substitution matrix and likelihoods of the sequence-elements in the context of the latent random variable. Taking into account artifacts that arise from the VAE such as a standard Normal prior.
I argue, that *if* we get a BLOSUM-like matrix with higher likelihood values across the diagonals we would subsequently get definite covariance kernel values. Fixing this discrepancy may lead to a corrected covariance function and further investigation is required.

5.4 Research Outlook

The reported work only looks at one covariance function. Here we explore what different potential kernels could be used in the context of the latent embedding. Instead of assessing the likelihoods of potential position specific interactions of residues we can also: (a) assess the spatial relations in the latent representation or, (b) use a Fisher kernel that takes into account the derivatives of the VAE optimization. By doing so we also eliminate the need to sum over the sequences and neighborhoods, taking into account explicit structure information.
Since publication in 2018, the Deep Sequence method has since been outperformed by transformer models [69]. This suggests that using information based on language models can be the next step in improving the workflow. For example replacing the VAE architecture with a transformer model and deriving something like a log-likelihood value to incorporate in a GP regression workflow may lead to better performance than VAEs.

5.4.1 Enabling Learning on Problematic Sequence Alignments

The VAE approach relies on a well-aligned, diverse MSA of sufficient quality. Failing to build a MSA for the problem domain means low to none improvement or limited applicability of the Deep Sequence or VAE based approaches. This also affects our proposed workflow and results in a low-performance of Protein-G when incorporating MSA information. Recent research has shown that the use of Auto-Regressive models can mitigate shortcomings of MSA based alignments and enable better method developments without the need of prior alignments [76]. This work greatly benefits future research into the development of models which capture latent information on unaligned sequences. However using models from natural language processing in general can be a benefit for the method.

5.4.2 Quantifying Distances in VAE-Space

A potentially different treatment of the likelihoods in the latent embedding can be the interpretation of them as coordinates in the representation. One hypothesis would be that variants closely related to one-another are closer related in the latent space as well. This could already be useful for applying a stationary distance kernel from the likelihoods of the derived latent embedding.
Defining a covariance function like that does not allow the derivation of likelihoods at individual positions as is currently done. A coordinate in the latent space represents one sequence. It could be argued, that variants of single residue difference are closely situated around an area in the latent representation. However, queries for position specific residues cannot be made.

5.4.3 Fisher-Kernel Covariance Functions

A different and more involved approach to quantify the distance of the latent representations is to account to the derivatives of the latent representations as well. One way to achieve this is through a kernel as proposed in [88, p. 102]:

$$k(x, x') = \phi_\theta^T(x) M^{-1} \phi_\theta(x').$$

In this case M is strictly positive and can be a Fisher information matrix, making this a Fisher kernel. An alternative and easier computation for M is the identity matrix.
If we make the assumption that higher order information such as structural information,

can also be captured in the derivatives of the optimized latent representation of the VAE, the Fisher kernel can utilize this.

5.4.4 Potential in Advanced MKL

The mGPfusion workflow uses multiple kernel learning with different substitution matrices. One potential improvement for future work can be the integration of different kernels together. Now that we have proposed different variants of other potential kernels, we could again optimize for the weighted sum of the covariance matrices. This would mean combining the matrices computed from e.g. a distance kernel, Fisher kernel and e.g. the matrices computed from BLOSUM together. An implementation would allow to assess the weights from the MKL, which would indicate which kernel performs best for certain proteins. Again, the explicit structural information would be required for the classical BLOSUM based covariance matrix, but not taken into consideration for distance calculations. Both result in a covariance matrix, the weighted sum of which can be optimized.

5.4.5 Combatting Complexity

An inherent problem of the Gaussian Process regression approach is, that GPs do not scale well with an increase in the amount of data, since the complexity is $O(n^3)$. When we want to extend this model to larger proteins or a multitude of variant sequences we have to take into account a cubed increase in runtime. There exist approaches to mitigate this issue. One can incorporate sparse approximate GPs as proposed by Candela and Rasmussen [64, 52]. Sparse approximations hold the potential to reduce complexity and run-time of the operations which would allow to better scale this method.

5.4.6 Fourier Transform for Kernel Analysis

One analysis that has not been conducted with respect to the proposed covariance function is the Fourier analysis to extract spectral information [88, p.207-208]. The motivation of this analysis in the case of stationary covariance functions is that frequencies in the underlying signal, captured by the covariance function can be analyzed. This work has not been done due to the fact, that the kernel performed sub-par within the mGP workflow and due to time-constraints, and its low-rank properties.

5.5 Approximate Models break Closed-form GP Computations

One can use Gaussian Processes for closed-form computation to solve regression problems. This can allow insights into the model through likelihood evaluations, effect of the priors, data fit and others. However, our proposed method breaks the closed form computation by including two separately optimized methods, namely the VAE and a Bayesian Regression model. It is true, that we still have a log-likelihood formulation for kernel learning, which can be optimized. Now, the goodness of fit relies in part of the results from the covariance function. Our covariance function in turn, is the result of an approximation deep-learning based model. We have extra layers of uncertainty that are associated with the properties of the VAE architecture. This also includes the hyperparameters that are involved in obtaining an optimal architecture, which in turn can provide an optimal covariance function.
In order to make the whole computation closed-form, end-to-end computable, one can consider Deep Kernel learning (DKL) [89]. With methods such as DKL we can optimize the latent representation or any other model, from which we derive a covariance function and optimize in closed form with respect to the output of the Gaussian Process regression.

6 Conclusion

In this work we have introduced different methods for protein variant property predictions, namely the supervised mGP approach and the unsupervised learning from MSA sequences through a VAE. This work shows, that the mGPfusion method is applicable to other protein properties, such as growth factors.

We succeeded to introduce information from learned latent representations into the Gaussian Process regression workflow. The two proposed ways to achieve this was to add the change in the log-likelihood (ΔELBO) or deriving a covariance function based on the VAE. We succeeded in fitting a function to transform ΔELBO into an applicable range to constitute in-silico values. Given that we have a good base-correlation between log-likelihood ratios and experimental observations adding an in-silico substitute, improves predictions for β-Lactamase and Ubiquitin. If the latent representation does not show correlations as was the case for Protein-G, there is no benefit in adding ΔELBO values.

We succeeded in deriving a covariance function from the latent embedding. The resulting matrix is not a covariance kernel, as it is not p.s.d.. It is required to enforce such kernel properties on the computed covariance function values, in order for it to be applicable to Gaussian Process regression. Using the adapted covariance kernel in this context leads to no observed correlation of the predictions.

Other means to quantify the relations of the protein variants have been discussed, such as deriving a distance kernel from the latent representation or a fisher kernel for future work.

Furthermore the mGP workflow depends on explicit structure information, from which the contacts are derived before covariance matrices can be computed; this can be considered a bottleneck. I suggest, that closed form learning from different deep-learning based kernels, has the potential to improve the method. Integrating the latent representation directly and not optimizing a sequence-based model separately can be implemented through closed-form Gaussian Process regression.

The presented method contributions are a state of the art protein variant property prediction, that incorporates available information from protein structure, family sequence alignments, derived in-silico values and experimental observations. The Gaussian Process regression offers a framework to combine different input values for a range of protein properties and provides uncertainty estimates with its predictions.

Acknowledgements

I thank the supervisors of this work for their diligence, the right questions, critical review of my work and inspirations along the way. Specifically, Simon Bartels for the insights on Gaussian Process regression, voice of reason when assessing results and insights to numerical stabilities in the optimization process, for the requests to test for numerical exactness and correctness in the method development. Section 3.4.3 is a direct contribution of his. Further, Wouter Boomsa provided insightful supervision, hard questions in the process, motivating insights to keep the research work interesting, and proposed the original formulation of the DES-kernel covariance function. Pengfei Tian, kept an eye on competing Machine Learning methods, smooth integration of Novozymes resources and the practicalities of method assessments. The Bioinformatics and Protein Design department at Novozymes provided data-sets of interest and computational resources for experimental case-studies. Jacob Kaestel-Hansen provided a curated data-set on BLAT and introductory insights from his work on VAEs. The previous work by Emil Petersen gave inspiration to the implementation of sequence weighting for VAE training. I thank Tone Bengtsen, without whom I would not have dived into the paper by Jokinen et al. Jakob Madsen for review and suggestions on the initial drafts. Last but not least, the Bio-ML group at DIKU provided interesting discussions and ideas.

7 Supplementary Information

7.1 Metrics for Assessing Predictions

In order to assess the predictions made by models we rely on well established methods, which have also been used in the initial reports of the presented methods. We assess the correlation by spearman's rank correlation coefficient r in the range of -1 and 1. Given two samples X, Y and their ranks denoted as $\text{rank}_X, \text{rank}_Y$ it is defined as (see [79, p. 447]):

$$r = \frac{\text{cov}(\text{rank}_X, \text{rank}_Y)}{\sigma_{\text{rank}_X} \sigma_{\text{rank}_X}}. \tag{7.1}$$

For a more detailed definition see [94, p. 367]. With respect to the method assessment reported in the supplementary data [38, p. 2], we report the correlation coefficient ρ as:

$$\rho = \frac{\sum_i^N (y_i - \bar{\mathbf{y}})(\mu(\mathbf{x}_i) - \bar{\mu})}{\sqrt{\sum_i^N {}^*(y_i - \bar{\mathbf{y}})^2 \sum_i^N {}^*(\mu(\mathbf{x}_i) - \bar{\mu})^2}}. \tag{7.2}$$

The difference between predictions and true observed values is reported as the mean-squared error and root-mean-squared error (see [38, p. 2], [55, p. 220]):

$$\text{mse} = \frac{1}{N^*} \sum_i^N {}^*(y_i - \mu(\mathbf{x_i}))^2), \tag{7.3}$$

and $\text{rmse} = \sqrt{\text{mse}}$.

7.2 Method and Implementational Details

The pdb files were processed with biopython v1.78 [8] and coordinates calculated using scipy.spatial v1.6.0 . Bayesian Regression, the VAE as well as the proposed kernel were implemented with Pyro v1.5.1 [6] and PyTorch v1.7.1+CPU [60]. The Gaussian Process Regressor was implemented in PyTorch and Numpy v1.20.1 [27]. As a framework to collect experimental results for cross-validation and model training, MlFlow v1.14.1 [92] was used. Experiments were run on an Intel i7 4700MQ CPU with 32 GB RAM, and a workstation with an Intel Xeon CPU E5+2687Wv2. Plots and figures were created using Python's matplotlib v3.3.3 [34] and seaborn v0.11.1

[85] and figures composed in draw.io . The code is available upon request on the github repository of the author pro_tooling, the commit at the time of submission is on parent 28a3cc - commit: 324117fc0af02fd69c70bee2cd522b68759a1d78, last updated 30/05/2021 at 15:21 o'clock. Further details can be found in the project directory.

7.3 Multiple Sequence Alignment

The MSAs were generated in accordance with the MSAs from [32]. That entailed:

1. searching the protein fasta against UniRef100 database [81] over 5 iterations using HMM search jackhmmer [20], for PGA HHblits was used [67] [93],

2. the bit-score threshold was set to 0.5 $\frac{\text{bits}}{\text{residue}}$, unless less than 80% coverage against the target sequence was achieved,

3. if not enough coverage was achieved the threshold increased by 0.05 bits/residue per step, vice versa if too many sequences there was a step-wise decrease,

4. goal was a count of non-redundant sequences n $\geq 10L$,

5. exclusions of sequences with $\geq 30\%$ gaps or less than 50% alignment against the target sequence.

The properties for the Protein-G MSA were relaxed to E=0.1 and 10% acceptance probability for sequences.
hhblits -cpu 8 -i ../results/7566882.in.a3m -d /cluster/toolkit/production/databases/hhblits/UniRef30 -o
/ebio/toolkit_rye/user/toolkit/production/jobs/7566882/results/7566882.hhr -oa3m
/ebio/toolkit_rye/user/toolkit/production/jobs/7566882/results/7566882.a3m -e 0.1 -n 5 -p 10 -Z 5000 -z 1 -b 1 -B 5000

7.4 mGP Benchmarks

		mutations	
Protein	**PDB ID**	**experimental**	**in-silico**
T4 Lysozyme	2LZM	349	3116
Protein-G	1PGA	89	1064
Ribonuclease H	2RN2	83	2945
Cold shock protein B	1CSP	80	1273
Apomyoglobin	1BVC	80	2907
Hen egg white lysozyme	4LYZ	63	2451
Ribonuclease A	1RTB	57	2356
Hydrolase, Guanyloribonuclease	1RGG	54	1824
Bovine pancreatic trypsin inhibitor	1BPI	53	1102
TEM-1 β-Lactamase	1FQG	4799	0
Total		**5707**	**19038**

Table 7.1: A Selection of 10 proteins used for the initial evaluation on the mGPfusion reference data. The experimental mutations of the first 9 proteins are from the ProTherm database as reported in [38]. The in-silico point-wise mutations are results from Rosetta simulations. Lastly 1FQG does not have in-silico simulations, as this data was not part of the original mGPfusion analysis.

	measure	ρ	rmse
	CV	pos.lvl.	pos.lvl.
Protein	Method		
1BVC	**mGPfusion**	-0.112626	1.462445
	2σ mGPfusion	-0.122320	1.459480
	NO mGPfusion	-0.129107	1.473530
	NO 2σ mGPfusion	-0.132961	1.433076
2LZM	**mGPfusion**	0.653866	1.639737
	2σ mGPfusion	0.621332	1.737129
	NO mGPfusion	0.653866	1.639737
	NO 2σ mGPfusion	0.621332	1.737129
1PGA	**mGPfusion**	0.393455	2.230193
	2σ mGPfusion	0.322050	2.326907
	NO mGPfusion	0.335846	2.223115
	NO 2σ mGPfusion	0.226469	2.312001
1CSP	**mGPfusion**	0.785746	1.270779
	2σ mGPfusion	0.789426	1.261061
	NO mGPfusion	0.801479	1.221864
	NO 2σ mGPfusion	0.805904	1.206363
1BPI	**mGPfusion**	0.018036	3.731088
	2σ mGPfusion	0.076626	3.743321
	NO mGPfusion	0.042360	3.556351
	NO 2σ mGPfusion	0.033142	3.596703
1RGG	**mGPfusion**	0.643692	4.794329
	2σ mGPfusion	0.638577	4.820200
	NO mGPfusion	0.744636	3.589416
	NO 2σ mGPfusion	0.713534	3.566745
1RTB	**mGPfusion**	0.674564	2.116296
	2σ mGPfusion	0.692725	2.054498
	NO mGPfusion	0.647827	2.209865
	NO 2σ mGPfusion	0.652133	2.196515
2RN2	**mGPfusion**	0.713904	1.129753
	2σ mGPfusion	0.716456	1.104862
	NO mGPfusion	0.674966	1.125409
	NO 2σ mGPfusion	0.618044	1.197871
4LYZ	**mGPfusion**	0.295430	1.498755
	2σ mGPfusion	0.271164	1.526664
	NO mGPfusion	0.308895	1.489939
	NO 2σ mGPfusion	0.275442	1.525041
1FQG	**mGPfusion**	-0.153751	2.463955
	2σ mGPfusion	-	-
	NO mGPfusion	0.046418	2.501896
	NO 2σ mGPfusion	-	-

Table 7.2: Initial Benchmark of the methods overview.

The mGPfusion is the method as proposed in the paper [38] and the initial benchmark has been conducted on a subset of the original method benchmark proteins with data from the Protherm database. The 2σ prefix indicates that noise has been applied twice, as in the mGPfusion implementational code. The NO prefix stands for no optimization and indicates that no optimization routine was run, but MKL was used under default settings.

7.5 VAE Specifications

	Details	
Architecture	prior	standard Multivariate Normal N(0, 1)
	Encoder, p	FFNN with 1700 nodes, from one-hot encoded input
	latent representation, z	55
	Decoder	FFNN with 1200 nodes, added dropout-layer (p=0.065), and Categorical label-encoded output.
Optimization	Optimizer	Adam Optimizer
	learn-rate	0.000027
	weight-decay	0.0007
	batchsize	128
	epochs	200
	validation set	10%

Table 7.3: VAE architecture used for experiments with BLAT, UBQ, PGA.

7.6 The Vectorized Kernel

Listing 7.1: The vectorized Python kernel

```
@torch.no_grad()
def log_likelihood(self, x: torch.Tensor, i: int) -> np.array:
    """
    returns log likelihood of sequence at index i
    shape: N x 1
    internal shape: N sequences x n samples x AA dims
    """
    N, L = x.shape
    c = L*np.log(20)
    oh_x = self.convert_one_hot(x)
    # compute x and y likelihoods
    z_x_loc, z_x_scale = self.vae.encoder(oh_x)
    z = z_x_loc + self.latent_sample[:, np.newaxis] * torch.sqrt(z_x_scale)
    q_z_x = Normal(z_x_loc, z_x_scale).log_prob(z).to(torch.float64).sum(-1)
    p_z = torch.mean(self.p_z, axis=0)  # prior mean across samples
    # decoder can only evaluate one z at a time
    categoricals = [Categorical(self.vae.decoder(z_i).to(torch.float64).exp()) for z_i in z]
    p_x_z_vec = torch.stack([cat.probs.log().to(torch.float64) for cat in categoricals])
    p_x_z = torch.stack([cat.log_prob(torch.Tensor(x)).to(torch.float64) for cat in categoricals])
    p_x_not_i = self.p_x_not_i(p_x_z=p_x_z, p_z=p_z, q_z_x=q_z_x, c=c, i=i, L=L)
    ll_x_i_x_not_i = torch.mean(p_x_z_vec[:, :, i] + p_x_not_i[:, :, np.newaxis] - (q_z_x/L)[:, :, np.newaxis]
                                + (c/L) + (p_z/L), axis=0)
    return ll_x_i_x_not_i

@torch.no_grad()
def p_x_not_i(self, p_x_z: torch.Tensor, p_z: torch.Tensor, q_z_x: torch.Tensor, c: float, i: int, L: int) -> torch.Tensor:
    """
    As defined in equation, sum over log-likelihoods
    => p(X_1=x_1, X_2=x_2, ... ,X_L=x_L) not including X_i
    """
    p_x_not_i_lower_idx = torch.sum(p_x_z[:, :, :i] + (p_z/L) - (q_z_x/L)[:, :, np.newaxis] + (c/L), axis=-1)  # sum per sequence
    p_x_not_i_higher_idx = torch.sum(p_x_z[:, :, (i+1):] + (p_z/L) - (q_z_x/L)[:, :, np.newaxis] + (c/L), axis=-1)
    return p_x_not_i_lower_idx + p_x_not_i_higher_idx

@torch.no_grad()
def log_likelihood_idx(self, x: torch.Tensor, i: int):
    c = x.shape[1]*np.log(20)
    p_x = Categorical(self.vae.decoder(self.compute_encoder_dist(x).loc).exp()).log_prob(torch.Tensor(x))
    ll = self.log_likelihood(x, i).to(torch.float64) - c
    p_x_not_i = torch.log(torch.sum(torch.exp(ll), axis=-1))
    normalized_p_x_i_x_not_i = torch.diag(ll[:, x[:, i]]) - p_x[:, i] - p_x_not_i
    return normalized_p_x_i_x_not_i.detach().numpy()[:, np.newaxis]
```

7.7 S-Matrix Additions

7.7.1 Original Substitution Matrices

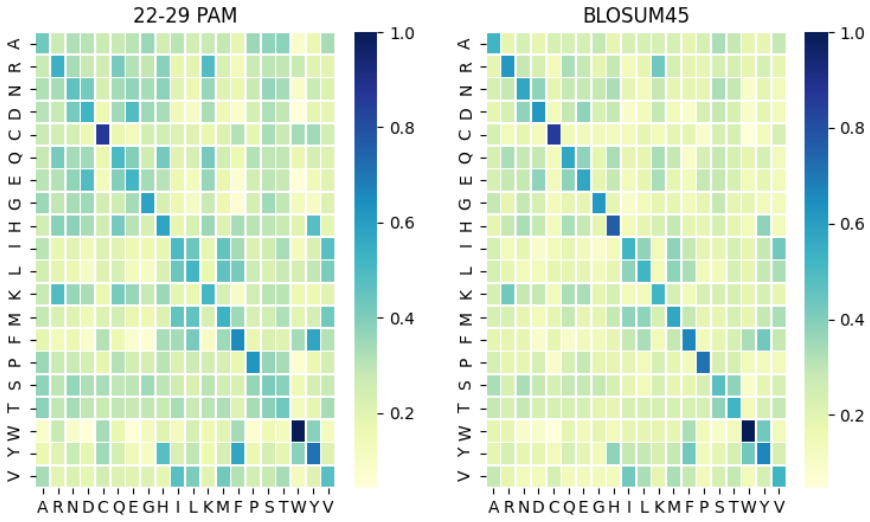


Figure 7.1: 22-29 PAM and BLOSUM45 matrices.
Original substitution matrices added for comparison against obtained matrices [29, 10].

7.7.2 Outer Product of Random Normal Vector

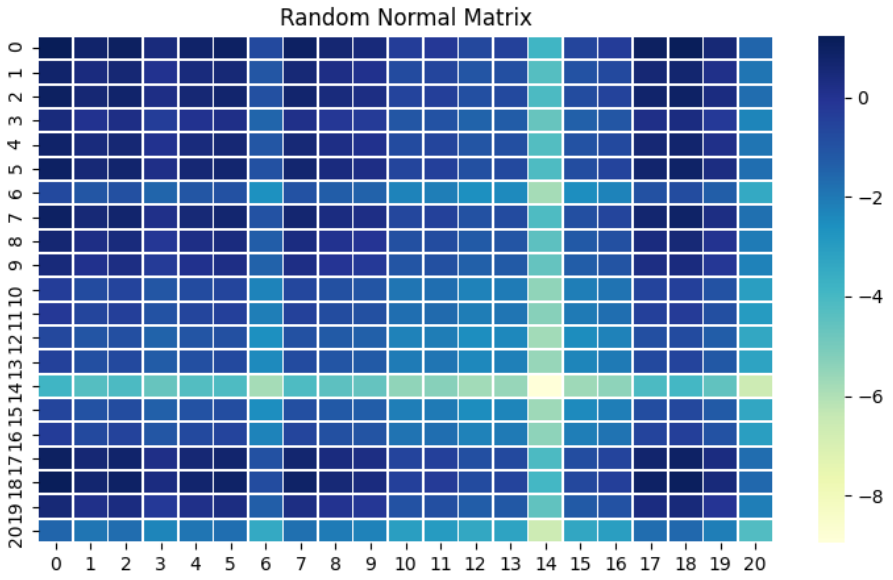


Figure 7.2: Matrix obtained from a random normal vector. The absolute value of a vector $\vec{v}$ of size 21 was taken, $\vec{v} = (v_1, v_2, ..., v_{21})^T$, with $v_i \sim \mathcal{N}(0, 1)$ and a matrix M obtained from the log of the outer-product $M = \log(\vec{v} \times \vec{v}^T)$. The matrix shows block-artifacts comparable to the ones in the S_{VAE} matrices.

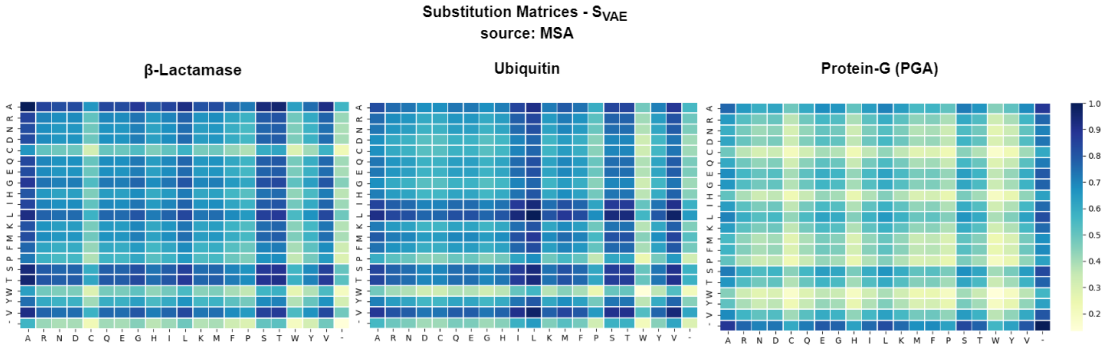
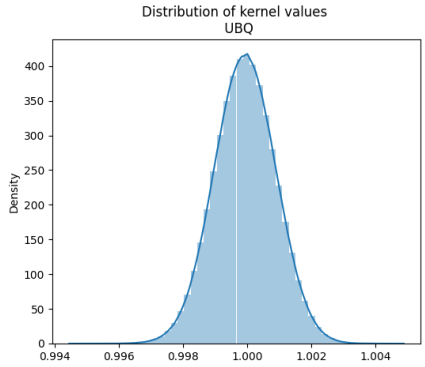
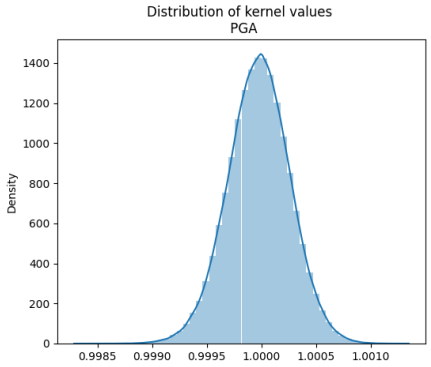


Figure 7.3: Substitution Matrix derived from MSA data.
Sequences from the multiple Sequence alignments, which were used for the VAE training. The S-Matrix equivalent was computed for a sub-selection (n=1000) for each protein from the normalized log-likelihoods. The structure in the results are consistent for BLAT and UBQ, but differ for PGA.

7.8 DES-Kernel Additions



(a) x are covariance function values for Ubiquitin variants.



(b) x are covariance function values for Protein-G variants.

Figure 7.4: Distribution over covariance function values.

7.9 Results over Positions

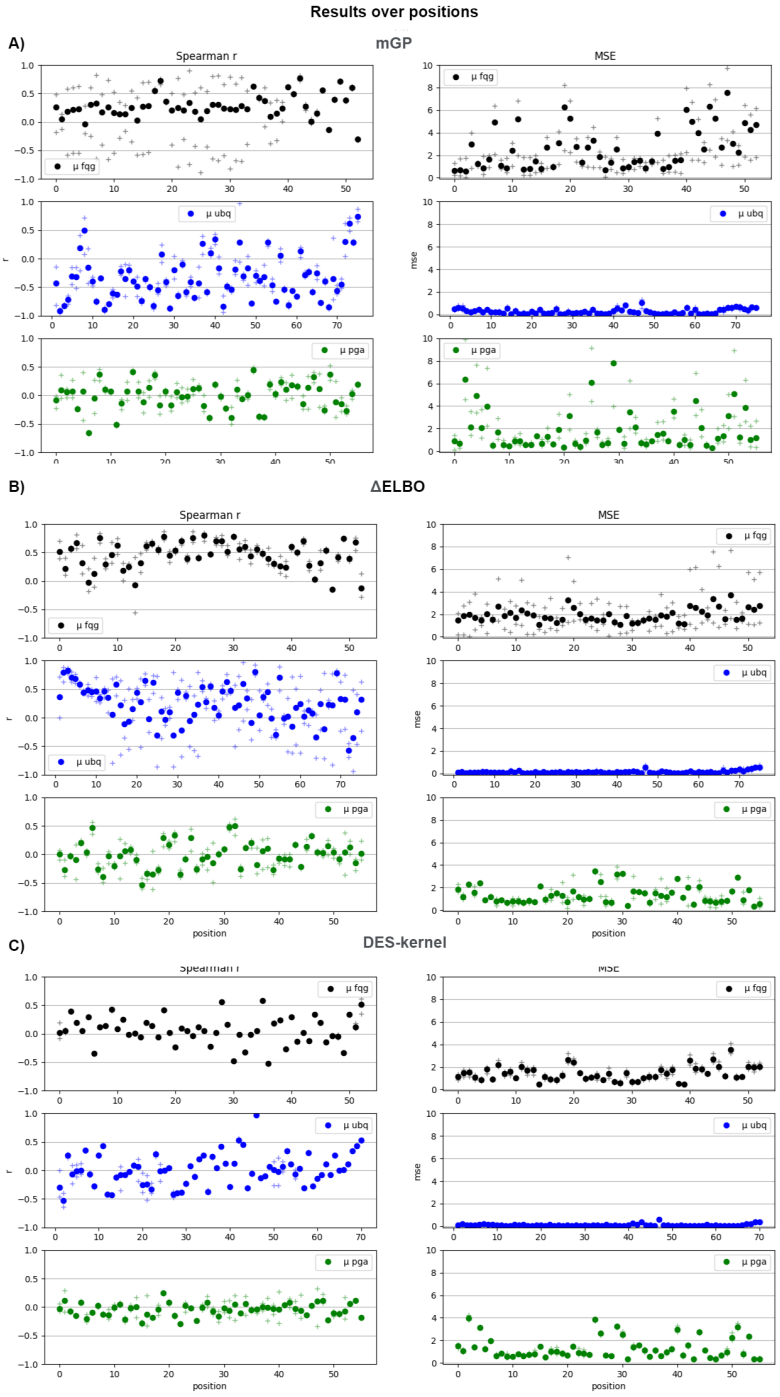


Figure 7.5: Results over positions.
Position level CV run with results per positions for the training subset of 25%, 50% and 100% were recorded. The mean values over the three runs are marked with bold dots, the min and max values of the results are marked as + over and under the mean per position. The results are reported for A) the mGP method, B) the added ΔELBO and C) the DES-kernel, with the left column for spearman-r correlation values and the right column mean-squared error.

7.10 Additional Benchmark Results

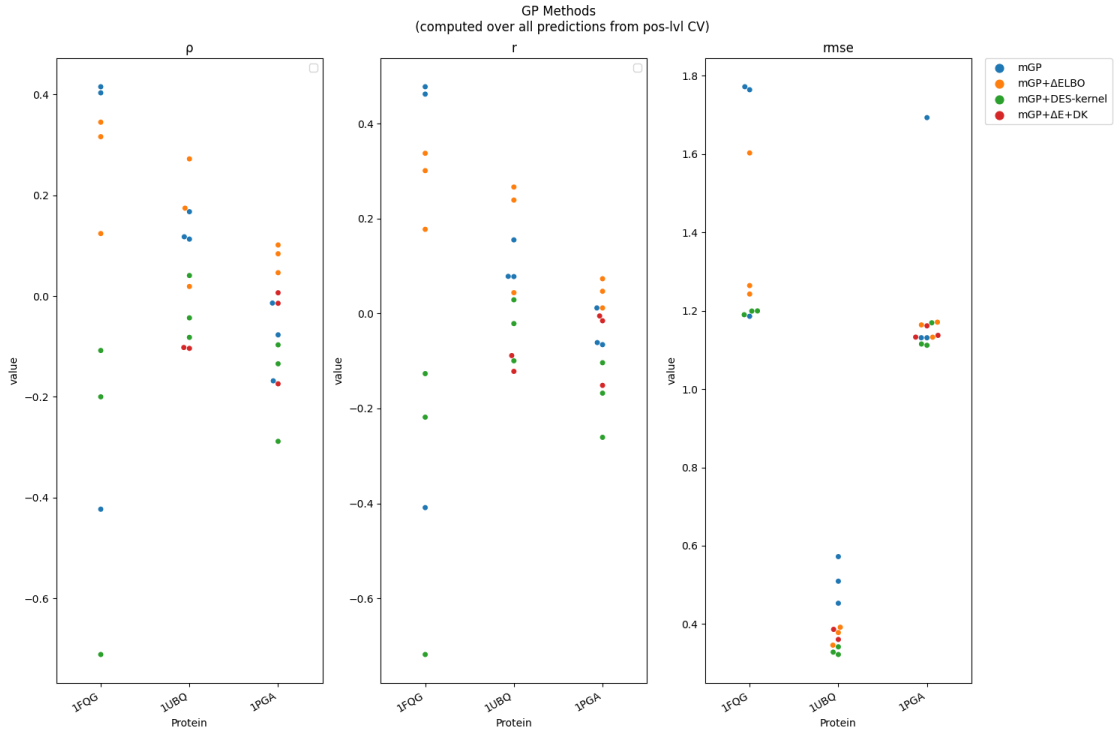


Figure 7.6: Overall predictions from 25%, 50%, 100% CV runs.
The total overall ρ, spearman r and root-mean-squared error has been computed from all predictions against all experimental observations. The runs were three different settings: sampling either 25%, 50% or 100% of training data from mutational variants per position-level cross-validation step for the mGP (blue), mGP+ΔELBO (orange), mGP+DES-kernel (green) and combination of ΔELBO and kernel (red) per protein. We observe that the values for 25% CV runs are generally smaller than 100%.

Protein	Method	ρ			rmse			r		
	training-set	25%	50%	100%	25%	50%	100%	25%	50%	100%
1FQG	mGP	-0.423003	0.043981	**0.403383**	**1.185628**	1.282796	1.763908	-0.408901	0.113288	**0.462604**
	mGP+ΔELBO	0.124101	0.345143	0.316285	1.264047	1.242453	1.602567	0.177602	0.301292	0.337882
	mGP+DES-kernel	-0.199953	-0.107982	-0.711408	1.189669	1.198959	1.199388	-0.218302	-0.126534	-0.718489
	mGP+ΔE+DK	-	-	-	-	-	-	-	-	-
1UBQ	mGP	0.113060	0.167425	0.117660	0.453242	0.509479	0.572218	0.078498	0.155322	0.078029
	mGP+ΔELBO	0.174466	0.019030	**0.272291**	0.346235	0.392037	0.378520	0.239243	0.044074	**0.266731**
	mGP+DES-kernel	0.040794	-0.043258	-0.081991	**0.322426**	0.328492	0.341986	0.029048	-0.021033	-0.099483
	mGP+ΔE+DK	-0.102048	-0.103794	-	0.360801	0.386595	-	-0.121733	-0.088492	-
1PGA	mGP	-0.077000	-0.013891	-0.168268	**1.130875**	1.130910	1.692729	-0.061082	0.011934	-0.065444
	mGP+ΔELBO	0.046406	0.083929	**0.101405**	1.132319	1.163715	1.170639	0.011909	0.047047	**0.073453**
	mGP+DES-kernel	-0.288344	-0.134398	-0.096823	1.114927	1.111660	1.169028	-0.260755	-0.167700	-0.103532
	mGP+ΔE+DK	-0.174126	-0.014341	0.006543	1.132509	1.137014	1.161280	-0.151279	-0.004945	-0.015113

Table 7.4: Benchmark of final overall results of the methods.

For the results, ρ, rmse, and spearman r were calculated from all predictions against true data, given the results for all positions from per CV run.

7.11 Individual Predictions over Positions

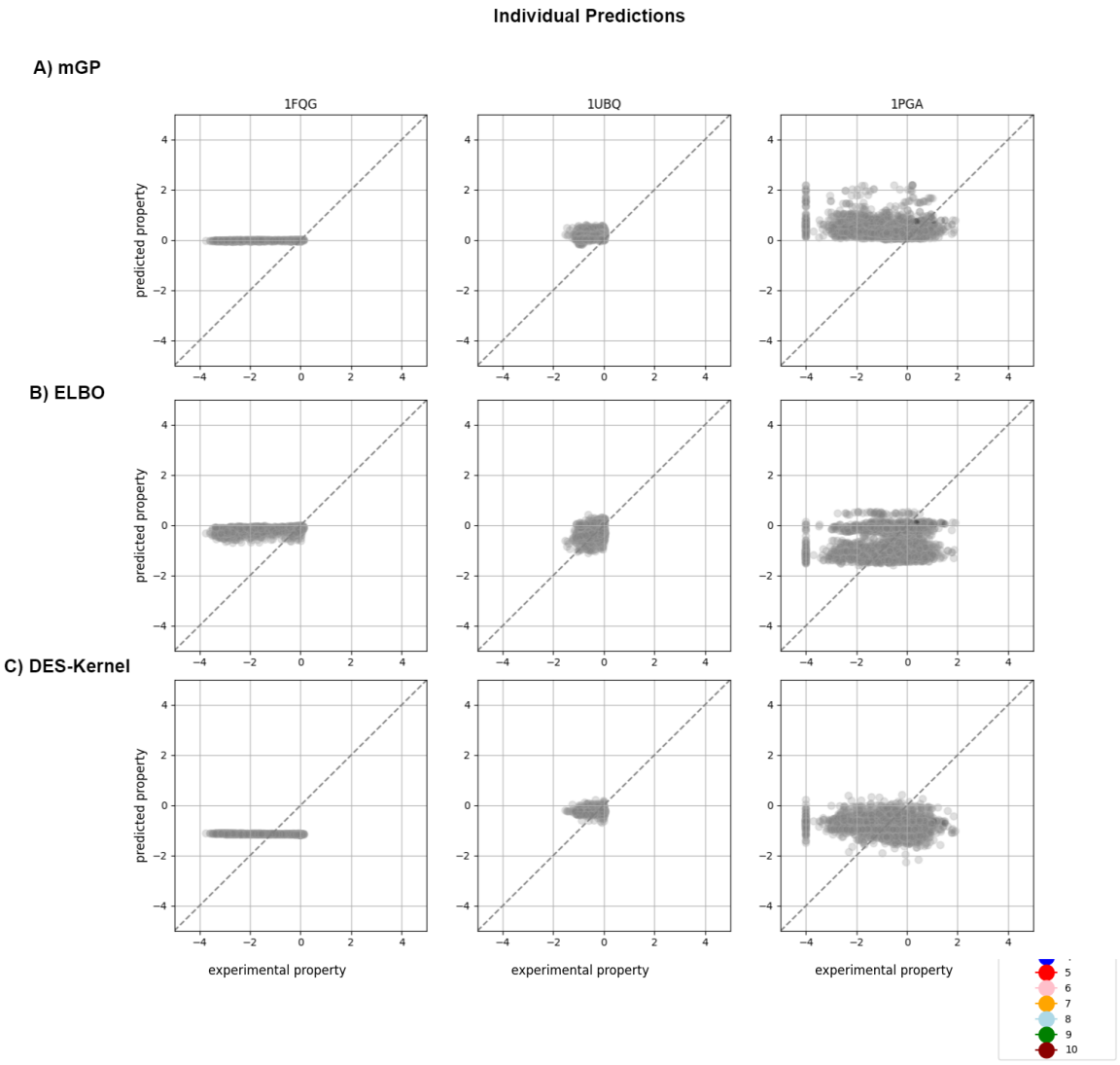


Figure 7.7: Individual Predictions per Protein.
Scatterplot of experimental observation (x-axis), against predicted values (y-axis) for the individual proteins for each method. The rows are the individual methods with A) mGP, B) mGP+ΔELBO and C) mGP+DES-kernel. The majority of mutations are single-site mutations (grey), with an exception of PGA, with few double mutations.

Method	pdb	training	spearman r	mse
mGP	1FQG	0.25	-0.164328	**1.408669**
		0.5	0.191892	1.651183
		1.0	**0.493366**	3.132168
	1PGA	0.25	-0.018582	1.297084
		0.5	-0.002104	**1.292068**
		1.0	**0.016465**	2.812843
	1UBQ	0.25	-0.355621	**0.221921**
		0.5	-0.361438	0.273851
		1.0	**-0.347524**	0.340269
mGP+DES-kernel	1FQG	0.25	0.067843	1.418381
		0.5	0.062158	1.440779
		1.0	0.056879	1.444897
	1PGA	0.25	-0.037587	1.271349
		0.5	-0.056986	1.262362
		1.0	-0.042197	1.381225
	1UBQ	0.25	-0.013916	0.104495
		0.5	0.031879	0.109558
		1.0	0.028433	0.116933
mGP+ΔELBO	1FQG	0.25	0.426837	1.595190
		0.5	0.428250	1.541967
		1.0	**0.489492**	2.590010
	1PGA	0.25	-0.008579	1.330133
		0.5	-0.020058	1.374236
		1.0	-0.009393	1.389042
	1UBQ	0.25	0.250143	0.125341
		0.5	0.195530	0.157171
		1.0	**0.281593**	0.152261
mGP+ΔELBO+DES-kernel	1PGA	0.25	0.013749	1.325455
		0.5	-0.009021	1.340319
		1.0	0.002592	1.394520
	1UBQ	0.25	-0.076411	0.132070
		0.5	-0.011264	0.149212

Table 7.5: Benchmark of results over position-wise analysis.

Per protein, per method the mean over the individual positions was taken, with respect to the training data-split. Where 25%, 50%, 100% describes the amount of uniformly sub-sampled training-data to conduct the position level CV run.

Bibliography

[1] Guillaume Alain and Yoshua Bengio. „What regularized auto-encoders learn from the data-generating distribution“. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 3563–3593.

[2] D Altschuh, T Vernet, P Berti, D Moras, and K Nagai. „Coordinated amino acid changes in homologous protein families“. In: *Protein Engineering, Design and Selection* 2.3 (1988), pp. 193–199.

[3] Carlos L Araya, Douglas M Fowler, Wentao Chen, Ike Muniez, Jeffery W Kelly, and Stanley Fields. „A fundamental protein property, thermodynamic stability, revealed solely from large-scale measurements of protein function“. In: *Proceedings of the National Academy of Sciences* 109.42 (2012), pp. 16858–16863.

[4] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[5] Jeremy M Berg, John L Tymoczko, Lubert Stryer, *et al. Biochemistry*. 2002.

[6] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. „Pyro: Deep universal probabilistic programming“. In: *The Journal of Machine Learning Research* 20.1 (2019), pp. 973–978.

[7] Jeffrey I Boucher, Daniel NA Bolon, and Dan S Tawfik. „Quantifying and understanding the fitness effects of protein mutations: Laboratory versus nature“. In: *Protein Science* 25.7 (2016), pp. 1219–1226.

[8] Peter JA Cock, Tiago Antao, Jeffrey T Chang, Brad A Chapman, Cymon J Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, *et al.* „Biopython: freely available Python tools for computational molecular biology and bioinformatics“. In: *Bioinformatics* 25.11 (2009), pp. 1422–1423.

[9] Gavin E Crooks and Steven E Brenner. „An alternative model of amino acid replacement“. In: *Bioinformatics* 21.7 (2005), pp. 975–980.

[10] M Cserzo, JM Bernassau, I Simon, and B Maigret. „New alignment strategy for transmembrane proteins“. In: *Journal of molecular biology* 243.3 (1994), pp. 388–396.

[11] Bassil I Dahiyat and Stephen L Mayo. „De novo protein design: fully automated sequence selection“. In: *Science* 278.5335 (1997), pp. 82–87.

[12] Gautam Dantas, Colin Corrent, Steve L Reichow, James J Havranek, Ziad M Eletr, Nancy G Isern, Brian Kuhlman, Gabriele Varani, Ethan A Merritt, and David Baker. „High-resolution structural and thermodynamic analysis of extreme stabilization of human procarboxypeptidase by computational protein design“. In: *Journal of molecular biology* 366.4 (2007), pp. 1209–1221.

[13] Payel Das, Tom Sercu, Kahini Wadhawan, Inkit Padhi, Sebastian Gehrmann, Flaviu Cipcigan, Vijil Chenthamarakshan, Hendrik Strobelt, Cicero Dos Santos, Pin-Yu Chen, *et al.* „Accelerated antimicrobial discovery via deep generative models and molecular dynamics simulations“. In: *Nature Biomedical Engineering* (2021), pp. 1–11.

[14] Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. „Hyperspherical variational auto-encoders“. In: *arXiv preprint arXiv:1804.00891* (2018).

[15] Yves Dehouck, Jean Marc Kwasigroch, Dimitri Gilis, and Marianne Rooman. „PoPMuSiC 2.1: a web server for the estimation of protein stability changes upon mutation and sequence optimality“. In: *BMC bioinformatics* 12.1 (2011), pp. 1–12.

[16] Nicki Skafte Detlefsen, Søren Hauberg, and Wouter Boomsma. „What is a meaningful representation of protein sequences?“ In: *arXiv preprint arXiv:2012.02679* (2020).

[17] Frank DiMaio, Andrew Leaver-Fay, Phil Bradley, David Baker, and Ingemar André. „Modeling symmetric macromolecular structures in Rosetta3“. In: *PloS one* 6.6 (2011), e20450.

[18] Xinqiang Ding, Zhengting Zou, and Charles L Brooks III. „Deciphering protein evolution and fitness landscapes with latent space models“. In: *Nature communications* 10.1 (2019), pp. 1–13.

[19] Richard Durbin, Sean R Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[20] Sean R Eddy. „Accelerated profile HMM searches“. In: *PLoS Comput Biol* 7.10 (2011), e1002195.

[21] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rihawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Debsindhu Bhowmik, *et al.* „ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Deep Learning and High Performance Computing“. In: *arXiv preprint arXiv:2007.06225* (2020).

[22] A Elisabeth Eriksson, Walter A Baase, Xue-Jun Zhang, Dirk W Heinz, MPBE Blaber, Enoch P Baldwin, and Brian W Matthews. „Response of a protein structure to cavity-creating mutations and its relation to the hydrophobic effect“. In: *Science* 255.5041 (1992), pp. 178–183.

[23] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamın Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. „Automatic chemical design using a data-driven continuous representation of molecules“. In: *ACS central science* 4.2 (2018), pp. 268–276.

[24] Jan W. Gooch. „Gibbs Free Energy“. In: *Encyclopedic Dictionary of Polymers*. Ed. by Jan W. Gooch. New York, NY: Springer New York, 2011, pp. 895–895.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. „Deep learning book“. In: *MIT Press* 521.7553 (2016), p. 800.

[26] Raphael Guerois, Jens Erik Nielsen, and Luis Serrano. „Predicting changes in the stability of proteins and protein complexes: a study of more than 1000 mutations“. In: *Journal of molecular biology* 320.2 (2002), pp. 369–387.

[27] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, *et al.* „Array programming with NumPy“. In: *Nature* 585.7825 (2020), pp. 357–362.

[28] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[29] Steven Henikoff and Jorja G Henikoff. „Amino acid substitution matrices from protein blocks“. In: *Proceedings of the National Academy of Sciences* 89.22 (1992), pp. 10915–10919.

[30] Ralf Herbrich. *Learning kernel classifiers: theory and algorithms*. MIT press, 2001.

[31] Matthew D Hoffman and Andrew Gelman. „The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo.“ In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1593–1623.

[32] 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“. In: *Nature biotechnology* 35.2 (2017), pp. 128–135.

[33] Po-Ssu Huang, Scott E Boyken, and David Baker. „The coming of age of de novo protein design“. In: *Nature* 537.7620 (2016), pp. 320–327.

[34] John D Hunter. „Matplotlib: A 2D graphics environment“. In: *IEEE Annals of the History of Computing* 9.03 (2007), pp. 90–95.

[35] Marcus Jäger, Yan Zhang, Jan Bieschke, Houbi Nguyen, Maria Dendle, Marianne E Bowman, Joseph P Noel, Martin Gruebele, and Jeffery W Kelly. „Structure–function–folding relationship in a WW domain“. In: *Proceedings of the National Academy of Sciences* 103.28 (2006), pp. 10648–10653.

[36] Johan Ludwig William Valdemar Jensen *et al.* „Sur les fonctions convexes et les inégalités entre les valeurs moyennes“. In: *Acta mathematica* 30 (1906), pp. 175–193.

[37] Xin Jiang, Jennifer Kowalski, and Jeffery W Kelly. „Increasing protein stability using a rational approach combining sequence homology and structural alignment: Stabilizing the WW domain“. In: *Protein Science* 10.7 (2001), pp. 1454–1465.

[38] Emmi Jokinen, Markus Heinonen, and Harri Lähdesmäki. „mGPfusion: predicting protein stability changes with Gaussian process kernel learning and data fusion“. In: *Bioinformatics* 34.13 (2018), pp. i274–i283.

[39] David T Jones. „Setting the standards for machine learning in biology“. In: *Nature Reviews Molecular Cell Biology* 20.11 (2019), pp. 659–660.

[40] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. „Gaussian processes and kernel methods: A review on connections and equivalences“. In: *arXiv preprint arXiv:1807.02582* (2018).

[41] Sofia Khan and Mauno Vihinen. „Performance of protein stability predictors“. In: *Human mutation* 31.6 (2010), pp. 675–684.

[42] Diederik P Kingma and Jimmy Ba. „Adam: A method for stochastic optimization“. In: *arXiv preprint arXiv:1412.6980* (2014).

[43] Diederik P Kingma and Max Welling. „An introduction to variational autoencoders“. In: *arXiv preprint arXiv:1906.02691* (2019).

[44] Diederik P Kingma and Max Welling. „Auto-encoding variational bayes“. In: *arXiv preprint arXiv:1312.6114* (2013).

[45] Christian Kramer and Peter Gedeck. „Leave-cluster-out cross-validation is appropriate for scoring functions derived from diverse protein data sets“. In: *Journal of chemical information and modeling* 50.11 (2010), pp. 1961–1969.

[46] Solomon Kullback and Richard A Leibler. „On information and sufficiency“. In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[47] MD Shaji Kumar, K Abdulla Bava, M Michael Gromiha, Ponraj Prabakaran, Koji Kitajima, Hatsuho Uedaira, and Akinori Sarai. „ProTherm and ProNIT: thermodynamic databases for proteins and protein–nucleic acid interactions“. In: *Nucleic acids research* 34.suppl_1 (2006), pp. D204–D206.

[48] Benjamin Leader, Quentin J Baca, and David E Golan. „Protein therapeutics: a summary and pharmacological classification“. In: *Nature reviews Drug discovery* 7.1 (2008), pp. 21–39.

[49] Kresten Lindorff-Larsen, Stefano Piana, Ron O Dror, and David E Shaw. „How fast-folding proteins fold“. In: *Science* 334.6055 (2011), pp. 517–520.

[50] Benjamin J Livesey and Joseph A Marsh. „Using deep mutational scanning to benchmark variant effect predictors and identify disease mutations“. In: *Molecular systems biology* 16.7 (2020), e9380.

[51] Ujjwal Maulik, Sanghamitra Bandyopadhyay, and Jason T Wang. *Computational intelligence and pattern analysis in biology informatics*. Vol. 20. Wiley Online Library, 2011.

[52] Mitchell McIntire, Daniel Ratner, and Stefano Ermon. „Sparse Gaussian Processes for Bayesian Optimization.“ In: *UAI*. 2016.

[53] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. „Equation of state calculations by fast computing machines“. In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[54] Tobias Müller, Sven Rahmann, and Marc Rehmsmeier. „Non-symmetric score matrices and the detection of homologous transmembrane proteins“. In: *Bioinformatics* 17.1 (2001), S182–S189.

[55] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[56] Arkadi S Nemirovski and David Berkovich Yudin. „Cesari convergence of the gradient method of approximating saddle points of convex-concave functions“. In: *Doklady Akademii Nauk*. Vol. 239. 5. Russian Academy of Sciences. 1978, pp. 1056–1059.

[57] Alex Nisthal, Connie Y Wang, Marie L Ary, and Stephen L Mayo. „Protein stability engineering insights revealed by domain-wide comprehensive mutagenesis“. In: *Proceedings of the National Academy of Sciences* 116.33 (2019), pp. 16367–16377.

[58] Art B. Owen. *Monte Carlo theory, methods and examples*. Stanford University, 2013.

[59] Gábor Pál, Jean-Louis K Kouadio, Dean R Artis, Anthony A Kossiakoff, and Sachdev S Sidhu. „Comprehensive and quantitative mapping of energy landscapes for protein-protein interactions by rapid combinatorial scanning“. In: *Journal of Biological Chemistry* 281.31 (2006), pp. 22378–22385.

[60] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. „Automatic differentiation in pytorch“. In: (2017).

[61] Stefano Piana, Kresten Lindorff-Larsen, and David E Shaw. „Protein folding kinetics and thermodynamics from atomistic simulation“. In: *Proceedings of the National Academy of Sciences* 109.44 (2012), pp. 17845–17850.

[62] Vladimir Potapov, Mati Cohen, and Gideon Schreiber. „Assessing computational methods for predicting protein stability upon mutation: good on average but not in the details“. In: *Protein engineering, design & selection* 22.9 (2009), pp. 553–560.

[63] Chun-xu Qu, Lu-hua Lai, Xiao-jie Xu, and You-qi Tang. „Phyletic relationships of protein structures based on spatial preference of residues“. In: *Journal of molecular evolution* 36.1 (1993), pp. 67–78.

[64] Joaquin Quinonero-Candela and Carl Edward Rasmussen. „A unifying view of sparse approximate Gaussian process regression“. In: *The Journal of Machine Learning Research* 6 (2005), pp. 1939–1959.

[65] JK Mohana Rao. „New scoring matrix for amino acid residue exchanges based on residue characteristic physical parameters“. In: *International journal of peptide and protein research* 29.2 (1987), pp. 276–281.

[66] Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John Canny, Pieter Abbeel, and Yun S Song. „Evaluating protein transfer learning with tape“. In: *Advances in Neural Information Processing Systems* 32 (2019), p. 9689.

[67] Michael Remmert, Andreas Biegert, Andreas Hauser, and Johannes Söding. „HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment“. In: *Nature methods* 9.2 (2012), pp. 173–175.

[68] Adam J Riesselman, John B Ingraham, and Debora S Marks. „Deep generative models of genetic variation capture the effects of mutations“. In: *Nature methods* 15.10 (2018), pp. 816–822.

[69] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, *et al.* „Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences“. In: *Proceedings of the National Academy of Sciences* 118.15 (2021).

[70] Carol A Rohl, Charlie EM Strauss, Kira MS Misura, and David Baker. „Protein structure prediction using Rosetta“. In: *Methods in enzymology* 383 (2004), pp. 66–93.

[71] Philip A Romero, Andreas Krause, and Frances H Arnold. „Navigating the protein fitness landscape with Gaussian processes“. In: *Proceedings of the National Academy of Sciences* 110.3 (2013), E193–E201.

[72] Benjamin P Roscoe, Kelly M Thayer, Konstantin B Zeldovich, David Fushman, and Daniel NA Bolon. „Analyses of the effects of all ubiquitin point mutants on yeast growth rate“. In: *Journal of molecular biology* 425.8 (2013), pp. 1363–1377.

[73] Tim Salimans, Diederik Kingma, and Max Welling. „Markov chain monte carlo and variational inference: Bridging the gap“. In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1218–1226.

[74] John A Schellman. „The thermodynamic stability of proteins“. In: *Annual review of biophysics and biophysical chemistry* 16.1 (1987), pp. 115–137.

[75] Robert Sheridan, Robert J Fieldhouse, Sikander Hayat, Yichao Sun, Yevgeniy Antipin, Li Yang, Thomas Hopf, Debora S Marks, and Chris Sander. „EVfold. org: evolutionary couplings and protein 3D structure prediction“. In: *biorxiv* (2015), p. 021022.

[76] Jung-Eun Shin, Adam J Riesselman, Aaron W Kollasch, Conor McMahon, Elana Simon, Chris Sander, Aashish Manglik, Andrew C Kruse, and Debora S Marks. „Protein design and variant prediction using autoregressive generative models“. In: *Nature communications* 12.1 (2021), pp. 1–11.

[77] Kim T Simons, Rich Bonneau, Ingo Ruczinski, and David Baker. „Ab initio protein structure prediction of CASP III targets using ROSETTA“. In: *Proteins: Structure, Function, and Bioinformatics* 37.S3 (1999), pp. 171–176.

[78] Kim T Simons, Charles Kooperberg, Enoch Huang, and David Baker. „Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions“. In: *Journal of molecular biology* 268.1 (1997), pp. 209–225.

[79] C. Spearman. „The Proof and Measurement of Association between Two Things“. In: *The American Journal of Psychology* 100.3/4 (1987), pp. 441–471.

[80] Michael A Stiffler, Doeke R Hekstra, and Rama Ranganathan. „Evolvability as a function of purifying selection in TEM-1 β-lactamase“. In: *Cell* 160.5 (2015), pp. 882–892.

[81] Baris E Suzek, Yuqi Wang, Hongzhan Huang, Peter B McGarvey, Cathy H Wu, and UniProt Consortium. „UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches“. In: *Bioinformatics* 31.6 (2015), pp. 926–932.

[82] Julie D Thompson, Desmond G Higgins, and Toby J Gibson. „CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice“. In: *Nucleic acids research* 22.22 (1994), pp. 4673–4680.

[83] Kentaro Tomii and Minoru Kanehisa. „Analysis of amino acid indices and mutation matrices for sequence comparison and structure prediction of proteins“. In: *Protein Engineering, Design and Selection* 9.1 (1996), pp. 27–36.

[84] Connie Y Wang, Paul M Chang, Marie L Ary, Benjamin D Allen, Roberto A Chica, Stephen L Mayo, and Barry D Olafson. „ProtaBank: A repository for protein design and engineering data“. In: *Protein Science* 27.6 (2018), pp. 1113–1124.

[85] Michael L. Waskom. „seaborn: statistical data visualization“. In: *Journal of Open Source Software* 6.60 (2021), p. 3021.

[86] Eli N Weinstein and Debora S Marks. „A structured observation distribution for generative biological sequence prediction and forecasting“. In: *bioRxiv* (2021), pp. 2020–07.

[87] Gregory A Weiss, Colin K Watanabe, Alan Zhong, Audrey Goddard, and Sachdev S Sidhu. „Rapid mapping of protein functional epitopes by combinatorial alanine scanning“. In: *Proceedings of the National Academy of Sciences* 97.16 (2000), pp. 8950–8954.

[88] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.

[89] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. „Deep kernel learning“. In: *Artificial intelligence and statistics*. PMLR. 2016, pp. 370–378.

[90] Kevin K Yang, Zachary Wu, and Frances H Arnold. „Machine-learning-guided directed evolution for protein engineering“. In: *Nature methods* 16.8 (2019), pp. 687–694.

[91] Charles Yanofsky, Bruce C Carlton, John R Guest, Don R Helinski, and Ulf Henning. „On the colinearity of gene structure and protein structure“. In: *Proceedings of the National Academy of Sciences of the United States of America* 51.2 (1964), p. 266.

[92] Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, *et al.* „Accelerating the Machine Learning Lifecycle with MLflow.“ In: *IEEE Data Eng. Bull.* 41.4 (2018), pp. 39–45.

[93] Lukas Zimmermann, Andrew Stephens, Seung-Zin Nam, David Rau, Jonas Kübler, Marko Lozajic, Felix Gabler, Johannes Söding, Andrei N Lupas, and Vikram Alva. „A completely reimplemented MPI bioinformatics toolkit with a new HHpred server at its core“. In: *Journal of molecular biology* 430.15 (2018), pp. 2237–2243.

[94] Daniel Zwillinger and Stephen Kokoska. *CRC standard probability and statistics tables and formulae*. Crc Press, 1999.