Exploring the latent space of deep generative protein models: Applications to GPCRs



Candidate no. 1047884

Word count: 10884 words

Submitted in partial completion of the

Master of Science, Computer Science

Trinity Term, August 2021

Abstract

Deep generative models have seen wide success in the fields of computer vision, natural language processing and speech processing, given their ability to learn or sample from complex data distributions. Recently, they have been applied to model protein sequence data, achieving state-of-the-art performance in protein fitness prediction, contact prediction, and structure prediction. However, tools for analysing the rich representations these models are learning remain limited. We apply a deep generative model to a widely-studied and clinically relevant family of proteins, the G protein-coupled receptors (GPCRs), and analyse the learnt latent representation. We present a clustering analysis to uncover how these proteins are grouped in latent space, and propose ways in which the clustering can be used to identify previously uncharacterised 'orphan' proteins. Thereafter, we augment our model architecture to predict biological properties of interest by incorporating labelled data, and discuss emperical difficulties when training on such a small proportion of labels (0.1% of the dataset is labelled). Finally, we propose ways in which this semi-supervised architecture can be used for protein design and which biological questions it might answer, particularly in the context of GPCRs.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Contributions 4

1.3 Dissertation outline 4

2 Background and Related Work **6**

2.1 Biology 7

2.1.1 Proteins 7

2.1.2 Multiple Sequence Alignment (MSA) 9

2.1.3 G protein-coupled receptors (GPCRs) 9

2.2 Machine Learning 11

2.2.1 Machine learning definitions 11

2.2.2 Neural Networks 13

2.2.3 Variational Autoencoders (VAE) 16

2.2.4 Bayesian VAE 20

2.3 Machine Learning for Proteins 21

2.3.1 Unsupervised protein sequence models 21

2.3.2 Evolutionary model of Variant Effects (EVE) 23

2.3.3 Interpreting protein representations 27

3 Uncovering GPCR functional families **29**

3.1 Objectives 29

3.2 Methods 30

3.3 Results and discussion 31

4 Signaling activation prediction **34**

4.1 Objectives 34

4.2 Methods 35

4.3 Results and discussion 37

5 Reflections and future directions **41**
5.1 Technical discussion 42
5.2 Future work 43
5.2.1 Methodological improvements 43
5.2.2 Deorphanisation 43
5.2.3 Going further with semi-supervision 44
5.2.4 Global vs local properties of latent space 44
5.3 Personal development 44

6 Conclusion **46**
6.1 Summary 46
6.2 Acknowledgements 47

References **48**

Appendices

A EVE model **56**
A.1 EVE score 56
A.2 EVE architecture 56
A.2.1 1x1 convolution 58
A.2.2 Input/output shapes 58
A.3 Results 59

1 Introduction

1.1 Motivation

Proteins are the essential building blocks of life - they are responsible for a wide array of critical functions in all living organisms. From enzymes that catalyse reactions in the cell, to hormones like adrenaline (epinephrine) and the receptors that recognise them, to the RNA polymerase that transcribes information from DNA into RNA. A better understanding of proteins could allow for more effective drug design, disease prediction and therapy, and understanding antibiotic resistance [1].

With the advent of high-throughput sequencing techniques (such as Next Generation Sequencing for DNA sequences), the cost of sequencing DNA and proteins has fallen dramatically and protein sequence datasets have grown to the order of hundreds of millions of sequences [2, 3].

However, labelled protein sequence data has not grown at the same rate. For example, there are only ∼160 000 protein structures available, compared to more than 280 million known protein sequences [4].

This motivates the use of unsupervised machine learning methods that can leverage unlabelled data to learn meaningful **representations** of these proteins for tasks such as structure and function prediction and protein/drug design.

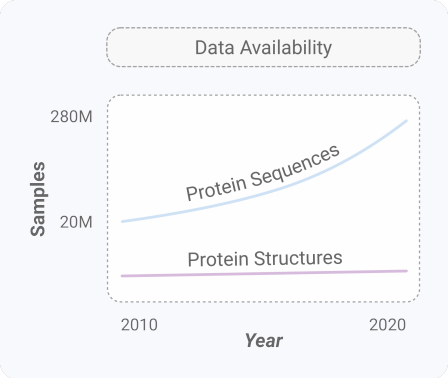


Figure 1.1: The rapid growth of protein sequence datasets is outpacing the relatively slow growth of labelled data (such as protein structure) [5].

A particular family of proteins, the G protein-coupled receptors (GPCRs), are among the most studied targets in drug design. According to Hauser et al. [6], 34 percent of all drugs approved by the US Food and Drug Administration (FDA) target GPCRs. Against this backdrop, "of the 360 non-sensory GPCR genes encoded in the human genome, one-third are still labelled as orphans to reflect the fact that either their ligands and/or signaling are unknown" [7]. This ligand identification problem is an active area of research [8].

Deep generative models such as Variational Autoencoders (VAEs) [9, 10], Generative Adversarial Networks [11] and autoregressive models such as BERT [12] using Transformers [13], have had a profound impact on the fields in computer vision, natural language processing, and speech processing, given their ability to learn or sample from complex data distributions.

Recently, they have been applied to model protein sequence data [14–20], achieving state-of-the-art performance in protein fitness prediction, contact prediction, or structure prediction. However, how these deep generative models represent protein sequences is not well known.

A better understanding of the representations of these deep generative protein models could allow for new applications (such as more targeted protein design), and accordingly, a better understanding of GPCRs could yield new insights (such as functional relationships between distantly related proteins, or the identification of ligands).

We highlight our recent results in advancing the understanding of both GPCRs and such deep generative models trained on proteins.

1.2 Contributions

- We build on prior deep generative protein models and apply them to the GPCR protein family.
- We present a clustering analysis to uncover how these proteins are grouped in latent space, and propose ways that this can be used to identify previously uncharacterised 'orphan' proteins.
- We augment our model architecture to predict biological properties of interest by incorporating labelled data, and discuss empirical difficulties with this approach when considering a small proportion of labels (0.1% of the dataset is labelled).
- We propose ways in which this semi-supervised architecture can be used for protein design and which biological questions it might answer.

1.3 Dissertation outline

In Chapter 2, we introduce some key concepts in biology and machine learning necessary to appreciate the methods of this work. Section 2.3 introduces some relevant prior protein representation learning models, in particular the Evolutionary model of Variant Effects (EVE) that we use in this work.

We then move on to the two main methods that form our contribution. Chapter 3 describes our clustering analysis in full, where we cluster the latent representations of 150 000 proteins in the GPCR family. Chapter 4 introduces the property prediction experiment, where we train a semi-supervised model to predict the signaling activation level of different proteins where only 0.1% of the dataset is labelled.

Thereafter, in Chapter 5, we discuss the implications of the results from these two experiments and their clinical relevance. We also present approaches that did not work as expected, discuss why they failed, how we adjusted our approach accordingly, and the future research opportunities they present. We provide some

final reflections on personal development from the dissertation experience before concluding with a summary in Chapter 6.

2

Background and Related Work

Contents

2.1 Biology 7
2.1.1 Proteins 7
2.1.2 Multiple Sequence Alignment (MSA) 9
2.1.3 G protein-coupled receptors (GPCRs) 9
2.2 Machine Learning 11
2.2.1 Machine learning definitions 11
2.2.2 Neural Networks 13
2.2.3 Variational Autoencoders (VAE) 16
2.2.4 Bayesian VAE 20
2.3 Machine Learning for Proteins 21
2.3.1 Unsupervised protein sequence models 21
2.3.2 Evolutionary model of Variant Effects (EVE) 23
2.3.3 Interpreting protein representations 27

We will begin by introducing the concept of proteins, and how we construct datasets of evolutionarily related protein sequences using sequence alignment. We introduce a clinically relevant and diverse (super)family of proteins, the G protein-coupled receptors (GPCRs), and describe its mechanism for cell signaling. Thereafter, in the machine learning section we cover different learning paradigms and introduce variational autoencoders (VAEs), the class of models used in this work. We end by discussing some prior work on unsupervised protein sequence

modelling, and describe the Evolutionary model of Variant Effect, a specific protein VAE architecture that we extend.

2.1 Biology

2.1.1 Proteins

Proteins consist of chains of amino acids held together by covalent bonds. These chains then 'fold' together to form stable three-dimensional structures.

Proteins can be described in terms of their *primary*, *secondary*, *tertiary*, or *quaternary* structure. A protein's primary structure is simply the sequence of amino acids. The secondary structure describes local folded three-dimensional structures of the amino acid chain. The tertiary structure is the complete folded three-dimensional structure of the entire protein, which is determined by the interactions between the amino acid side chains. The quaternary structure is when different protein molecules combine to form **subunits** of a bigger three-dimensional structure, called a **protein complex**.

Proteins are assembled from amino acids using information from an organism's genome. These proteins, therefore, undergo evolution through natural selection of the organism, and the diversity of proteins across organisms reflects billions of evolutionary 'experiments'.

This evolutionary information can be used to infer functional or structural properties of a protein. Gobel (1994) [23] hypothesized that amino acids in contact would tend to evolve together to preserve function: "if one residue mutates, its contact partners also tend to mutate". This evolutionary information and the co-evolution of different amino acid positions (**residues**) of the protein sequence are at the heart of what we consider in this dissertation.

A **protein family** is a group of proteins that "share a common evolutionary origin, reflected by their related functions and similarities in sequence and structure" [24]. This evolutionary relationship among biological entities (in our case, proteins) is called **phylogeny**.

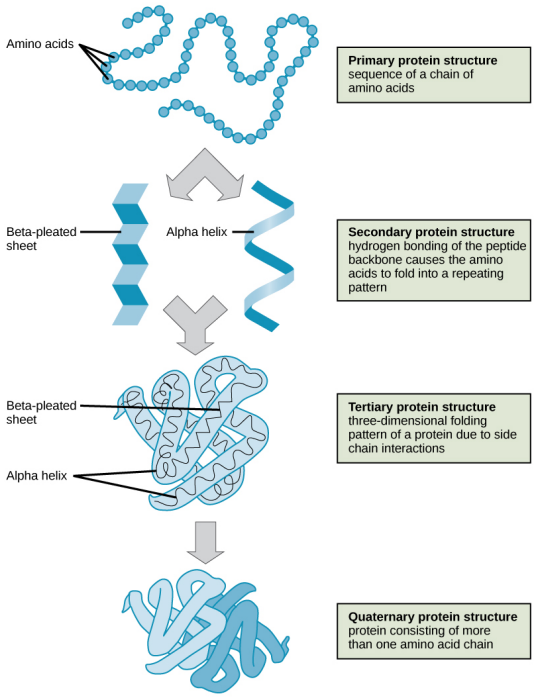


Figure 2.1: Illustration of different orders of protein structure, from [21] under the CC BY 3.0 licence [22].

Cell signaling is the "process by which cells communicate with their environment and respond temporally to external cues that they sense there" [25]. The membrane receptors we examine in this work recognise extracellular signal molecules (**ligands**) and bind them - the ligand and receptor form a complex. The **receptor** will have a certain structure that only allows it to bind with specific ligands. This binding "leads to a change in receptor conformation [its structure changes] and thence to intracellular signals and responses" [26]. This cascade of biochemical activity and signaling within the cell set off by the ligand binding the receptor is called a signaling pathway.

Deep mutational scanning (DMS) is a type of high-throughput functional assay used to assess the effects of hundreds of possible mutations[1] in parallel. For a given protein of interest (the 'wild-type' protein), the variants of interest (such as every possible amino acid change) are synthesised. These proteins are

[1]typically single-amino-acid mutations, but multiple residue deep mutational scans are also possible, although not as common

barcoded so that activity can be tracked in parallel. The activity of each of these proteins is measured and compared to the activity of the wild-type protein to assess the effects of a single mutation at each position. We will refer to deep mutational scanning later when discussing variant effect prediction models (such as EVE [17], on which this work is based).

2.1.2 Multiple Sequence Alignment (MSA)

Multiple Sequence Alignment is the output of searching a database of amino acid sequences for similarity to a given query sequence (or sequences). The resulting sequences may be functionally, structurally, or evolutionarily related. The sequence search technique models point mutations (single amino acid changes for proteins), as well as insertions or deletion mutations (called *indels*) that occur as a result of evolution. The MSA can therefore be represented as a matrix, where each sequence of amino acid residues is represented as a row of equal length in the matrix, with appropriate substitutions, insertions and deletions to align to the query sequence.

Software tools such as BLAST [27] JackHMMER [28] and HHBlitz [29] may be used to generate these alignments. The problem of aligning multiple sequences has been shown to be NP-complete [30], so many methods rely on heuristics and different objective functions to generate these alignments.

The information from an MSA can also be used for downstream tasks such as phylogenetic reconstruction, inferring evolutionary couplings between amino acid sites [31], and mutation effects.

Multiple Sequence Alignment is one of the most widely used scientific tools, with a 1994 paper describing ClustalW [32] ranked as the 10th most cited scientific paper of all time in 2014 [33], with BLAST [27] not far behind.

2.1.3 G protein-coupled receptors (GPCRs)

We focus on a particular superfamily of proteins, the G protein-coupled receptors (GPCRs), both due to their medical significance as the target of 34 percent of all approved drugs [6] and the abundance of sequence data, experimental labels,

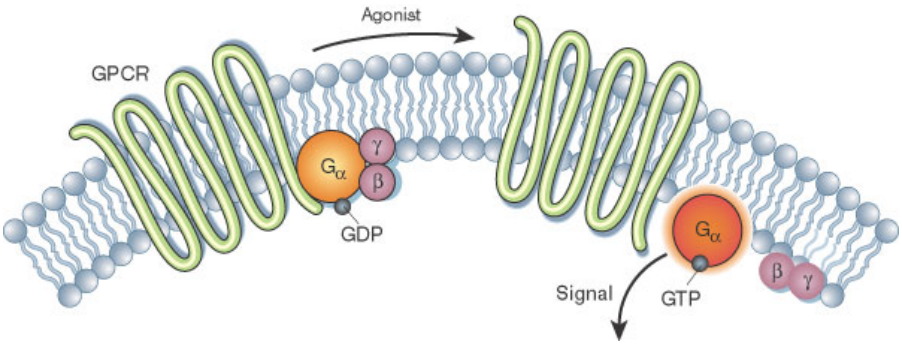


Figure 2.2: In the inactive state, the GPCR is bound to the G protein consisting of the alpha, beta and gamma subunits. Upon stimulation by a ligand (agonist), the GPCR changes shape (conformation) and activates the G-alpha subunit, which then dissociates and activates other molecules in the cell. Figure taken from [34].

and functional diversity. GPCRs are responsible for recognising a wide variety of ligands, from small molecule drugs to entire proteins.

The **G-protein** with which the G protein-coupled receptor interacts, shown in Figure 2.2, consists of three subunit proteins (alpha, beta, gamma) on the inside of the cell membrane. Upon binding a ligand, the GPCR undergoes conformational change and activates the G protein, after which the G-alpha subunit dissociates from both the GPCR and the beta-gamma complex. At this point, both the G-alpha subunit and the beta-gamma complex are free to interact with other molecules in the cell to promote signaling cascades, in turn activating other proteins (usually kinases) and changing gene transcription [34] (see Figure 2.2).

One of our two main themes, Chapter 4 on signaling activation prediction, focuses on predicting this G-alpha protein coupling and downstream activation for different GPCRs.

While many GPCR proteins have been characterised as binding to a known ligand, others remain "orphaned" without any known ligand and/or unknown function [35, 36]. This problem of identifying the ligand class of orphan proteins (or finding out whether they self-activate in the absence of a ligand) is known as 'deorphanisation' and is useful from a medical perspective. Discovering the specific ligand that binds to an orphan GPCR expands our understanding of GPCR

roles in the body (and its role in diseases), and this new drug target may result in novel therapeutic treatments.

2.2 Machine Learning

We will begin by discussing the three important regimes of machine learning (supervised, unsupervised and semi-supervised) used in this work, define some basic terms (classification, regression, clustering) and introduce two relevant algorithms: PCA and k-means.

We then introduce neural networks, a powerful and flexible class of machine learning models, and detail the training algorithms and some relevant technical terms. We end with variational autoencoders (VAEs), a class of latent variable models that use neural networks, and Bayesian VAEs.

2.2.1 Machine learning definitions

Supervised learning refers to machine learning algorithms that use labelled datasets (typically predicting a given output $\boldsymbol{y}$ given an input $\boldsymbol{x}$). These supervised learning models can either be used for **classification** (assigning inputs to one or more categories) or **regression** (predicting a numerical output value for a given input).

Unsupervised learning algorithms are used for datasets where labels are not available. **Clustering** is an example of an unsupervised learning technique in which data points are grouped together according to some measure of similarity. **k-means** is an example of a clustering technique used later in this work: The algorithm aims to find k cluster 'centroids' such that the *within-cluster sum-of-squares* (or *inertia*) of each cluster is minimised. In other words, we want to partition the dataset so that clusters are as small and homogenous as possible. The method shown in Figure 2.3 achieves this by using Lloyd's algorithm, a variant of the Expectation-Maximisation (EM) algorithm. The algorithm 1) assigns each point to its nearest cluster centre, and 2) updates the cluster centres (centroids) to the mean of their assigned observations. Under typical distance measures, the

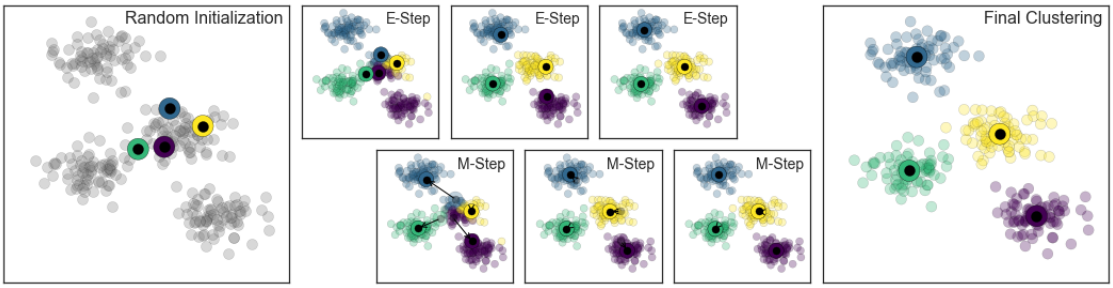


Figure 2.3: The k-means algorithm uses a variant of the Expectation-Maximisation (EM) algorithm to 1) assign each point to its nearest cluster centre, and 2) set the cluster centres to the mean of its assigned observations. From Jake VanderPlas's Python Data Science Handbook [37].

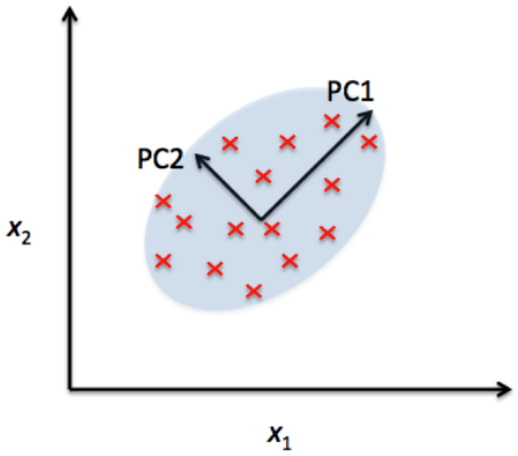


Figure 2.4: A simple representation of PCA: the first principal component preserves most of the variation of the data along its axis. From [40].

algorithm is guaranteed to converge to some solution, although the solution is not guaranteed to be optimal (the random centroids initialisation may effect the results). **Dimensionality reduction** is another form of unsupervised learning which is typically used when the dimension of the input is too large for analysis. The dimension is reduced to some manageable size, while still maintaining relevant properties of the original data, such as information or distance (how these properties are determined is a choice for the practitioner). For example, Principal Component Analysis (PCA) [38] is a technique that seeks to preserve most of the data's original variability (i.e. statistical information) [39], by finding the axes of maximum variation in the original data space (see Figure 2.4).

Finally, with **semi-supervised learning**, only a portion of the examples may be labelled, while the rest of the dataset is unlabelled, so one may have to use a combination of supervised and unsupervised methods. For example, one may want to augment unsupervised clustering methods to incorporate the few available labels to yield a better classifier than either the unsupervised or supervised methods alone could provide. In Chapter 4, we will introduce a semi-supervised model that we use for the task of signaling activation prediction (where only 0.1% of the dataset is labelled).

2.2.2 Neural Networks

Neural networks are a powerful class of machine learning models that have found wide success in the fields of computer vision, natural language processing, reinforcement learning and bioinformatics. Neural networks perform particularly well when applied to large datasets of complex, possibly high-dimensional, data.

In its simplest form, an (Artificial) Neural Network is referred to as a **multilayer perceptron (MLP)**. An MLP is comprised of multiple layers of **neurons** (or **units**) which each take as input the previous layer's outputs. A given hidden unit h_i will compute a weighted sum of the previous layer's outputs x_j (with an optional bias, or intercept term b_i), such that $h_i = \sigma(\sum_j w_{ij}x_j + b_i)$, where w_{ij} is the weight assigned to a particular unit in the previous layer. A non-linear **activation function** $\sigma()$, such as $\text{ReLU}(x) = \max(0, x)$, is applied to this weighted sum[2].

Figure 2.5 shows such an example of an MLP with two hidden layers between an input and output layer.

Loss function and training

Neural networks (and many other machine learning methods) optimise their parameters (weights and biases) to minimise a chosen loss function (or cost/objective function). For example, for a supervised regression task (predicting a continuous

[2]The use of non-linear activation functions applied to every layer is what gives neural networks their expressive power. Multiple layers of weighted sums with no non-linearities would simplify to a single weighted sum of the input (linear projection).

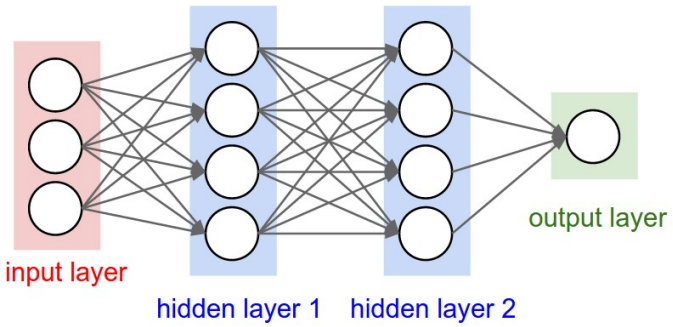


Figure 2.5: A feedforward neural network with two hidden layers of 4 units each and an output layer of 1 unit. From the Stanford CS231n course notes: [41].

output), we could use the squared error (or quadratic) loss function. We could then minimise an empirical estimator of the squared-error loss: the mean-squared error (MSE) between the predictions $\hat{Y}$ and the true targets Y, over N data points:

$$\text{MSE}(\hat{Y}, Y) = \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

Neural networks (and many other machine learning models) are typically trained using an iterative algorithm called **gradient descent**[3]. Given a loss function $\mathcal{L}(\omega)$ for a given configuration of weights ω, we can take small steps with coefficient η (referred to as the **learning rate**) in the negative direction of the gradient of the loss function with respect to the weights $\omega_{new} = \omega_{old} - \eta \nabla_\omega \mathcal{L}(\omega)$ to approach the optimal set of weights that minimises the loss function. The gradient of the loss with respect to the weights can be computed efficiently one layer at a time (starting backwards from the output layer) by application of the chain rule, in a process called backpropagation [42]. In the case of large datasets, it may be computationally expensive to calculate the estimated empirical loss over the entire training set, so we may want to estimate the gradient using only one random example at a time (e.g. using a loss function depending on targets y_i and predictions $\hat{y}_i$ as before): $\omega_{new} = \omega_{old} - \eta \nabla_\omega \ell_\omega(y_i, \hat{y}_i)$, called stochastic gradient descent (SGD). SGD, due to taking steps using noisy estimates of the gradient, is also less vulnerable to saddle points than full (batch) gradient descent. Typical methods will instead compute the

[3]Many alternative methods for training neural networks without backpropagation or gradient descent exist.

gradient across a mini-batch of examples, compromising between a noisy gradient estimate using one example and the expensive full (batch) gradient gradient across the entire dataset. More information on neural network training presented above (as well as omitted details such as regularisation or adaptive optimisation) can be found in Chapters 5.2, 5.3, and 5.5 of Bishop (2006) [43]

Categorical data

When using numerical inputs, the input to the neural network is simply a real number $\boldsymbol{x} \in \mathbb{R}$, or a d-dimensional vector $\mathbb{R}^d$ for multidimensional inputs. However, for categorical inputs (like one of 26 letters from the alphabet, or one of 20 amino acids) where there is no ordering over categories, we can use **one-hot encoding** to represent the input. The process is as follows:

- Assign a unique index from 1 to d to each of the d categories (say, A=1, B=2, C=3). In this example, the letter B will have index 2.
- The one-hot encoded vector is a vector of all 0s, but 1 in the i'th position, where i is the index corresponding to the category of the input. So our one-hot encoding of the letter B will be [0,1,0].

The **softmax** function is used for multiclass classification problems in neural networks (and for multinomial logistic regression), as it gives us a differentiable 'soft' assignment to 1 of many categories. It can also used more generally for problems in which we predict categorical outputs (such as a Variational Autoencoder with categorical inputs and outputs). It turns a vector of K real values into a vector of values between $[0, 1]$ which sum up to 1, resembling a vector of probabilities.[4]

The definition of the softmax function is as follows:

$$\text{softmax}(x_1, \ldots, x_d) = \left(\frac{e^{x_1}}{\sum_{i=1}^{d} e^{x_i}}, \ldots, \frac{e^{x_d}}{\sum_{i=1}^{d} e^{x_i}} \right) \tag{2.1}$$

[4]The softmax function can be seen as a generalisation of the sigmoid function (for binary logistic regression) to the multiclass setting, or a soft approximation to the argmax function.

When we interpret the outputs from the softmax function as categorical probabilities (such as supervised classification problems), we can use the cross-entropy loss function between predictions and true class labels to train a network.

2.2.3 Variational Autoencoders (VAE)

A **generative model** seeks to approximate the true probability distribution of data $p(\boldsymbol{x})$ given N samples $x_1, \ldots, x_N \sim p(\boldsymbol{x})$. We can approximate this distribution using a chosen model $p_\theta(\boldsymbol{x})$ with parameters θ.

The generative model can be trained to maximise the probability of generating real data samples. Assuming the data points were sampled independently, this likelihood factorises as:

$$p_\theta(\mathcal{D}) = \prod_{i=1}^{N} p_\theta(x_i) \tag{2.2}$$

We commonly write the above equation using log-likelihoods, because then we can express the result as a sum of log-likelihoods of data points instead of a product, for numerical stability:

$$\log p_\theta(\mathcal{D}) = \sum_{i=1}^{N} \log p_\theta(x_i) \tag{2.3}$$

Due to the monotonicity of the logarithm function, maximising the log-likelihood corresponds to maximising the likelihood.

The advantage of generative models is that after inference, we can sample new instances x' that appear to come from the data generating distribution $p(\boldsymbol{x})$. **Deep generative models** refer to generative models parametrised by a deep neural network.

We may want to assume some sort of structure to the data generating process. In the case of latent variable models (such as a VAE), we assume that the process involves some unobserved continuous random variable $\boldsymbol{z}$ as follows:

- An unobserved **latent variable** z_i is sampled from a prior distribution[5] $p(\boldsymbol{z})$ over $\boldsymbol{z}$.

[5]So called as it is not conditioned on observed data $\boldsymbol{x}$ [44].

- An observed data point x_i is then sampled from some conditional distribution $p(\boldsymbol{x}|z_i)$ that depends on latent variable z_i.

In the case of digits (as shown in Figure 2.8), we could imagine that latent variables would include the choice of digit and some other high-level unobserved characteristics, according to which a digit is generated. We can then perform inference to estimate the values of these (unobserved) latent variables.

Variational Autoencoders (VAEs) [9, 10] provide a method for "jointly learning latent-variable models and corresponding inference models" parameterised by neural networks with parameters θ and ϕ respectively [44]. This combination of neural networks is then trained end-to-end using mini-batches and stochastic gradient descent, as is standard with neural networks.

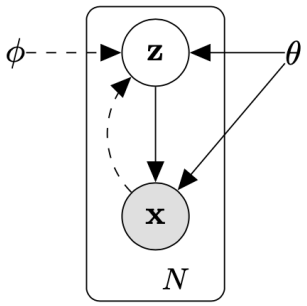


Figure 2.6: We learn a generative model $p_\theta(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})$ (solid lines) and approximate posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ (dashed lines) by optimizing θ and ϕ. From [9].

Figure 2.6 illustrates the VAE as a directed graphical model: The generative model $p_\theta(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})$ described earlier is denoted by the solid lines. The corresponding inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ approximates the true (intractable) posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$, and is denoted by dashed lines. We would like to learn both by optimizing the parameters θ and ϕ. Ideally, we would like to learn a different inference model $q_{\phi_i}(\boldsymbol{z}|x_i)$ to find the optimal variational parameters ϕ_i of the approximate posterior distribution for each observation. However, each new observation would require optimising the inference model to obtain these parameters, and optimising the generative model would require a full pass through the entire dataset. Instead, we use a single set ϕ of global parameters to output latent variables $q_\phi(\boldsymbol{z}|x_i)$ for each data point, *amortising* the cost of training the network across all data points. This *amortised variational inference* allows us to efficiently approximate variational parameters for new, unseen observations (and later allows us to train the inference and generative networks end-to-end).

Optimising our generative model through directly optimising the marginal likelihood of our data $p_\theta(\boldsymbol{x}) = \int p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})d\boldsymbol{z}$ is intractable (as it would involve integrating over all possible values for the latent variable $\boldsymbol{z}$), but through some algebraic manipulation we can find a lower bound on this marginal likelihood (or *evidence*), referred to as the **evidence lower bound** (ELBO) (Also known as the *variational lower bound*). There are varying derivations of this objective, and a comprehensive tutorial for interested readers is provided by Kingma and Welling (2019) [44].

$$\log p_\theta(\boldsymbol{x}) \geq \mathsf{ELBO}(\boldsymbol{x}, \theta, \phi) := \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})) \quad (2.4)$$

Maximising this quantity simultaneously maximises our marginal likelihood $p_\theta(\boldsymbol{x})$ (resulting in a better generative model) while minimising the distance between our approximate posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and the true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ (leading to a better approximation of the posterior) [44].

Practically, the architecture of a variational autoencoder resembles that of a traditional autoencoder [45–47]: An encoder, a lower-dimensional 'bottleneck' representation, and a decoder. The inference/encoder network takes as input a data

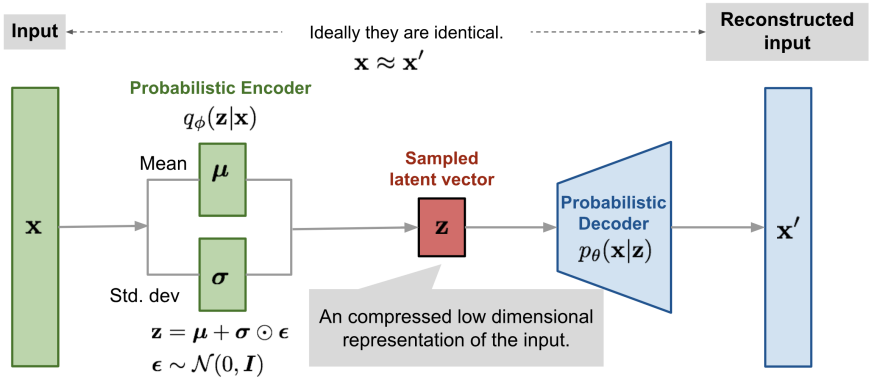


Figure 2.7: The architecture of a VAE with the diagonal multivariate Gaussian assumption, showing similarity to an autoencoder. From the popular Lil'Log blog by Lilian Weng [48].

point x and outputs the mean μ_x and standard deviation σ_x parameters for a diagonal multivariate Gaussian $\mathcal{N}(\boldsymbol{z}; \mu_x, \sigma_x^2\mathbf{I})$, representing the approximate posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$.

After sampling a latent variable from this distribution $z_x \sim \mathcal{N}(\boldsymbol{z}; \mu_x, \sigma_x^2\mathbf{I})$, the generative/decoder network outputs a sample x' from $p_\theta(\boldsymbol{x}|z_x)$.

Training the network would require backpropagating gradients through the sampling operation $z_x \sim \mathcal{N}(\boldsymbol{z}; \mu_x, \sigma_x^2\mathbf{I})$, which is not possible since sampling is a stochastic operation. However, using the *reparameterization trick* [9, 10], we can express our sampling operation as a deterministic, differentiable product $z_x = \mu_x + \sigma_x \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ is regarded as an external source of noise.

VAE loss function and training

Although VAEs serve a different purpose to autoencoders [45–47] and have their roots in variational Bayesian methods, it may be useful to the reader to decompose the terms of the ELBO estimator in terms of an autoencoder's *reconstruction* and *regularisation* errors (See Figure 2.7).

The first term of the ELBO, $\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})]$, resembles the negative *reconstruction error* of an autoencoder. Taking L random samples from the approximate posterior $z_1, \ldots, z_L \sim q_\phi(\boldsymbol{z}|x_i)$ for a given input x_i, we have the following Monte Carlo estimator for the reconstruction error:

$$\mathcal{L}_{rec} = -\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|x_i)}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] \approx -\frac{1}{L}\sum_{j=1}^{L} \log p_\theta(\boldsymbol{x}|z_j) \tag{2.5}$$

In the case where our inputs and outputs to the network are d-dimensional binary data $x_i \in \{0, 1\}^d$, we can use a Bernoulli MLP decoder $f_\theta(\boldsymbol{z})$ that outputs probabilities from a latent variable $\boldsymbol{z}$. This likelihood term $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ has an analytical solution (in the form of the binary cross-entropy between the decoded probability $f_\theta(\boldsymbol{z})$ and original $\boldsymbol{x}$):

$$\log p_\theta(x_i|z_i) = x_i \log(f_\theta(z_i)) + (1 - x_i) \cdot \log(1 - f_\theta(z_i)) \tag{2.6}$$

The second KL divergence term, resembling the *regularisation loss* of an autoencoder, has an analytical solution in the case of a Gaussian prior $\mathcal{N}(\boldsymbol{z}; \mathbf{0}, \mathbf{I})$ and Gaus-

sian posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \mu_x, \sigma_x^2\mathbf{I})$ (derived in Kingma and Welling (2014) [9]):

$$\mathcal{L}_{reg} = D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})) = \frac{1}{2}\sum_{j=1}^{J}(1 + \log((\sigma_x^{(j)})^2) - (\mu_x^{(j)})^2 - (\sigma_x^{(j)})^2) \quad (2.7)$$

where $\sigma_x^{(j)}, \mu_x^{(j)}$ refer to the j'th elements of the parameters for a multivariate Gaussian $\mathcal{N}(\boldsymbol{z}; \mu_x, \sigma_x^2\mathbf{I})$ output by the inference network for a given input x.

Therefore, maximising the ELBO estimator for a typical VAE using a single forward pass ($L = 1$) is equivalent to minimising the following loss function for a given x_i:

$$\begin{aligned}\mathcal{L} &= -\mathsf{ELBO}(x_i, \theta, \phi) \\ &= \mathcal{L}_{rec} + \mathcal{L}_{reg} \\ &\approx [x_i \log(f_\theta(z_l)) + (1 - x_i) \cdot \log(1 - f_\theta(z_l))] \\ &+ \frac{1}{2}\sum_{j=1}^{J}(1 + \log((\sigma_x^{(j)})^2) - (\mu_x^{(j)})^2 - (\sigma_x^{(j)})^2)\end{aligned} \quad (2.8)$$

To generate new data points from $p_\theta(x)$, we can simply sample latent variables from the prior $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \mathbf{0}, \mathbf{I})$ and decode them using the generator network. One can also sample along different points in latent space, to interpolate between latent variables z_1 and z_2. Figure 2.8 from one of the original VAE papers by Kingma and Welling [9] shows such interpolation between a grid of points in the latent space of a VAE trained on MNIST digits.

2.2.4 Bayesian VAE

A Bayesian VAE extends the variational approximation in a VAE to learn distributions over both the latent variables $\boldsymbol{z}$ and the model parameters θ. Instead of learning point estimates θ for each of the parameters in the neural network, we learn a distribution over the parameters $\theta \sim p(\theta)$. A common approximation uses a Gaussian approximate posterior over the weights (similar to the diagonal Gaussian approximate posterior form for the latent space in a VAE). Further details

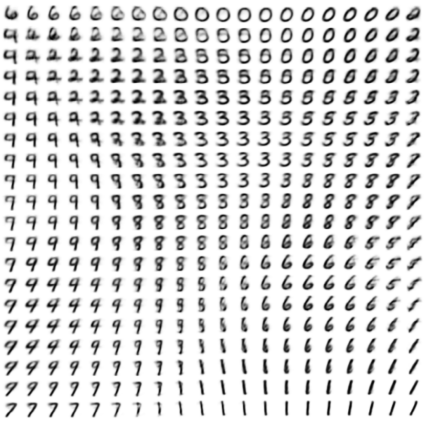


Figure 2.8: Visualisation of the learnt data manifold for a VAE trained on MNIST digits. Generative decodings $p_\theta(\boldsymbol{x}|\boldsymbol{z}_i)$ are shown for a grid of points z_i in the learnt latent space. One can see smooth interpolation between the decodings from different regions in latent space. From [9].

for the interested reader can be found at [49], but below follows a description of its utility for VAEs.

Bayesian Neural Networks (BNNs) allow us to sample different predictions from the network for a single input (by sampling a new set of weights for every prediction). For Bayesian VAEs, we can sample different decodings from a single point in latent space and obtain a measure of the uncertainty of the BNN decoder for that input point.

2.3 Machine Learning for Proteins

2.3.1 Unsupervised protein sequence models

The goal of unsupervised protein sequence models is to capture the natural distribution of amino acid protein sequences observed across organisms. Over millions of years, evolution selects against proteins that decrease the fitness of an organism, and so the resulting natural distribution we see today implicitly captures constraints on the protein sequence that maintain fitness [17]. As mentioned in Section 2.1.2, evolutionary information in the form of covariation of amino acid residues can be used to infer functional or structural properties [50, 51].

Traditional unsupervised approaches would model pairwise covariation between evolutionarily related sequences with a Potts model (an instance of a Markov

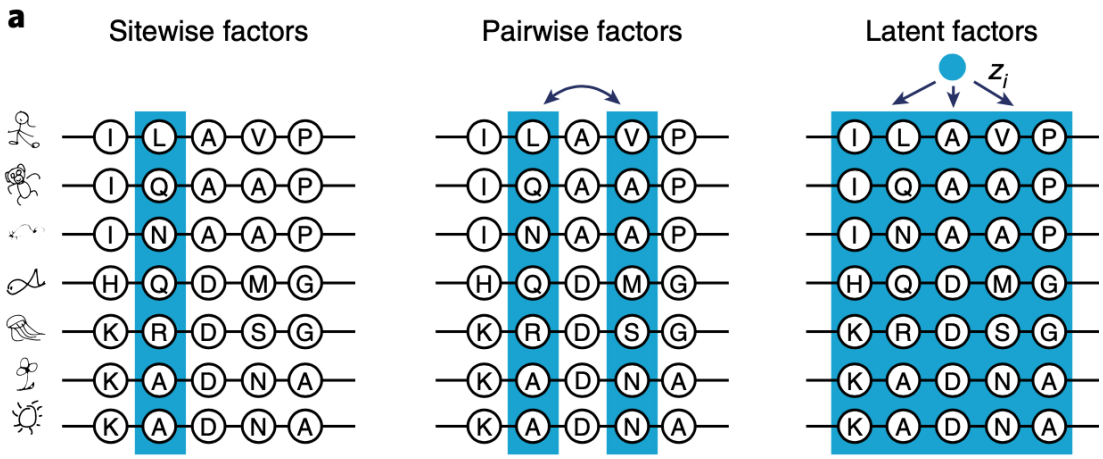


Figure 2.9: From Riesselman et al. [14]: A nonlinear latent-variable model captures higher-order dependencies in proteins and RNAs.

Random Field model) [51], but later deep learning approaches were able to capture higher-order correlations (see Figure 2.9).

DeepSequence [14] trained a VAE for each protein family, using as training data a Multiple Sequence Alignment (MSA) of homologous protein sequences across organisms. DeepSequence successfully predicted the effects of mutations in Deep Mutational Scanning experiments better than existing approaches at the time. DeepSequence is a predecessor of the Evolutionary model of Variant Effects (EVE) [17], the protein VAE we use in this work.

Several unsupervised models have used techniques from natural language processing (NLP) to build so-called 'protein language models'.

Bepler and Berger [52] (and later Heinzinger et al. [53]) trained a LSTM-based language model based on ELMo [54], an LSTM trained with a language modelling objective to predict the next word in a sequence.

Another significant protein language model was UniRep [16], based on the ULMFiT language model, which generated embeddings that encoded 'physicochemistry, organism-level information, structure, evolutionary and functional information'.

Recently, research labs in industry have released state-of-the-art protein language models based on Transformers [13], such as Facebook AI Research's Transformer from the TAPE paper [55], Evolutionary Scale Model (ESM-1b) [56], and MSA

Transformer [19]. Salesforce AI have also designed protein language models such as ProtBERT [57] and ProGen (for protein generation) [58].

Lastly, although not unsupervised, a similar Transformer-based architecture with a masked language modelling loss was most recently used as a component of DeepMind's recent AlphaFold2 model [59, 60] for structure prediction.

For our setting, exploring the latent space of a protein VAE such as DeepSequence/EVE is most appropriate for our goals, since we would like a natural way to go from embeddings to novel generated sequences by decoding points from the latent space. While it is possible to generate novel, fit proteins with protein language models (optionally specifying certain desirable properties), there is no structure enforced in the space (unlike VAEs), so decoding novel proteins from specific points in the embedding space may generate many invalid proteins.

2.3.2 Evolutionary model of Variant Effects (EVE)

We extend the protein VAE model from Frazer et al. [17], so it is appropriate to describe the architecture here in-depth.

The Evolutionary model of Variant Effects (EVE) is a Bayesian VAE with a similar architecture to DeepSequence by Riesselman et al. [14]. This Bayesian VAE is trained on an MSA generated for each protein of interest, and this MSA is constructed by searching the UniRef database [61] of ∼250 million natural protein sequences across hundreds of thousands of organisms.

The objective of EVE (and of DeepSequence on which it is based) is to predict protein fitness in a fully unsupervised manner by generating 'evolutionary indices' for given mutations.

Evolutionary index

Evolutionary indices are simply a measure of how likely a given protein/mutation is under the evolutionary distribution of proteins. By sampling from the approximate

posterior of the VAE, the authors compute the 'evolutionary index' of each single amino acid variant of the reference sequence, which approximates the log-ratio

$$\log \frac{p(\boldsymbol{x}_s \mid \boldsymbol{\theta})}{p(\boldsymbol{x}_{wt} \mid \boldsymbol{\theta})} \tag{2.9}$$

the relative likelihood of a mutated sequence $\boldsymbol{x}_s$ compared with the wild-type (reference) sequence $\boldsymbol{x}_{wt}$. This same log-ratio was used in the DeepSequence paper (and Hopf et al. (2017) [62]) to predict mutation effects, and is closely related to the log-odds conservation scores used in previous mutation effect predictors [63].

Since these exact likelihoods are intractable in practice, the log-likelihoods are approximated using the ELBO (see Equation 2.4 for background). The evolutionary index E_s of a sequence $\boldsymbol{x}_s$ is therefore approximated as follows:

$$E_s \approx \log \frac{p(\boldsymbol{x}_s \mid \boldsymbol{\theta})}{p(\boldsymbol{x}_{wt} \mid \boldsymbol{\theta})} \approx ELBO(\boldsymbol{x}_s) - ELBO(\boldsymbol{x}_{wt}) \tag{2.10}$$

Lastly, these quantities are estimated via generating N Monte-Carlo samples $z_1, \ldots, z_N \sim q_\phi(\boldsymbol{z}|\boldsymbol{x}_s)$ from the approximate posterior as follows (from Equation 2.4):

$$ELBO(\boldsymbol{x}_s) \approx \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(\boldsymbol{x}_s|\boldsymbol{z}_i) + D_{KL}((q_\phi(z_i|\boldsymbol{x}_s)||p_\theta(z_i)) \tag{2.11}$$

Further details on evolutionary indices and their use in the EVE architecture are provided in the Appendix Section A.

VAE architecture and training

A crucial difference between the DeepSequence [14] and EVE architectures and a vanilla VAE described in Section 2.2.3 is that both DeepSequence and EVE use a Bayesian MLP as a decoder[6]. The Bayesian VAE architecture is shown in Figure 2.10.

As input, the VAE takes a one-hot encoded amino acid sequence. A short background to one-hot encodings can be found in the Neural Network background

[6]Background Section 2.2.3 includes more information on Bayesian VAEs.

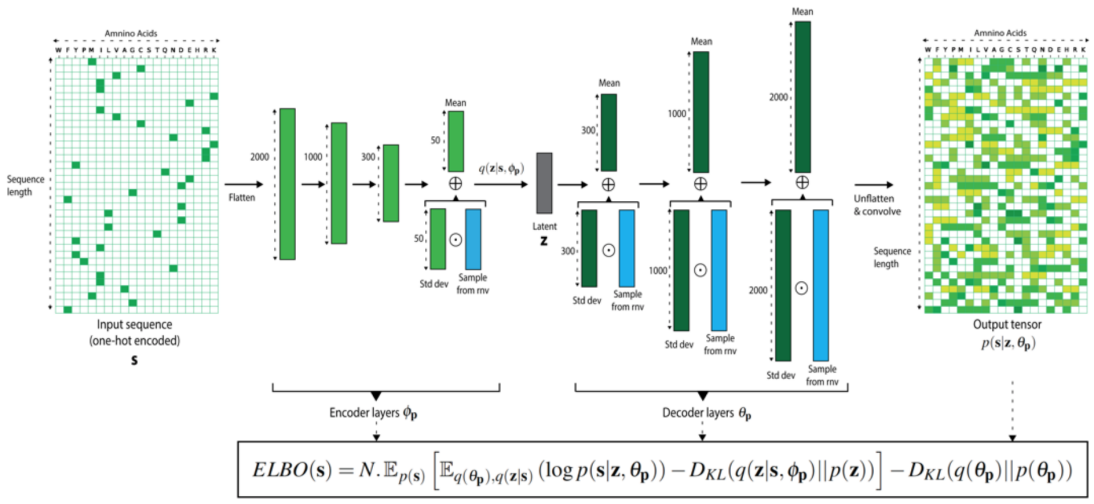


Figure 2.10: The Bayesian VAE architecture consists of symmetric 3-layer encoder and decoder networks (with 2000, 1000, 300-dimensional hidden layers in the encoder and the opposite in the decoder) and a latent dimension of 50. A one-hot encoded sequence is flattened before input to the encoder, and the softmax outputs are again reshaped to yield probabilities across amino acids for each position in the sequence (right: higher probabilities in dark green, lower probabilities in light green). The ELBO is defined in terms of the reconstruction error and KL-divergence of the latent representation from the prior, as well as the KL divergence of the fully-factorized decoder weights distribution $q(\theta_p)$ from a standard multivariate normal prior. The Bayesian VAE is trained by maximising this lower bound, which in turn maximises the intractable log-likelihood of the data $p(\boldsymbol{x})$. Taken from [17].

section, Section 2.2.2. The output of the network contains continuous scalars (logits), one for each choice of amino acid in the sequence. The magnitude of these values denotes the relative probability of each amino acid occurring in the sequence - therefore, to get valid (normalised) probabilities between $[0, 1]$ with $\sum_i p_i = 1$, the softmax function is applied across every amino acid position (see Neural Network Section 2.2.2 for a description of the softmax function).

To translate these output probabilities to a sequence, one can simply sample each of the amino acid positions according to its probability (or take the amino acid with the maximum probability).

The authors use a ReLU activation function and dropout [64] for the decoder and a 1x1 convolution after the final hidden layer. Further information on the EVE model and code can be found in the Appendix Section A or the model

repository https://github.com/OATML/EVE.

Loss function and training

Recall that in the VAE background (Section 2.2.3), the ELBO was defined in terms of reconstruction error and KL divergence of the approximate posterior from the prior (Equation 2.4). In this case, the KL divergence of the decoder weights distribution from its standard multivariate normal prior is also included in the ELBO (see Figure 2.10):

$$\mathsf{ELBO}(\boldsymbol{x}, \theta, \phi) := N \cdot \Big[\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}))\Big] - D_{KL}(q(\theta_p)||p(\theta_p)) \tag{2.12}$$

Using the negative ELBO as the objective function, which we minimize using gradient descent, in turn minimizes the negative marginal log-likelihood $-\log p_\theta(\boldsymbol{x})$.

Similar to DeepSequence, we assume that the conditional distribution of the data $\log p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is a categorical (multinoulli) distribution to model discrete characters, and therefore use categorical cross-entropy between the decoder outputs and data as the reconstruction loss.

Data: MSA and sequence re-weighting

The model was trained to minimise the negative of the ELBO estimator, using mini-batches of sequences from the MSA. The sequences available in genomic databases aren't sequenced uniformly across all living organisms - they are biased by human sampling (where certain species more likely to be studied and sequenced), and by evolutionary sampling (phylogeny). As a result, the MSA training data does not reflect the evolutionary distribution of sequences across organisms. To correct for these biases, and to allow the model to better reflect the evolutionary distribution of sequences, the authors use a re-weighting scheme described in Ekeberg et al.(2013) [65] and Hopf et al. (2017) [62], assigns a weight to each sequence inversely proportional to the number of similar sequences within a Hamming distance threshold T. Similar to DeepSequence [14], the authors set $N = N_{eff}$ in the ELBO

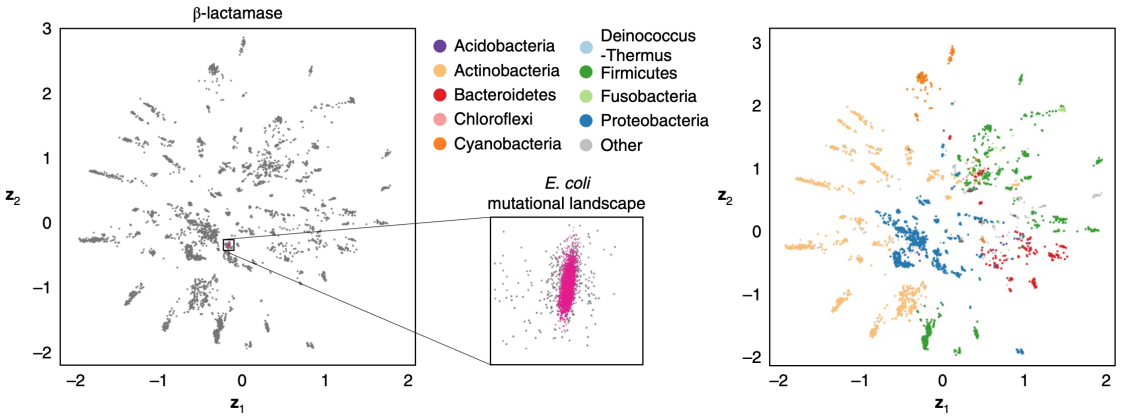


Figure 2.11: A DeepSequence VAE with a 2-dimensional latent space, trained on the β-lactomase family of proteins. A single deep mutational scanning experiment occupies a small region of the latent space (centre, pink). The 2-dimensional representation exhibits a structure that corresponds to phylogenetic groupings. Figure from the original DeepSequence paper [14].

equation above, where N_{eff} is the sum of the sequence weights. During training, mini-batches of sequences were then sampled according to these weights.

Latent space exploration

In the original DeepSequence paper, the authors examined a 2-dimensional (rather than 30-dimensional) latent space version of their model, trained on the β-lactomase family of proteins. They found that the latent space exhibited a structure that corresponded to the phylogenetic grouping of evolutionarily related proteins. This investigation formed the basis of our work, in which we aim to examine the latent space of this model trained on a much bigger and diverse (super)family of proteins, the GPCRs.

2.3.3 Interpreting protein representations

Alley et al. [16] found that the UniRep model's vector space encoded structural, functional, and evolutionary information.

Instead of using latent embeddings as we do with protein VAEs, one can use the learnt attention maps as embeddings. Salesforce AI research team found attention heads that targeted contact points (residues in contact in 3D but far in sequence space), binding sites, and progressively more complex properties with increasing

layer depth. Rives et al. [56] found biochemical properties (negative/positive charge, hydrophobic/aromatic, polar, unique) encoded in the weight matrix of the final embedding layer of their ESM-1b model. Rao et al. [66] subsequently found that using these attention maps from ESM-1b for unsupervised contact/structure prediction outperformed existing Potts models and improved this result in later work by taking the entire MSA as input to the Transformer-based architecture [19].

Hawkins-Hooker et al. [67] used a protein VAE for protein design and also found that the latent space encoded biochemical properties. Their goals are similar to one of our themes (signaling activation prediction, Section 4). However, they use a conditional VAE which learns a different conditional distribution for each property value they are interested in, and this conditioning on properties cannot be done post hoc after a latent space is trained. Furthermore, they assume that a label exists for every protein sequence (by using software that estimates solubility for any protein), so they do not have to deal with sparse labels like we do.

In our experiments, we aim to find such functional or evolutionary information in the latent space of a protein VAE (and with specific applications to GPCRs).

3

Uncovering GPCR functional families

The code and notebooks used in both experiments are attached to this dissertation as a zip file. For the signaling activation prediction experiments, changes were made to the original EVE repo (https://github.com/OATML/EVE [17]), and our forked version is available online (link redacted for submission). Most of the code was written in PyTorch [68] and Scikit-learn [69], two open-source machine learning frameworks.

3.1 Objectives

As mentioned earlier, the DeepSequence paper [14] included a brief exploration of a 2-dimensional latent space version of their model, trained on the β-lactomase family of proteins (Figure 2.9). They found phylogenetic groupings in this space, and this analysis prompted our exploration into the latent space of a model trained on the GPCR superfamily. Other papers have examined the latent space of protein VAEs before, mostly considering phylogenetic groupings in a qualitative manner [67, 70].

Performing a quantitative clustering analysis could validate to what extent our model does, in fact, capture these biological properties (such as phylogeny).

Clusters allow us to uncover new subgroups of proteins (for instance, different subgroups of 'orphaned' proteins without any known ligand). We could examine

how these clusterings of orphans in the latent space compare to experimental data of constitutive activity[1] of orphan class-A GPCRs from Martin et al. [71], in order to better characterise orphan receptors. We hope that this clustering analysis can shed light on GPCR functionality.

3.2 Methods

We propose a clustering analysis on the latent space of the VAE trained on multiple sequence alignment (MSA) around the beta-2 adrenergic receptor (ADRB2) human protein in order to classify different subgroups. The MSA was constructed and provided to us by our collaborators at the Marks Lab (see Acknowledgements). The same MSA dataset is also used for our other main theme of analysis (Chapter 4).

The MSA was in the form of a CSV file with $n = 150535$ different protein sequences, each of length 286. Since sequences are not sampled independently from the evolutionary distribution, a re-weighting scheme is employed (as described in the EVE background, Section 2.3.2). The model was trained according to the training process suggested in the original EVE paper [17], and we used experimental data of two separate deep mutational scans [72, 73] to validate the trained EVE model's performance as a variant effect predictor. The ligand class labels were obtained from the GLASS (**G**PCR-**L**igand **Ass**ociation) database [74]. However, it is notable that only 9% (13 567) of the sequences in our MSA had corresponding ligand class labels available.

We use standard clustering algorithms[2] (either k-means clustering or hdbscan [75]) to analyse different subgroups of GPCR proteins. We use both unsupervised clustering metrics (silhouette coefficient) and supervised clustering metrics (adjusted mutual information, adjusted rand index) to evaluate the clustering performance. Explanations of each of these metrics are outside the scope of this dissertation, but are provided by the specific scikit-learn [69] documentation (from the implementation we used): https://scikit-learn.org/stable/modules/clustering.html#

[1] receptors activating without the presence of a ligand

[2] For more information on clustering, see Machine Learning Background Section 2.2

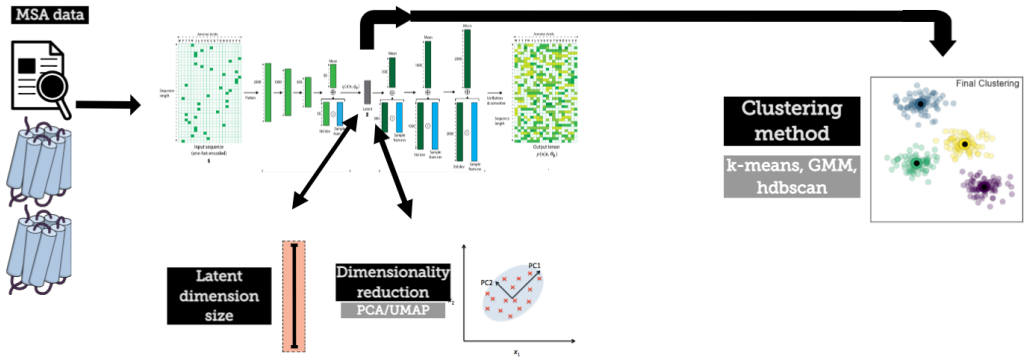


Figure 3.1: An illustration of the experiment process. First, a protein VAE model based on the EVE architecture was trained on an MSA of GPCR sequences. We experiment using different latent dimension sizes, and different dimensionality reduction techniques before clustering.

clustering-performance-evaluation. The supervised metrics are calculated using ligand families or phylogenetic groupings as targets, although they can be calculated using any form of "ground truth" class assignments.

For VAEs trained with relatively high-dimensional latent spaces (e.g., dimensionality greater than 10), we first reduce the dimensionality of the latent representations using PCA [38] before clustering to avoid issues resulting from clustering in high-dimensional spaces [76]. Lower-dimensional representations are also useful for visualising qualitative results. Other clustering methods such as hdbscan [75] and Gaussian Mixture Models yielded poorer performance compared to k-means. We considered using UMAP [77] instead of PCA for dimensionality reduction but did not find improved performance. The time requirements of UMAP were also prohibitive compared to PCA for our prototyping (30 mins compared to 5 mins, for 150k sequences and at most 5720 dimensions for the raw sequences). We also experimented with the k-means number of clusters and found $k = 15$ clusters to yield the best results across experiments.

3.3 Results and discussion

Figure 3.2 shows an example of different subgroups (ligand classes) in the latent space. The plot shows the first two principal components of the 50-dimensional latent space of a VAE trained on the MSA of the beta-2 adrenergic receptor



Figure 3.2: 2D representation (first two principal components) of the 5-dimensional latent space of a VAE trained on the MSA of the beta-2 adrenergic receptor (ADRB2) protein. Different labelled ligand classes are highlighted. Known GPCR families seem to be clustered together in the latent space.

(ADRB2) protein. We can see that the latent space structure roughly corresponds to known ligand class labels.

Embedding dimension	↑ AMI (0-1)	↑ ARI (0-1)	↑ FMI (0-1)	↑ SC (-1,1)
5	**0.5308**	**0.3515**	**0.4575**	0.2565
2-dim PCA from 5-dim latent space	0.4690	0.2781	0.3907	0.3855
2-dim PCA of raw sequences	0.4034	0.1895	0.3113	**0.4087**

Table 3.1: Clustering results for k-means with 15 clusters, compared with ligand class labels (best results highlighted in bold). AMI: Adjusted Mutual Info Score (supervised). ARI: Adjusted Rand Index (supervised). FMI: Fowlkes-Mallows Index (supervised). Silhouette: Silhouette coefficient (unsupervised). ↑: Higher is better. Baseline: 2-dimensional PCA of raw amino-acid sequences.

Table 3.1 shows the results for k-means clustering on a 5-dimensional latent space, relative to ligand class labels from the GLASS (**G**PCR-**L**igand **Ass**ociation)

database [74]. Interestingly, although the 5-dimensional embedding of the latent space produces the best results for the supervised clustering metrics, the baseline 2-dimensional PCA of the raw sequence achieves the best unsupervised score (silhouette coefficient).

The raw sequences have $286 \times 20 = 5720$ dimensions for sequences of length 286 and 20 amino acids per position. The silhouette coefficient is a measure of the cluster cohesion (the intra-cluster distance versus the distance to the nearest cluster of another label). This may mean that the 2-dimensional PCA projection of the raw sequences produces tightly-packed clusters that are separated well by k-means (and therefore produce a high silhouette coefficient), but that these clusters do not correspond to the biological property of interest (ligand class), as measured by the supervised metrics.

We remarked that dimensionality reduction is often needed before clustering to avoid some of the issues resulting from clustering in high-dimensional spaces [76]. However, this dimension reduction results in some difficulties when judging qualitative results. For example, Figure 3.2 shows the ligand class labels for proteins in the 5-dimensional latent space of our protein VAE. It may seem from the figure that some of the classes are overlapping, but these clusters may actually be separable in the 5-dimensional latent space and then overlapping when projected down to two dimensions. The results seem to suggest that the clustering in the original 5-dimensional space does exhibit better separation between classes, and both PCA and UMAP dimensionality reduction techniques, by definition, cannot retain all the information/structure of the original space.

This preliminary analysis presented above provides promising momentum for finding biologically plausible clusters of interest in the latent space.

Further experiments could consider other labels to find if the latent space also encodes other biological categories of interest. Due to time constraints, we were not able to use these clusters to characterise 'orphan' GPCRs to ligand classes, which is elaborated on in our Future Work discussion, Section 5.2.

4
Signaling activation prediction

4.1 Objectives

We would like to use the structure of the learnt sequence embeddings to uncover unknown quantitative biological properties of the corresponding protein sequences. For example, the gain/loss of protein function may be measured quantitatively and may correspond to a certain direction or subspace in the latent space. This is especially helpful for variant effect prediction, where continuous estimation of effects may be more informative than simply binary benign/pathogenic predictions.

As an example application and point of departure, we seek to uncover directions that correlate with downstream signaling activation for 11 distinct Gα proteins [78]. A brief description of protein binding and signaling is provided in Section 2.1.1: 'Background: Proteins'. These latent directions can be found by performing a linear regression of the latent space onto the signaling activation values. We can also jointly train the VAE and linear regression (See Section 5.2) to improve the performance of the linear regression (and therefore, the alignment between latent directions and true activation values).

As an example, projecting the latent space along two of these latent directions may give us a further signal to analyse different subgroups of GPCR proteins (Fig. 4.3). We could use this projected structure of the latent space to tease

apart different clusters of proteins, or to design proteins that maximize a particular property of interest. Therefore, we can conserve the global structure of the latent space, and the utility of the model to capture the evolutionary distribution of sequences (enabling e.g. protein function prediction) while transforming it into another representation according to features of interest.

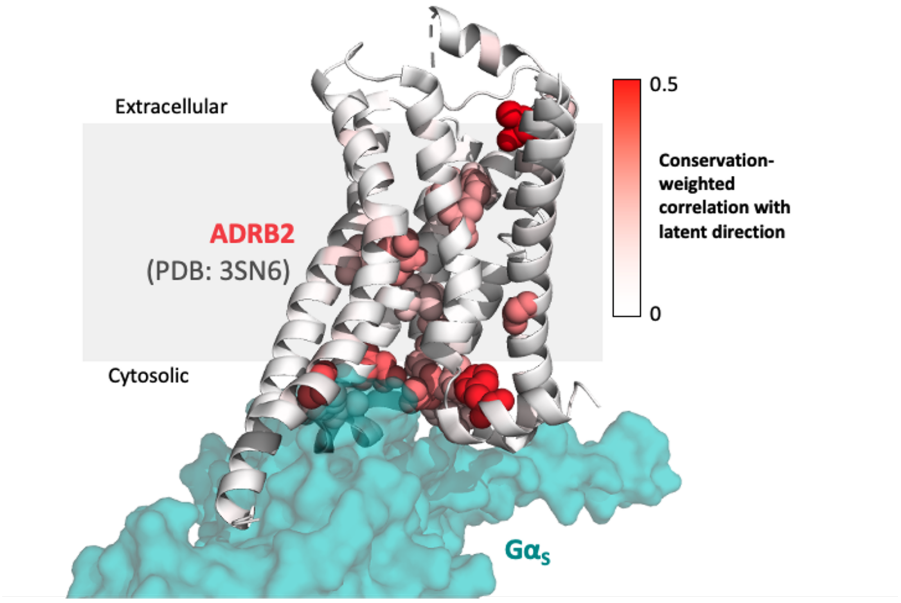


Figure 4.1: Mapping latent direction onto the structure of the beta-2 adrenergic receptor: Residues that vary along this dimension include some that interact directly with the G-protein, but also others throughout the protein. Figure from an analysis done by Sam Berry at Marks Lab (see Acknowledgements Section).

4.2 Methods

For prediction, we used a simple linear regression from the 50-dimensional latent space of the EVE model onto the signaling activity of up to 11 distinct Gα proteins from an experimental dataset collected by Inoue et al [78]. In the results below, we predict signaling activity for one specific Gα protein, the $G\alpha_s$ subunit. Using a linear model on top of the latent space follows a similar precedent set by large pretrained models (such as BERT [12] for text and ESM [56] for proteins). These

works posit that if a representation encodes relevant properties of interest, a single output layer should be sufficient for prediction or fine tuning on downstream tasks.

We implemented the linear regression using a simple linear layer (with a weight and bias) with no activation function from the PyTorch library [68]. There are 128 labelled sequences in this dataset, and the corresponding MSA of the beta-2 adrenergic receptor (ADRB2) protein contains around 150k sequences. This means that only $128/150000 \approx 0.09\%$ of the dataset is labelled. The same MSA was used as in the clustering analysis presented earlier, as well as the same sequence re-weighting scheme when sampling mini-batches for training.

We tried two different forms of training to account for this relatively small portion of labelled data:

In the **frozen** training setting, the VAE and property prediction components are trained separately. The Bayesian VAE is trained as before on unlabelled data according to the process in the EVE paper [17]. Thereafter the VAE encoder/decoder parameters are frozen and the linear regression is trained on the sampled embeddings in the latent space (using only the labelled data). This resembles a fine-tuning setup where a task is trained on a large dataset, and then only a subset of parameters are fine-tuned on a smaller dataset.

In the **joint training** setting, we trained both the VAE parameters and the linear regression property prediction jointly. We looked at either joint training from a VAE checkpoint (a VAE previously trained for 400 000 steps, as usual, using only the unsupervised loss [17]) or from a randomly initialised VAE. Each batch contains a proportion of sampled labelled sequences and a proportion of unlabelled sequences sampled from the MSA (these sample weights are generated during the EVE training process [17]).

We optimise over all the parameters by minimising a loss function which is simply the sum of the unsupervised VAE objective (the negative of the ELBO defined in Equation 2.12) and an auxiliary property prediction loss (MSE). This resembles the training setup used by Gomez-Bombarelli et al. [79] for property prediction from chemical SMILES sequences. Similar joint training has been used

Mode	CV MSE ± s.d. ↓
Baseline (mean)	0.57 ± 0.1
Frozen	0.55 ± 0.07
Baseline (supervised MLP)	$\mathbf{0.43 \pm 0.06}$
Joint	$0.46 \pm 0.21^{*}$

Table 4.1: The cross-validation mean squared error (MSE) of each model. ↓: Lower is better.

in the context of protein design using conditional VAEs [67, 80], although each of these assumes access to an 'oracle' predictive function for the property of interest and therefore they do not have to deal with few labels like we do.

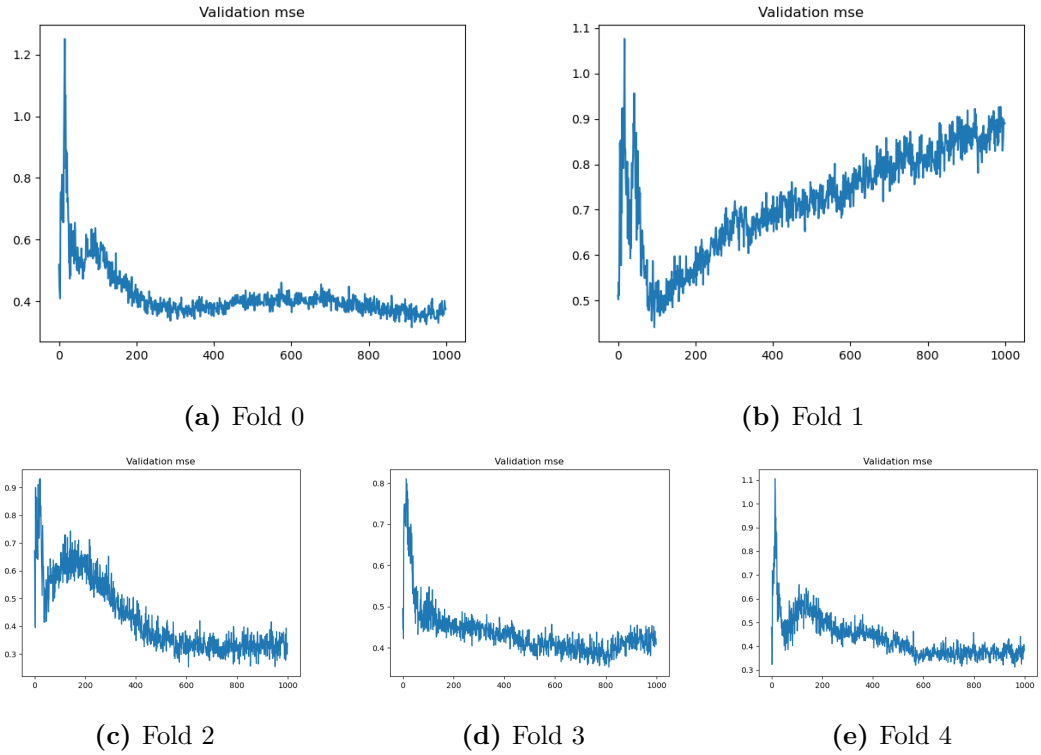
We used a dummy model simply predicting the mean value of the labels as our simple baseline. This would effectively be the same as training a linear regression on random noise as input. As our second baseline (comparing to the joint training) we optimise the parameters of the encoder MLP and property prediction network only on the supervised objective. This would be effectively the same as training the whole VAE and property prediction using only the supervised objective (since the decoder is irrelevant for the task of property prediction).

4.3 Results and discussion

We used one round of 5-fold cross-validation (with a reproducible seed/split across experiments) for our results below.

The frozen linear regression results did not significantly outperform the simple baseline of predicting the mean value of the labels. We interpret this to mean that the latent space after unsupervised training did not contain meaningful information about Gα signaling activation, at least not using linear relationships. The joint training and encoder-only baseline did perform better than the frozen and simple mean baseline, which we interpret as these models extracting valuable information from the sequence.

The joint model did not outperform the encoder-only baseline, but we believe this might be due to methodological issues. For example, the joint model had



(a) Fold 0 **(b)** Fold 1

(c) Fold 2 **(d)** Fold 3 **(e)** Fold 4

Figure 4.2: Validation MSE across cross-validation folds for the joint training experiment. As a qualitative comparison, we can see that Fold 1 did not converge, skewing the results, unlike the other folds.

the following MSE metrics for the five folds: $[0.36, 0.9^*, 0.33, 0.43, 0.37]$ to yield a mean of 0.46 and a standard deviation of 0.21.

*Looking closer at the validation MSE for fold 1 (final MSE = 0.9; hypothesized to be an outlier) compared to the other validation mse curves in Figure 4.2, we can see that Fold 1 in fact did diverge, unlike the other curves. When assessing the training loss curves of this experiment in Figure 4.2, we found that all training losses converged, and conclude that Fold 1 may have started overfitting at some point (at around step 100 on the figure). The encoder-only baseline did not seem to exhibit this overfitting behaviour, and all folds had similar MSE (as shown by the sample standard deviation of 0.06). In subsequent experiments, we were unable to track down the source of this overfitting, and further work is needed to see if significantly better results can be achieved using a joint semi-supervised training scheme.

We compared joint training starting from a VAE checkpoint vs. joint training starting from a randomly initialised VAE. We found joint training to perform

better when starting from scratch (i.e. training using the unsupervised + auxiliary loss). Our interpretation of this performance is as follows: When training from a checkpoint, similar to the 'frozen' training setting, the latent space includes only the information useful for reconstruction/generation. Recall that if this information in the learnt latent space does not include the property of interest (Gα signaling activity) encoded using a linear relationship, the frozen training will perform poorly. Similarly, the joint training setting may also fail to adjust the latent representation (via the encoder) to encode this property when trained from a checkpoint if the necessary information is not present in the learnt latent space. Training from scratch should alleviate these issues, since the auxiliary loss forces the latent space to encode the relevant information from the start. Conversely, the auxiliary property prediction loss may also hamper the training of the VAE by constraining the structure of the latent space representation. Further analysis is therefore needed to evaluate which approach works better for our purposes.

We further discuss these results and alternative approaches in Section 5. We propose methodological improvements in Future Work (Section 5.2) which we hope can improve performance significantly better than baseline.

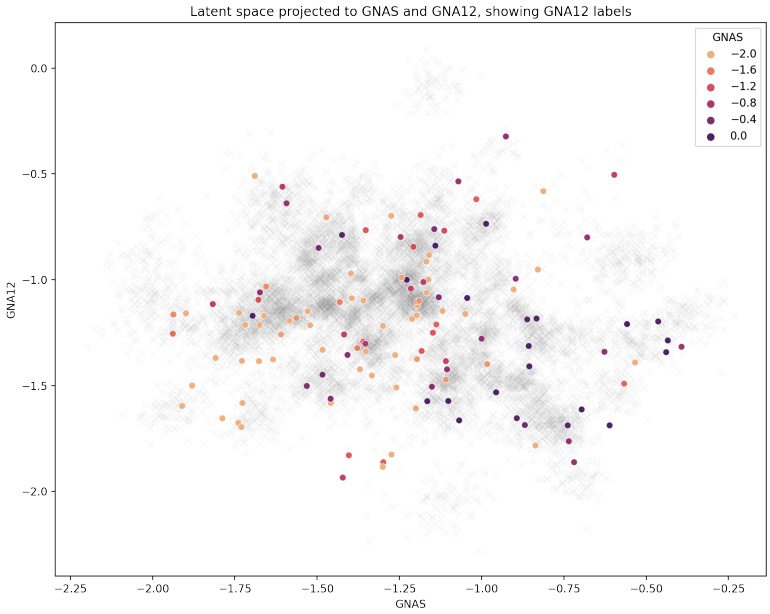


Figure 4.3: VAE latent space trained on GPCR proteins, projected onto vectors corresponding to levels of signaling activation of two different G-proteins by these GPCRs (found by linear regression on 'frozen' latent space). Points are coloured according to their $G\alpha_s$ subunit activations, which should correlate with the estimated $G\alpha_s$ subunit activation vector (x-axis = left-right direction). For example, for orphan proteins, we could keep the learnt structure of the latent space but use these directions to tease apart different clusters.

5
Reflections and future directions

The results and discussion presented in the clustering section provide promising insight that the latent space of a VAE trained on protein data does encode higher-level biological properties (such as the types of ligands to which it binds). The extent to which this would be helpful for orphan characterisation is unknown - it would be useful to look closer into which proteins belonged to 'pure' homogenous clusters, and which clusters contained a variety of different ligand classes. A difficulty with this analysis is that high-level quantitative (or even qualitative) results do not give us adequate insight, and further effort by a domain expert is needed to examine edge cases (or interesting query proteins) on an individual basis.

For the semi-supervised signaling activation prediction experiment, the results seemed to suggest that, unlike the ligand classes, the latent space did not encode the signaling activity without supervision (at least not with linear relationships). However, when trained with labels the encoder was able to extract this information from the sequence. It was hypothesised in the experiment's discussion that the semi-supervised joint training results suffered from implementation issues such as overfitting. We are confident that once these issues are overcome, the semi-supervised joint training results will surpass the supervised training results, even though there are more constraints on the structure of the latent space in this form.

Even with moderate performance, an accurate signaling activity predictor coupled with a learnt latent representation could be used for protein design. Since the EVE VAE model that we use can give us fitness scores for each protein, we can constrain the search to generate functional proteins with desired properties. As discussed in Chapter 4, this sort of semi-supervised process has been used successfully for chemical design [79] and a conditional VAE has been used for functional protein design [67], although neither this low-data regime nor this application (signaling activation prediction) has been considered in these works.

For both themes, it would be interesting to go further with this preliminary analysis. Probing the clustering to examine specific edge cases, or designing real-world proteins with desired signaling activation for different $G\alpha$ subunits and testing performance in the lab could yield more insight.

5.1 Technical discussion

Aside from our main results, there were also several approaches that we tried that did not work as expected and we had to adjust our methods as a result. There were also some practical lessons that I learnt throughout this work.

For the activation prediction theme, we considered using a β-VAE to enforce independence between dimensions in the latent space. There has been prior work on including a small percentage of labels [81, 82], but we abandoned this idea after a brief experiment and no improved performance. We also discussed this notion of 'disentanglement', and a central assumption of the algorithm discussed above is that the ground truth factors of variation of the data are known and independent, whereas we posit that biological factors of variation may well be correlated, with reason.

A limitation of our experimental setup is that I underestimated the time required for a round of analysis and the record-keeping that was required. Often, we would try a new idea (clustering new data, or trying a different architecture) 'temporarily', and then would need to eventually go back and reproduce this analysis later. Even with decent logging and experiment tracking on our experiments on the university cluster, on the order of $\sim$100 experiments this quickly became difficult to track and

manage. In the future, we may need to be more disciplined when implementing 'quick prototypes', and an online, queryable experiment tracker may help manage the many iterations of such a research project.

On the computational biology side, this was new territory and there was a lot of domain knowledge necessary to understand the problem area. I appreciated more that the most critical/useful projects may well be different from using the conventionally 'cutting-edge'/fancy methods, and with good reason. Sometimes finding an important application or critical insight into a specific biological subject is much more impactful than the 'next big model' that does not address questions of interest. In sum, I was reminded of the importance of domain knowledge, and getting a sense for which questions to answer - and this took time to ramp up speed.

5.2 Future work

5.2.1 Methodological improvements

Switching to an alignment-free encoder/decoder (as in [15, 83]) could allow the model to be trained on a wider range of proteins of varying lengths. Currently, all sequences have to be aligned to a query sequence, and the model is only trained on fixed-length protein alignments.

Regressing on both the amino-acid sequence as well as the latent space as recommended by Hsu et al. for a similar property prediction problem could also improve performance [84].

5.2.2 Deorphanisation

As mentioned in Chapter 3, further analysis is needed for the task of characterising 'orphaned' GPCR proteins for which there is no known ligand. We could either look at the latent space clusters to assign orphans to the most probable ligand class (in-depth validation would be needed to do this with confidence). For GPCRs which we suspect might 'self-activate' (activating in the absence of any ligand), we could compare the clusterings of orphans in the latent space to experimental data of constitutive activity of orphan class-A GPCRs from Martin et al. [71].

5.2.3 Going further with semi-supervision

Beyond protein engineering/design, it may be interesting to use semi-supervision in training for clustering/classification. For example, instead of training a property prediction regression model on the latent space, one could train a classifier / clustering model on top of the latent space. This model could be trained jointly with the standard unsupervised VAE as before to create a latent space structure with maximum separation between classes/clusters.

5.2.4 Global vs local properties of latent space

We have so far focused our explorations on "global" properties of the latent space but an open question of great importance is what tradeoffs are being made between global and more fine-grained local properties, such as predicting distinct phenotypes amongst all possible single amino acid substitutions (a form of pleiotropy). Another example is variant effect prediction, which is a local property of only single-amino-acid changes around a protein of interest: it would be interesting if this variant effect prediction could be transferred directly to other proteins in the latent space. The optimal model for global and local latent representations may well not be the same.

5.3 Personal development

The dissertation project was a valuable experience personally. My supervisor encouraged me to take initiatives on how to tackle several research questions (as I would have done when doing a PhD), but the team occasionally had to help me out to keep the momentum and focus of the project.

I did get despondent at times when I had certain expectations of positive results, and big expectations for experiments that I hoped we could realistically finish in time (for example, an in-depth protein engineering application, or some interesting differentiable clustering). We had a few pivots and many new ideas during the project, so it was hard to pin down a few concrete applications/experiments, but this taught me to focus more.

I was also lucky enough to present our work as a poster at a conference workshop, which was both stressful and a lot of fun.

Overall, it was an enriching experience, and my collaborators and supervisors played a big role in making it worthwhile.

6 Conclusion

6.1 Summary

We examined a popular class of models, deep generative models trained on proteins, to address questions about what these models learn from evolutionary protein sequence data.

We apply one such model to a widely studied and clinically relevant family of proteins, the G protein-coupled receptors (GPCRs). The GPCR protein (super)family presents unique challenges: it is functionally diverse (recognising a wide range of ligands) and therefore complex to capture its distribution in a model. It also presents unique opportunities: it is well studied and there is an abundance of sequence and experimental data.

We presented a clustering analysis in the latent space and included a quantitative analysis of its performance in classifying ligand classes. We also encountered a few difficulties with this approach, discussing those along with the promising potential to characterise 'orphaned' proteins.

We applied a semi-supervised approach to predicting properties of interest from the latent space, building on prior work in this area. We encountered some difficulties when training our with such a small proportion of labels (0.1% of

the dataset had labels), and experimented with different training strategies to mitigate these difficulties.

Lastly, we included an extensive discussion based on our experience and results, highlighting our improved understanding of the problem area and opportunities. Notable suggested avenues include a proposed method for 'deorphanisation' using clusters in latent space and/or self-activation prediction, and further exploration into global vs. local structure in the latent space.

We are excited about developments in this area of machine learning on proteins (and particularly applications to GPCRs), and we personally plan to use this momentum to continue pursuing these topics over the coming year.

6.2 Acknowledgements

Redacted for submission.

References

[1] Yu Li et al. "HMD-ARG: hierarchical multi-task deep learning for annotating antibiotic resistance genes". In: *Microbiome* 9.1 (2021), pp. 1–12.

[2] UniProt Consortium. "UniProt: A worldwide hub of protein knowledge". In: *Nucleic Acids Research* 47.D1 (Jan. 2019), pp. D506–D515. URL: https://www.ebi.ac.uk/proteins/api/doc/.

[3] Alex Bateman et al. "The Pfam protein families database". In: *Nucleic Acids Research* 32.DATABASE ISS. (Jan. 2004), pp. D138–D141. URL: http://www.sanger.ac.uk/Software/Pfam/%20http://pfam.wustl.edu/%20http://pfam.jouy.inra.fr/%20http://pfam.cgb.ki.se/.

[4] Mohammed AlQuraishi. "Blog: The Future of Protein Science will not be Supervised". In: *Blog* April 1 (2019), pp. 2–5. URL: https://moalquraishi.wordpress.com/2019/04/01/the-future-of-protein-science-will-not-be-supervised/.

[5] Ali Madani et al. *ProGen: Language Modeling for Protein Generation*. Tech. rep. arXiv: 2004.03497v1.

[6] Alexander S. Hauser et al. "Pharmacogenomics of GPCR Drug Targets". In: *Cell* 172.1-2 (Jan. 2018), 41–54.e19. URL: https://pubmed.ncbi.nlm.nih.gov/29249361/.

[7] Simon D Harding et al. "The IUPHAR/BPS Guide to PHARMACOLOGY in 2018: updates and expansion to encompass the new guide to IMMUNOPHARMACOLOGY". In: *Nucleic Acids Research* 46.D1 (Jan. 2018), pp. D1091–D1106. URL: https://academic.oup.com/nar/article/46/D1/D1091/4628131.

[8] Sebastian Raschka. "Automated discovery of GPCR bioactive ligands". In: *Current Opinion in Structural Biology* 55 (Apr. 2019), pp. 17–24.

[9] Diederik P. Kingma and Max Welling. "Auto-encoding variational bayes". In: *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, Dec. 2014. arXiv: 1312.6114. URL: https://arxiv.org/abs/1312.6114v10.

[10] Danilo J Rezende, Shakir Mohamed, and Daan Wierstra. *Stochastic Backpropagation and Approximate Inference in Deep Generative Models*. Tech. rep. 2014. arXiv: 1401.4082v3.

[11] Ian J Goodfellow et al. *Generative Adversarial Nets*. Tech. rep. arXiv: 1406.2661v1. URL: http://www.github.com/goodfeli/adversarial.

[12] Jacob Devlin et al. "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* 1.Mlm (2019), pp. 4171–4186. arXiv: 1810.04805.

[13] Ashish Vaswani et al. *Attention Is All You Need*. Tech. rep. 2017.

[14] 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. URL: http://dx.doi.org/10.1038/s41592-018-0138-4.

[15] Adam Riesselman et al. "Accelerating protein design using autoregressive generative models". In: *bioRxiv* (2019).

[16] Ethan C Alley et al. *Unified rational protein engineering with sequence-only deep representation learning*. 2019.

[17] J. Frazer et al. "Large-scale clinical interpretation of genetic variants using evolutionary data and deep learning". In: *bioRxiv* (2020).

[18] Alexander Rives 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 (Apr. 2021). URL: https://www.pnas.org/content/118/15/e2016239118.

[19] Roshan Rao et al. "MSA Transformer". In: *bioRxiv* (2021), p. 2021.02.12.430858. URL: https://doi.org/10.1101/2021.02.12.430858.

[20] Kathryn Tunyasuvunakool et al. "Highly accurate protein structure prediction for the human proteome". In: *Nature 2021* (July 2021), pp. 1–7. URL: https://www.nature.com/articles/s41586-021-03828-1.

[21] OpenStax. *3.4 Proteins - Biology 2e | OpenStax*. URL: https://openstax.org/books/biology-2e/pages/3-4-proteins (visited on 08/10/2021).

[22] Creative Commons. *Creative Commons — Attribution 3.0 Unported — CC BY 3.0*. 2018. URL: https://creativecommons.org/licenses/by/3.0/ (visited on 08/10/2021).

[23] Ulrike Göbel et al. "Correlated mutations and residue contacts in proteins". In: *Proteins: Structure, Function, and Bioinformatics* 18.4 (1994), pp. 309–317. URL: https://pubmed.ncbi.nlm.nih.gov/8208723/.

[24] European Bioinformatics Institute. *What are protein families? | Protein classification*. 2020. URL: https://www.ebi.ac.uk/training/online/courses/protein-classification-intro-ebi-resources/protein-classification/what-are-protein-families/.

[25] Ralph A. Bradshaw and Edward A. Dennis. *Handbook of cell signaling, 2/e*. Vol. 1-3. Elsevier Inc., 2010.

[26] P. J. Thomas. "Membrane Receptors". In: *Characterization of Proteins*. Totowa, NJ: Humana Press, 1988, pp. 217–232. URL: https://ezproxy-prd.bodleian.ox.ac.uk:2120/protocol/10.1007/978-1-59259-437-5%7B%5C_%7D8%20http://link.springer.com/10.1007/978-1-59259-437-5%7B%5C_%7D8.

[27] Stephen F. Altschul et al. "Basic local alignment search tool". In: *Journal of Molecular Biology* 215.3 (Oct. 1990), pp. 403–410.

[28] L. Steven Johnson, Sean R. Eddy, and Elon Portugaly. "Hidden Markov model speed heuristic and iterative HMM search procedure". In: *BMC Bioinformatics* 11.1 (Aug. 2010), p. 431. URL: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-11-431.

[29] Michael Remmert et al. "HHblits: Lightning-fast iterative protein sequence searching by HMM-HMM alignment". In: *Nature Methods* 9.2 (Feb. 2012), pp. 173–175. URL: https://www.nature.com/articles/nmeth.1818.

[30] Lusheng Wang and Tao Jiang. "On the Complexity of Multiple Sequence Alignment". In: *Journal of Computational Biology* 1.4 (June 1994), pp. 337–348. URL: https://www.liebertpub.com/doi/abs/10.1089/cmb.1994.1.337.

[31] Thomas A. Hopf et al. "The EVcouplings Python framework for coevolutionary sequence analysis". In: *Bioinformatics* 35.9 (2019), pp. 1582–1584.

[32] 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 (Nov. 1994), pp. 4673–4680.

[33] Richard Van Noorden, Brendan Maher, and Regina Nuzzo. "The top 100 papers". In: *Nature* 514.7524 (Oct. 2014), pp. 550–553.

[34] Nature Education. "GPCR | Learn Science at Scitable". In: *Nature Education*. Figure 1. 2014, pp. 1–5. URL: https://www.nature.com/scitable/topicpage/gpcr-14047471/%20https://www.nature.com/scitable/definition/mapping-282/%7B%5C%%7D0Ahttps://www.nature.com/scitable/topicpage/gpcr-14047471.

[35] S. Chung, T. Funakoshi, and O. Civelli. "Orphan GPCR research". In: *British Journal of Pharmacology*. Vol. 153. SUPPL. 1. Wiley-Blackwell, Mar. 2008, S339. URL: /pmc/articles/PMC2268047/%20/pmc/articles/PMC2268047/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2268047/.

[36] Simon R. Foster et al. "Discovery of Human Signaling Systems: Pairing Peptides to G Protein-Coupled Receptors". In: *Cell* 179.4 (Oct. 2019), 895–908.e21. URL: https://doi.org/10.1016/j.cell.2019.10.010.

[37] Jake VanderPlas. *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media, 2016, p. 541. URL: http://shop.oreilly.com/product/0636920034919.do%7B%5C%%7D0Ahttps://jakevdp.github.io/PythonDataScienceHandbook/05.01-what-is-machine-learning.html.

[38] Karl Pearson. "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572.

[39] Ian T. Jolliffe and Jorge Cadima. "Principal component analysis: a review and recent developments". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (Apr. 2016). URL: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2015.0202.

[40] Sebastian Raschka. *What is the difference between LDA and PCA for dimensionality reduction?* 2013. URL: https://sebastianraschka.com/faq/docs/lda-vs-pca.html (visited on 09/01/2021).

[41] Fei Fei Li, Justin Johnson, and Andrej Karpathy. "CS231n: Convolutional Neural Networks for Visual Recognition". In: *Stanford University* (2020), pp. 1–9. URL: https://cs231n.github.io/neural-networks-1/%20https://cs231n.github.io/convolutional-networks/%20http://cs231n.github.io/neural-networks-3/.

[42] Seppo Linnainmaa. "Taylor expansion of the accumulated rounding error". In: *BIT Numerical Mathematics 1976 16:2* 16.2 (1976), pp. 146–160. URL: https://link.springer.com/article/10.1007/BF01931367.

[43] Christopher M Bishop. *Pattern Recognition*. 2006.

[44] Diederik P. Kingma and Max Welling. "An Introduction to Variational Autoencoders". In: *Foundations and Trends in Machine Learning* 12.4 (June 2019), pp. 307–392. arXiv: 1906.02691. URL: https://arxiv.org/abs/1906.02691v3.

[45] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. "A learning algorithm for boltzmann machines". In: *Cognitive Science* 9.1 (Jan. 1985), pp. 147–169.

[46] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[47] G. E. Hinton and R. R. Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *Science* 313.5786 (July 2006), pp. 504–507. URL: http://arxiv.org/abs/physics/0601055www.sciencemag.org/cgi/content/full/313/5786/502/DC1.

[48] Lilian Weng. *From Autoencoder to Beta-VAE*. 2018. URL: https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html%7B%5C#%7Dvae-variational-autoencoder%20https://lilianweng.github.io/2018/08/12/from-autoencoder-to-beta-vae.html.

[49] Laurent Valentin et al. "Hands-on Bayesian Neural Networks-a Tutorial for Deep Learning Users". In: (2020). arXiv: 2007.06823v1. URL: https://doi.org/10.1145/nnnnnnn.nnnnnnn.

[50] Debora S. Marks et al. "Protein 3D Structure Computed from Evolutionary Sequence Variation". In: *PLOS ONE* 6.12 (Dec. 2011), e28766. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0028766.

[51] Thomas A. Hopf et al. "Three-dimensional structures of membrane proteins from genomic sequencing". In: *Cell* 149.7 (June 2012), pp. 1607–1621. URL: http://www.cell.com/article/S0092867412005090/fulltext%20http://www.cell.com/article/S0092867412005090/abstract%20https://www.cell.com/cell/abstract/S0092-8674(12)00509-0.

[52] Tristan Bepler and Bonnie Berger. "Learning protein sequence embeddings using information from structure". In: *arXiv* (Feb. 2019). arXiv: 1902.08661. URL: http://arxiv.org/abs/1902.08661.

[53] Michael Heinzinger et al. "Modeling the language of life - Deep learning protein sequences". In: *bioRxiv* (2019).

[54] Matthew E Peters et al. "Deep contextualized word representations". In: *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*. Vol. 1. 2018, pp. 2227–2237. arXiv: 1802.05365. URL: http://allennlp.org/elmo.

[55] Roshan Rao et al. "Evaluating protein transfer learning with TAPE". In: *bioRxiv* (2019), pp. 1–20. arXiv: 1906.08230.

[56] Alexander Rives et al. "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences". In: *bioRxiv* 29 (2019), pp. 27–47.

[57] Jesse Vig et al. "BERTology meets biology: Interpreting attention in protein language models". In: *bioRxiv* (2020). arXiv: 2006.15222.

[58] Ali Madani. *ProGen: Using AI to Generate Proteins*. 2020. URL: https://blog.einstein.ai/progen/ (visited on 04/30/2021).

[59] John Jumper et al. "High Accuracy Protein Structure Prediction Using Deep Learning". In: *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book)* (2020), pp. 22–24.

[60] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* (July 2021), pp. 1–11. URL: https://www.nature.com/articles/s41586-021-03819-2.

[61] Baris E. Suzek et al. "UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches". In: *Bioinformatics* 31.6 (Mar. 2015), pp. 926–932. URL: https://academic.oup.com/bioinformatics/article/31/6/926/214968.

[62] Thomas A. Hopf et al. "Mutation effects predicted from sequence co-variation". In: *Nature Biotechnology* 35.2 (2017), pp. 128–135.

[63] Boris Reva, Yevgeniy Antipin, and Chris Sander. "Determinants of protein function revealed by combinatorial entropy optimization". In: *Genome Biology* 8.11 (Nov. 2007), R232. URL: /pmc/articles/PMC2258190/%20/pmc/articles/PMC2258190/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2258190/.

[64] Geoffrey E. Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: (July 2012). arXiv: 1207.0580. URL: https://arxiv.org/abs/1207.0580v1%20http://arxiv.org/abs/1207.0580.

[65] Magnus Ekeberg et al. "Improved contact prediction in proteins: Using pseudolikelihoods to infer Potts models". In: *Physical Review E* 87.1 (Jan. 2013), p. 012707. URL: https://journals.aps.org/pre/abstract/10.1103/PhysRevE.87.012707.

[66] Roshan Rao et al. *Transformer protein language models are unsupervised structure learners*. Dec. 2020. URL: https://doi.org/10.1101/2020.12.15.422761.

[67] Alex Hawkins-Hooker et al. "Generating functional protein variants with variational autoencoders". In: *PLoS Computational Biology* 17.2 (2021).

[68] Adam Paszke et al. "PyTorch: An imperative style, high-performance deep learning library". In: *Advances in Neural Information Processing Systems*. Ed. by H Wallach et al. Vol. 32. Curran Associates, Inc., 2019, pp. 8024–8035. arXiv: 1912.01703. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[69] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html.

[70] Xinqiang Ding, Zhengting Zou, and Charles L. Brooks III. "Deciphering protein evolution and fitness landscapes with latent space models". In: *Nature Communications 2019 10:1* 10.1 (Dec. 2019), pp. 1–13. URL: https://www.nature.com/articles/s41467-019-13633-0.

[71] Adam L. Martin, Michael A. Steurer, and Robert S. Aronstam. "Constitutive activity among orphan class-A G protein coupled receptors". In: *PLoS ONE* 10.9 (Sept. 2015), e0138463. URL: www.cdna.org.

[72] Jeremiah D. Heredia et al. "Mapping Interaction Sites on Human Chemokine Receptors by Deep Mutational Scanning". In: *The Journal of Immunology* 200.11 (June 2018), pp. 3825–3839. URL: http://www.jimmunol.org/content/200/11/3825.

[73] Eric M. Jones et al. "Structural and functional characterization of G protein-coupled receptors with deep mutational scanning". In: *eLife* 9 (Oct. 2020), pp. 1–52.

[74] Wallace K. B. Chan et al. "GLASS: a comprehensive database for experimentally validated GPCR-ligand associations". In: *Bioinformatics* 31.18 (Sept. 2015), pp. 3035–3042. URL: https://academic.oup.com/bioinformatics/article/31/18/3035/240608.

[75] Leland McInnes, John Healy, and Steve Astels. "hdbscan: Hierarchical density based clustering". In: *Journal of Open Source Software* 2.11 (Mar. 2017), p. 205. URL: https://joss.theoj.org/papers/10.21105/joss.00205.

[76] Michael Steinbach, Levent Ertöz, and Vipin Kumar. "The Challenges of Clustering High Dimensional Data". In: *New Directions in Statistical Physics*. 2004, pp. 273–309.

[77] Leland Mcinnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. Tech. rep. 2020. arXiv: 1802.03426v3.

[78] Asuka Inoue et al. "Illuminating G-Protein-Coupling Selectivity of GPCRs". In: *Cell* 177.7 (June 2019), 1933–1947.e25. URL: https://pubmed.ncbi.nlm.nih.gov/31160049/.

[79] Rafael Gómez-Bombarelli et al. "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules". In: *ACS Central Science* 4.2 (2018), pp. 268–276. arXiv: 1610.02415. URL: https://pubs.acs.org/sharingguidelines.

[80] David Brookes, Hahnbeom Park, and Jennifer Listgarten. *Conditioning by adaptive sampling for robust design*. May 2019. URL: http://proceedings.mlr.press/v97/brookes19a.html.

[81] Francesco Locatello et al. "Challenging common assumptions in the unsupervised learning of disentangled representations". In: *RML@ICLR 2019 Workshop - Reproducibility in Machine Learning* (2019).

[82] Francesco Locatello et al. "Disentangling Factors of Variation Using Few Labels". In: (2019), pp. 1–24. arXiv: 1905.01258. URL: http://arxiv.org/abs/1905.01258.

[83] Jung Eun Shin et al. "Protein design and variant prediction using autoregressive generative models". In: *Nature Communications* 12.1 (2021), pp. 1–30.

[84] Chloe Hsu et al. "Combining evolutionary and assay-labelled data for protein fitness prediction". In: *bioRxiv* (Mar. 2021), p. 2021.03.28.437402. URL: https://www.biorxiv.org/content/10.1101/2021.03.28.437402v1%20https://www.biorxiv.org/content/10.1101/2021.03.28.437402v1.abstract.

Appendices

A EVE model

The Evolutionary model of Variant Effects (EVE) by Frazer, Notin, Dias et al. [17] is used to obtain our latent representations for the clustering analysis, and extended in our semi-supervised experiments.

A background to the EVE model is provided in Section 2.3.2, and further details are included below.

A.1 EVE score

The authors remarked that the evolutionary indices (as introduced in the main text under Subsection 2.3.2) happened to separate benign from pathogenic labels, suggesting that unsupervised clustering methods could be used to separate these classes. A two-component global-local Gaussian Mixture Model (GMM) is fitted to these evolutionary indices across all proteins. The output of this GMM is an 'EVE score' for each single amino-acid variant - a continuous measure of pathogenicity that can be used for binary classification. It suffices to mention here that a GMM is simply a type of unsupervised clustering algorithm.

Due to using a Bayesian decoder in the VAE model architecture, it is also possible to compute the predictive entropy for each of these variants as a measure of uncertainty. The authors thus finally use the 'EVE scores' together with predictive entropy to classify variants into either 'Benign', 'Uncertain' or 'Pathogenic'.

A.2 EVE architecture

EVE included a number of changes to the DeepSequence architecture [14] after an extensive ablation study. These are (from [17]):

- Symmetrization of the encoder and decoder network architectures;
- Increased number of layers and increased layer width for the encoder and decoder networks (2, 000 - 1, 000 - 300 and 300 - 1,000 - 2,000 respectively);
- Increased size of the latent space (50);

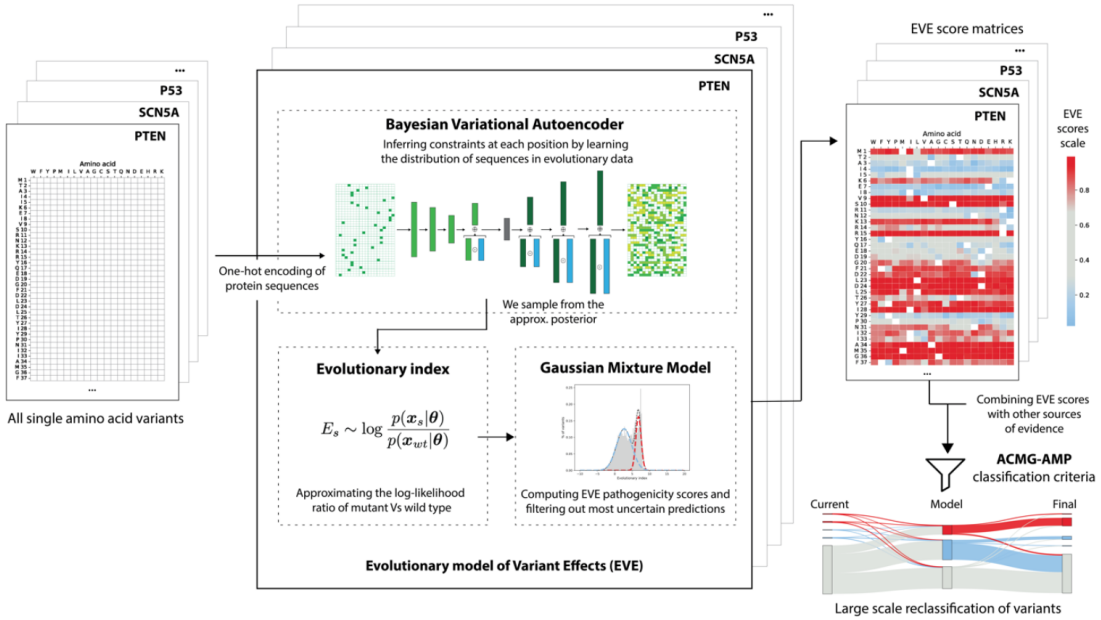


Figure A.1: The EVE architecture, taken from the original paper [17]. After being trained on an MSA of a protein of interest, the model takes as input all single acid amino acid variants in one-hot-encoded format. For each amino acid variant, the model outputs an 'evolutionary index' (described in text). These evolutionary indexes for all variants can be clustered to yield EVE scores (centre). EVE scores can be combined with other sources of evidence to separate benign, uncertain and pathogenic classes in an unsupervised manner (right).

- Larger number of training steps to train the more complex architecture (400, 000 training steps);
- Lower learning rate to stabilize learning process (10^{-4});
- Removal of the group sparsity priors, responsible for significant performance drops for certain proteins;

The full ablation study behind these choices is included in the original paper. Therefore, the full architecture and hyperparameters of the EVE model are as follows (available at https://github.com/OATML/EVE/blob/master/EVE/default_model_params.json):

- Encoder hidden layer widths: [2000,1000,300]
- Encoder activation function: ReLU
- Encoder dropout probability: 0.0
- Latent dimension size: 50
- Decoder hidden layer widths: [300,1000,2000]
- Decoder activation function: ReLU
- Decoder dropout probability: 0.1
- Decoder convolution depth: 40

The decoder also applies a '1x1 convolution' to the output of the last hidden layer.

A.2.1 1x1 convolution

We will describe 1x1 convolutions in the context of sequences (as we will use in the VAE), although 1x1 convolutions can be applied to 2D image data with multiple channels (or higher dimensions).

A 1x1 convolution on a tensor of size $[L, C]$ (where L is the length of the sequence and C the number of 'channels'/dimensions per token in the sequence) simply applies the same linear projection to each token in the sequence, typically to reduce the number of channels C to a smaller C_{out}.

Parameterizing this linear projection by a weight matrix W of size $[C, C_{\text{out}}]$ would yield the following equation for a 1x1 convolution on sequence S_{in}:

$$\underset{L\times C_{\text{out}}}{S_{\text{out}}} = \underset{L\times C}{S_{\text{in}}} \times \underset{C\times C_{\text{out}}}{W} \tag{A.1}$$

Which is equivalent to applying the W to each token ${S_{\text{in}}}^{(i)}$ in the sequence:

$$\underset{1\times C_{\text{out}}}{{S_{\text{out}}}^{(i)}} = \underset{1\times C}{{S_{\text{in}}}^{(i)}} \times \underset{C\times C_{\text{out}}}{W} \qquad \forall i \in [1, L] \tag{A.2}$$

In this case, the last hidden layer dimension (2000) is projected via a fully-connected layer to a dimension ($d = 40L$), which is then reshaped to $(L \times 40)$, where L is the length of the amino acid sequence. The 1x1 convolution reduces this depth down to $(A < 40)$ for a final size of $(L \times A)$ for each sequence, with A being the size of the alphabet, or the number of amino acids. This recovers the original shape of the input sequence.

Training parameters:

- Training steps: 400000,
- Learning rate: 1e-4,
- Optimizer: Adam
- Batch size: 256
- KL latent scale: 1.0
- KL global params scale: 1.0
- L2 Regularization: 0.0

The KL latent scale and KL global parameters scale refer to the weights of the latent space KL divergence term and decoder KL divergence term in the loss function respectively (as shown by the ELBO in Figure 2.10).

Dropout [64], where applied, is applied to each hidden layer.

A.2.2 Input/output shapes

Figure 2.10 shows the EVE VAE architecture with inputs and outputs, but the details may be difficult to follow.

Flattening each sequence in the batch yields size `[B, sequence_length * alphabet_size]`, and each element in the batch is processed independently by the network.

For illustration, the first layer will have an input size of (`sequence_length*alphabet_size`) and an output size of 2000, so a flattened input batch of size `[B, sequence_length*alphabet_size]` will be passed through the layer to yield an output size of `[B, 2000]`.

The output of the network is again a tensor of size `[B, sequence_length*alphabet_size]` which is finally reshaped to a tensor of size `[B, sequence_length, alphabet_size]`.

A.3 Results

EVE accurately predicted clinical significance labels (benign/pathogenic) for a set of 1,081 human genes that have been associated to human disease. To evaluate the model, the authors compute the area under the receiver operating characteristic curve (AUC) and accuracy with respect to clinical labels for the 1,081 proteins (both in aggregate and per protein). When compared with clinical labels, EVE pathogenicity scores achieved an aggregate AUC of 0.90 and mean AUC per protein of 0.92.

The authors found that EVE outperforms several state-of-the-art supervised pathogenicity predictors on AUC and accuracy. These results are also robust to number of labels per protein, unlike supervised methods which rely on an adequate number of labels per protein to train.

The original data and results can be found at evemodel.org, along with detailed information on every gene and variant of interest.