To my parents, for loving me unconditionally and always supporting me

To Alicja, for giving me the most fulfilling relationship of my life

To my dog Sole, for reminding every single day of the importance of sleep and play

Table of Contents

1. Introduction *2*

Proteins 2

Protein Mutations 6

2. Related work and Research Question *9*

Literature Context 9

DeepSequence and latent variable models 12

AlphaFold 14

Research Question 15

3. Methods *18*

High level flow of data 19

AlphaFold's MSA representation 20

Pytorch implementation of DeepSequence 21

Variational Autoencoders 23

Bayesian Decoder 23

4. Training Details *24*

5. Results and Analysis *31*

Comparison of the models: baseline vs. augmented VAE 31

Correlation analysis of the MSA representation 33

Experiments using Neural Networks 36

6. Conclusions *40*

References *41*

1. Introduction

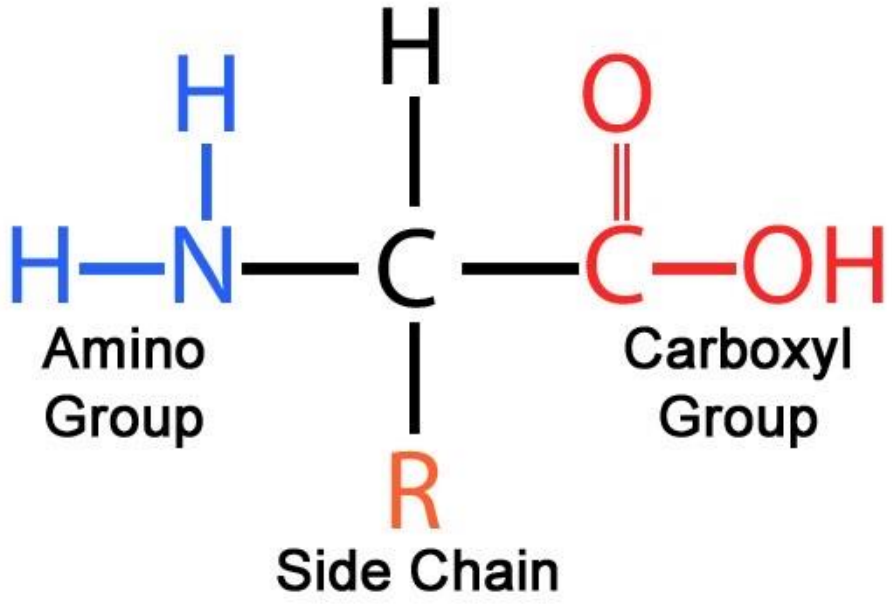
Proteins

Among the classes of biological macromolecules, proteins are the most versatile as they perform a vast array of functions within organisms. To name a few, proteins are responsible for proving structure and for carrying out specific functions like transporting other molecules and catalyzing reactions that otherwise would not occur. Proteins play a major role in response to stimuli by engaging in the complex process of cell signaling. Furthermore, proteins are also fundamental in every phase of cell division and DNA replication and translation. In other words, if something happens within a cell or an organism, proteins are always involved at least to some extent. For this reason, they are usually referred to as 'working molecules' or 'molecular machines'.

Proteins, much like a train made up of a sequence of wagons, are characterized by a sequence of amino acids linked together sequentially through a peptide bond, a link between the end of one amino acid and the beginning of another. More formally, every amino acid is characterized by 3 sites: an *amino* (-NH2) group, a *carboxyl* (-COOH) group, and a *side chain* usually referred to as R. The peptide bond is formed through a *condensation reaction* between the amino group of one amino acid and the carboxyl group of another amino acid. By joining more monomers together, proteins can be seen as biological polymers where the amino acids are its building blocks.

There are more than 500 amino acids found in nature, but the proteins belonging to the human proteome are made up of only 20 different building blocks. Amino acids differ only for their side chain R: the simplest monomer is Glycine, where R is simply an hydrogen atom. Much like the

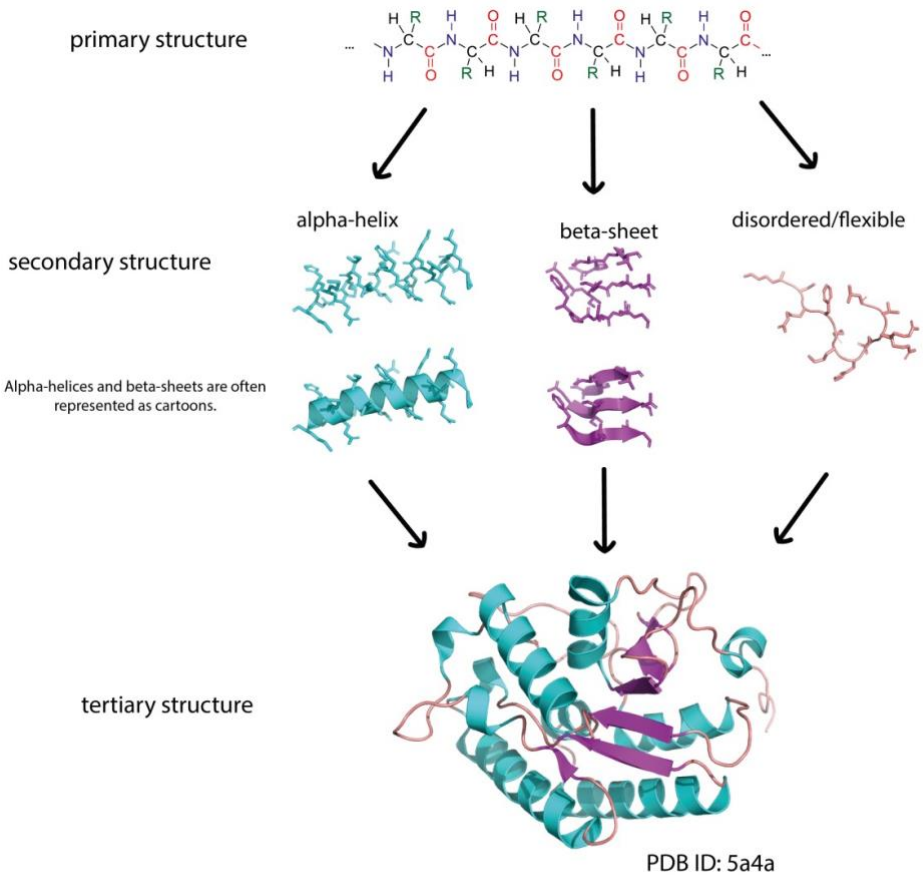
letters of the alphabet can create the infinite complexity of words, sentences and ideas, the limited set of the 20 fundamental amino acids, stacked one after the other, can create all the proteins responsible for life itself. Predicting the behavior, or even the stability, of a protein is an open problem in modern Biology.



Amino acid structure. (Source: Amino Acids and Their Production during the Photolysis of Astrophysically Relevant Ices. Astrochem.org. http://www.astrochem.org/sci/Amino_Acids.php.)

To understand the function of a protein, its most important characteristic is its shape. Much like a key can only open a lock if it first fits in the lock itself, biological reactions happen between big and complex molecules, and it is not enough that the molecules have sites that can react together: they also need to fit properly with each other. When a molecule loses its functional shape because of changes in the environment its efficacy in carrying out its function – called *activity* – can diminish dramatically, even though the sequence of amino acids itself is left untouched. This phenomenon is called *denaturation*. An example of denaturation happens when people with curly hair iron them straight by breaking the disulfide bridges in Keratin, a fibrous structural protein presents in hair, but also nails and claws or horns of animals.

The study of the shape of a protein starting from its amino acid sequence is called *protein folding*. Contrary to the analogy of a protein as a train made up of wagons, each amino has an electric charge – or lack of it – which leads some sites of the polymer to repel each other, and other sites to attract each other. Most importantly, protein folding is a spontaneous process in which the chain folds itself to find the most stable shape: the one characterized by the lowest energy. Much like two magnets that are thrown in the air, they will orient *naturally* so that opposite poles match and attach themselves to minimize the repulsive force of same-sign-poles and maximize the attraction of opposite-sign-poles. In the same way, the *Anfinsen's dogma* states that the shape of a protein is the *natural* result of its so-called primary structure, which is the sequence of amino acids that make up the protein.



Protein folding process. (Source: From gaming to cutting-edge biology: AI and the protein folding problem. Scienceinschool.org. https://www.scienceinschool.org/article/2021/gaming-cutting-edge-biology-ai-and-protein-folding-problem/)

Protein folding happens at different levels of complexity. The first step in the folding process is the formation of the secondary structure from the primary one. Intramolecular hydrogen bonds between the monomers are responsible for the formation of two basic structure, *alpha helices* and *beta sheets*, that give rise to the secondary structure. Because both alpha helices and beta sheets are *amphipatic*, the tertiary structure emerges by isolating in the core of the protein its hydrophobic sites and exposing on the outside its hydrophilic components, as the environment in which proteins are present is usually aqueous. As mentioned before for the Keratine, the tertiary structure can be further stabilized by the presence of disulfide bridges, which are covalent bonds. The tertiary structure is the final shape of a protein made up of a single polymer, but some proteins

emerge as the union of more polymers combined, giving rise to the quaternary structure. An example of a protein with a quaternary structure is Hemoglobin, where the protein is made of 2 couple of protein sub-units.

Protein Mutations

Very related to the problem of protein folding is the problem of mutation analysis, which consist in being able to determine whether a mutation at a specific position of a protein – or other biological polymers like RNA – will disrupt its function or not. Understanding the relationship between a protein sequence and its phenotype is a central step for designing new "smart" proteins for a variety of use cases potentially as vast as the roles of proteins themselves in living beings.

As stated by the *Central Dogma of Molecular Biology*, there is a direct flow of information from DNA to RNA through a process called *Transcription*, and from RNA to protein through a process called *Translation*. Usually, when referring to a mutated protein, it is intended a protein produced by a mutated gene, that encodes said protein. In this context, instead, a protein mutation refers to any change from a focus protein that will serve as reference. Possible mutations are deletion and addition, where an amino acid is removed or added, or swaps between different amino acids at a given position in the sequence.

Understanding protein mutations means understanding the underlying evolutionary processes that lead to the proteins that today are found in nature. Mutations, or more generally mistakes in biological processes, are the driving force of the evolutionary process which adds a random component without which the status quo would prevail unchallenged forever. When a random mutation is introduced in a system, its effects will

be propagated to the entire population through natural selection, so that only beneficial mutations survive in the long run.

A perfect example to talk about mutations and natural selection is Covid-19. For a virus, to be successful means to spread as much as possible without killing its host, because that would be the scenario most convenient for the virus. Mutations that lead toward that outcome will be tend to "win" simply because they virus will replicate more in such scenarios and eventually outnumber the original version of the virus.

As stressed already, the shape of a protein is fundamental for its function, and consequently for the survival of the organism. If mutations happen to change the sequence of amino acids in the Hemoglobin such that its shape would be altered, the protein would not be able anymore to carry Oxygen through the bloodstream as it was supposed to, so this mutation would be eliminated by natural selection. It is possible that a random mutation could make Hemoglobin even more efficient in its job, so that it would spread to the next generations, but it is far more likely for a new protein mutation that survives the natural selection process to be another arrangement of amino acids that folds into the same shape, or very similarly. In other words, while the primary structure changes and evolves over generations, its final shape tends to change much more conservatively. By looking at the differences in a protein between different organisms, it is possible to find examples of different sequences of a protein with the same function, which very likely fold in extremely similar shapes. Leveraging *homologous sequences* is a key point in both protein folding and mutation analysis.

Different organisms, when compared to each other, have both similarities and differences. Similarities originate from the common ancestor, as it is

now believed that all the organisms present today originate from a common ancestor. The differences, instead, are a result of the random mutations and the effect of the selection process since the two different organisms have diverged. Almost all vertebrates have the Hemoglobin molecule, but each different organism is likely to have each own version of the protein that carries out the same role. In other words, for each protein that is shared by multiple organisms, it is possible to create a collection of sequences that encode for the same shape, or at the very least the same function. These sequences are called *homologous sequences*.

2. Related work and Research Question

Literature Context

Although modern deep mutational scans allow biologists to perform experiments at impressive rates – up to 10^5 unique variants of a protein per experiment[1] – the possible mutations that a protein could be subjected to are immense. Statistical and computational models can complement the work done in the lab by distilling the knowledge from the results of the experiments.

As stressed in the previous section, protein mutations naturally occur in nature and can be found by comparing functionally equivalent proteins in different organisms. In fact, one could argue that Nature itself is the vastest experiment ever performed to our knowledge, and it is still ongoing. Since the dawn of life, evolution has performed its own mutation and selection experiments.

Over the past decades, computational methods have been developed to make sense of the diversity of mutations naturally occurring. A first intuitive strategy for predicting the effect of mutations is by exploiting the evolutionary conservation of specific sites within a protein. Consider the following examples: assume that while comparing proteins that carry out the same specific function in different organisms, after aligning the sequences to compare them properly, at a specific position *p1* always the same amino acid is found. On the other hand, at another position *p2* different amino acids are found in different homologous proteins. In such a scenario, it seems natural to assume that the amino acid at position *p1* is important for the fitness of the protein, otherwise there would be no

1 Starita L, Fields S. Deep Mutational Scanning: A Highly Parallel Method to Measure the Effects of Mutation on Protein Function. *Cold Spring Harb Protoc*. 2015;2015(8):pdb.top077503. doi:10.1101/pdb.top077503

reason to not observe a different mutation at *p1*. On the contrary, the amino acid at the site *p2* is somewhat free to mutate without disrupting the function of the protein. In other words, because Nature is carrying out its own experiment and we do not observe any protein with a different amino acid at position *p1*, it is extremely likely that such a protein would fold is a shape such that the function could not be carried out.

While such site independent models – that consider the likelihood of a protein only based on the probability of each amino acid in their respective position – are useful, reality is unfortunately often more complex. For example, in genetics it is widely known that very few characteristics are related to the presence or not of a single gene. Most likely, the phenotype is the result of the interaction of multiple genes, which makes hard to predict the effects of changes in the genome. For proteins, the settings are very similar: because of the way protein folds, amino acids that are very far apart in the chain can actually be very close to each other spatially. Some pairs of amino acid might be more fit to each other to be close spatially than others, so looking just at the probability of an amino acid at a specific site is simply not enough because all the context is loss. Consider another example: assume that, like before, researchers are comparing proteins that carry out the same specific function in different organisms. While focusing on a specific site *p1*, from the sequences in the sample there is a 50% probability the position will be occupied by the amino acid *A* and 50% probability by amino acid *B*. This information is definitely valuable as there are 20 possible amino acids and a model based on this statistic would deem fit only proteins in which either amino acid *A* or amino acid *B* occupied *p1*, therefore discarding a lot of alternatives. On the other hand, if one were to consider not only site probability but pair interaction, it might happen to notice a striking correlation: whenever amino acid *A* is at *p1*, amino acid *X* is at another position *p2*. Also, whenever amino acid *B* is at *p1*, amino acid *Z* is at *p2*. This correlation might be due to the fact that, after

folding, sites *p1* and *p2* are very close to each other and form a bond. In such settings, if one of the two amino acids were to change, the only way for the protein to retain the same shape is for the relationship between *p1* and *p2* to stay the same. Sometimes, a single amino acid can change without affecting the shape of the entire molecule, but other times also the sites that closely interact with the mutated one have to change in order to account for the difference.

For the sake of simplicity and without loss of generality in order to convey why a single site in a protein might not be independent of the rest of the protein, it is possible to view amino acids as magnets once again. Assume now that the magnets at positions *p1* and *p2* have opposite charge, and therefore form a bond and attract each other. If a mutation would happen changing the charge at *p1*, now the couple of amino acid would have the same charge, so they would repel each other and change the shape of the molecule. The only way for the mutation to lead to a stable homologous protein if for the site at *p2* to also change its charge. This example is very simplistic for a multitude of reasons: first of all, some amino acids have a charge, but they cannot be reduced to simple magnets, although it can be stated that some amino acids have higher or lower affinity for others. Secondly, it might be that *p1* and *p2* are not actually independent either as a pair, and that the interaction could be even more complex and involving more sites.

The complex interaction between sites in a protein is a phenomenon called *epistasis* and modern mutation analysis models take it into account explicitly for achieving better predictive performance. Once again, more precisely: residues mutations in a protein are not independent, therefore it is paramount to take into consideration the genetic interaction between mutations and the sequence background.

A natural extension of the site independent model, also following the logic of the previous examples, is to model all possible pair interaction instead of all single sites. Such a model, proposed by [2], is called the EV method and takes account of epistasis, but only up to second order interaction. The biological literature seems to indicate that higher order epistasis is not only present but pervades in their importance in determining the phenotype of a protein. It follows that a pairwise interaction model can never capture the big picture of mutation analysis, but still achieve results that make it extremely useful.

One could argue that, if the EV method only satisfies second order epistasis and it is believed also higher order epistasis to be important for prediction purposes, why not simply try to capture all the higher orders of interactions between residues together. Unfortunately, such a model would be intractable because, every time a new order of interaction *k* is considered, the parameters needed to model are subject to a *combinatorial explosion*. To be more precise, the combinatorial explosion would happen to *n/2*, where *n* is the length of the sequence. Unfortunately, this additional piece of information does not solve the tractability of the model.

DeepSequence and latent variable models

To overcome the inability to parameterize a model which consider all possible interactions, *latent variable models* present a new framework to model protein mutations. Instead of considering the observed position-position variants to model the correlation between residues, the idea is to assume the existence of a set of hidden variables that affect the observed

2 Hopf T, Ingraham J, Poelwijk F et al. Mutation effects predicted from sequence co-variation. *Nat Biotechnol*. 2017;35(2):128-135. doi:10.1038/nbt.3769

genotype and to try model these latent variables instead. Latent variable models can provide the flexibility to model higher-order interaction without considering them explicitly, but at the cost of loss of interpretability of the results. In fact, in this context, a mutation prediction is not explained anymore from observed pattern in a protein which can be compared to all the other similar proteins in nature. Instead, with a latent variable model, the fitness of a mutation is determined by how the sequence satisfies constraints which are assumed to be underlying of the evolutionary processes.

To understand latent variable models, it is useful to draw a comparison with the simpler Principle Component Analysis. PCA is commonly used in data analysis to reduce the dimensionality of a dataset. Especially when not too many observations are available, if a dataset consists of a lot of columns – commonly referred to as *features* – one needs to make extra care not to incur in the so-called *curse of dimensionality*. In few words, the more features a model takes into account, the more observations are needed for the results to be statistically significant. To be clear, it is this inability to find a reliable pattern in higher dimensional spaces that makes the combinatorial explosion of the parameters an issue when trying to model higher order interaction. When there are so many parameters to estimate with a reduced sample size, it becomes more and more likely to pick up random noise as part of the pattern, actually reducing the predictive power of the model.

The curse of dimensionality is very common in all kinds of biological problems: for example, if one wants to model the likelihood of a disease based on the entire human DNA, the model would consider so many features that any pattern would be feasible unless with an incredibly large dataset, or relying only on astoundingly significant statistical patterns.

Similarly, in protein mutations, if one were to work with all possible interaction of order below k, the features would simply be too many to consider with the availability of naturally occurring sequences to have statistically significant results. To ease the problem would be to reduce the number of features without sacrificing too much the information carried out by them. This is exactly what PCA does in a nutshell: given a dataset of n observations and k features, it tries to find p new features – with $p < k$ or even $p << k$ – which explains as much of the variance as possible.

The idea is to model all possible interactions because the biological literature states clearly that those interaction can and do play a role in determining biological processes, including the overall stability of a protein and its function. Unfortunately, as previously stated, parameterizing such a model is not feasible. Therefore, the question is: is it possible to reduce the dimensionality of such a problem? In other words, if one were to consider all possible interactions as features of the datasets, is it possible to find new latent features to use instead to model the same information, while using less parameters?

DeepSequence is a latent variable model implemented as a Variational AutoEncoder, which differs from PCA for two reasons: firstly, contrary to PCA which is a linear mapping, a VAE implements a non-linear transformation using a Neural Networks. Secondly, specifically for DeepSequence, the conditional distribution of the data is categorical rather than Gaussian in order to model discrete characters.

AlphaFold

AlphaFold is an algorithm that gained popularity in 2018 by placing first at the 13th Critical Assessment of Techniques for Protein Structure Assessment (CASP), a bi-annual experiment in which more than 100

research groups compete to predict the spatial structure of a newly crystallized protein from its amino acid sequence. In 2020, AlphaFold 2 (AF2) scored so impressively that some experts even declared the problem of protein folding to be solved. Other voices in the scientific community points out that the shapes of up to a third of the protein of the CASP assessment are still not predicted to a satisfactory level and that the model is not able to point towards the actual mechanisms involved in protein folding, making of AF2 "only" a black-box model. Nevertheless, AF2 is an impressive technical achievement, and it can propel much more scientific output since not only the code has been open-sourced, but also the weights of the model, giving to any researcher the possibility of running experiments with AF.

Protein folding and mutation analysis are extremely closely related problems, as they both depend on the shape of the protein. As such, AF2 and other protein folding models use the same strategy used in mutation analysis: instead of considering just one amino acid sequence, is much more effective to first search the wider family to which the focus protein belongs to. In the case of AF2, the output of the search if a Multiple Sequence Alignment (MSA). The MSA is used to find sequences closely related to a focus one. For DeepSequence, the MSA is used to understand the latent evolutionary constraints in the protein family. For AF2, the MSA is used to help the algorithm with similar sequences whose shape is known.

Research Question

The research question of this thesis is to investigate the link between models for protein folding and mutation analysis. AF2 certainly represents a turning point in computational biology and much more research needs to be done to understand the full potential of these models for the

understanding of biological processes and phenomena. By joining AF2 and DeepSequence, this work has the aim of finding more informative protein representations and their effectiveness.

In the past decade, the Artificial Intelligence community has witnessed the incredible surge of self-supervised learning (SSL). One of the main advantages of SSL has been the creation of better and semantically-meaningful representations of a variety of different modalities of data. For a long time, the field of Natural Language Processing (NLP) used one-hot encoding as the main strategy to represent textual data. One-hot encoding is extremely straightforward, but it leaves a lot of room for improvement. Only in 2012, [3] introduced the concept of words embedding with Word2Vec. The main idea behind the paper is to let a Neural Network do the work of creating better representation system, by designing a problem in which the model has to compare words and understand the relationship between them. In simpler terms, instead of designing a way to embed words just to be able to distinguish one from the other, an embedding should also be informative of what it is describing. In the case of Word2Vec, the result was a semantically meaningful word embedding system. What this means is that other than just encoding the words "good" and "bad" as vectors, it is much more useful if such embeddings are able to capture that "good" and "bad" are opposite words, while "good" and "excellent" are very related concepts.

Today, the field of computational biology resembles what NLP was 10 years ago and there is still the need to find a way to encode amino acid sequences such as to retain important properties or relationships between different proteins. This thesis investigates the effectiveness of the AF's

3 Mikolov T, Chen K, Corrado G, Dean J. *Efficient Estimation Of Word Representations In Vector Space*. Mountain View, CA; 2013. https://arxiv.org/pdf/1301.3781.pdf. Accessed September 8, 2022.

MSA representation for mutation analysis, much like Word2Vec used the hidden representation of a network as word embeddings.

3. Methods

The goal of this section is to explain the integration between DeepSequence and the AlfaFold's protein representation. Most importantly, the flow of data between the two models is the key to understand how the experiments have been carried. All the code for this work is available for better understanding and reproducibility at [4].

The DeepSequence model is open-sourced and available at [5]. For the purpose of this thesis, it was convenient to re-write the relevant sections of the model in Pytorch, as the original DeepSequence is written in Theano. The model is not pre-trained because it is trained specifically for each family of proteins. Details on the Pytorch implementations are discussed later in this chapter.

As previously stated, the AF2 model and weights have been open sourced by DeepMind and they are available at [6]. Nevertheless, this thesis uses OpenFold, which is *"A faithful but trainable PyTorch reproduction of DeepMind's AlphaFold 2",* available at [7].

The main hypothesis behind this work is that the hidden protein representation from AF2 is a better encoding for a protein than a simple one-hot encoding of the amino acid sequence. Therefore, the goal is to train two versions of DeepSequence that differ only on the representation of the proteins in the MSA.

4 https://github.com/egruttadauria98/MScThesis_mutation_analysis#readme
5 https://github.com/debbiemarkslab/DeepSequence
6 https://github.com/deepmind/alphafold
7 https://github.com/aqlaboratory/openfold

High level flow of data

To understand the integration between DeepSequence and AF2 it is useful to focus on a high-level view of the flow of the data.

First, the MSA is computed. The MSA is a computational procedure which aims at finding all the homologous sequences of a given starting focus sequence.

The repository of OpenFold offers a Colab notebook that can be easily run to try a lighter version of the model on the cloud. The major advantage of using the Colab notebook is that all the genetic databases needed for the MSA to look for homologous sequences can be used on the cloud instead of downloading them locally.

Once the MSA is available, it is possible to train the baseline DeepSequence model. As stated before, the goal of the study is to compare the effectiveness of the AlphaFold's hidden representation, compared to a more classic one-hot encoding representation of the protein as an amino acid sequence. Therefore, the baseline model is a DeepSequence model trained with the MSA, in which each protein is represented with a one-hot encoding.

The one-hot encoding represents each protein by a matrix of shape (length of the focus sequence, 20). All the elements of the matrix are 0, except the index of the amino acid at each position in the protein. In other words, for each row of the matrix, there is at most a value 1 which corresponds to the index of the column in the matrix of the correct amino acid. Therefore, the one-hot representation of the entire MSA is a 3D matrix of shape (number of sequences in the MSA, length of the focus

sequence, 20). This representation is clearly sparse and non-directly informative of any properties of the protein.

The same MSA can be given also to the AF2 model to obtain the MSA representations of the proteins in the MSA. This hidden representation is nothing more of a side product of the actual goal of AF2, which is predicting the 3D structure of the protein. Nevertheless, such a side product might encode protein information in a semantically meaningful way, which could be useful for a lot of downstream tasks.

Finally, the augmented DeepSequence model is trained with the MSA representation and it is benchmarked with the baseline model.

AlphaFold's MSA representation

One of the components of AF2 is a transformer block – named *Evoformer* – that takes as one of the inputs a representation of the MSA. The output of this transformer block will then be the input to the structure model, which is in charge of actually determining the spatial structure of the protein. This whole procedure is repeated three times, under a process called recycling. To obtain the representation to train the DeepSequence model, one needs to extract the MSA representation after the last recycling.

The shape of the MSA representation, once extracted, is (512, length of the focus sequence, 256). In other words, the MSA representation consists of 512 sequences represented as 256 channels for each position of the original amino acid sequence. The reason why only 512 sequences are present in the matrix is because the computational complexity of the transformer is quadratic in number of sequences. It follows that if the MSA representation kept all the sequences, inference time would become

infeasible. To obviate to this problem, the AF2 model samples 512 different sequences at each recycling.

In order to obtain a dataset of the MSA representation, AF is run 10 times and each time the MSA representation of 512 sequences is saved, along with the index of the selected sequences for the MSA representation. Collecting the index of the sequences in the MSA representation is paramount to link the reach representation with the sequence of the MSA it belongs to.

As remarked before, normally AF goes though 3 recycling steps. It is also important that, at each recycle, the same sequences are sampled. This way, one can be sure that the hidden MSA representation does not depend on different sequences – which is instead the behavior AF was designed for. Alternatively, one could force AF to perform 0 recycling to be absolutely sure that the sampling of sequences for the MSA representation happens only once, and that the representation does not depends on a mixture of different sequences. Both options have been tested, but the MSA representation after 3 recycling steps has proved to perform best.

Pytorch implementation of DeepSequence

Pytorch is one of the most used deep learning frameworks in use, and is currently the most widely adopted within the academic community. During the initial experiments while working with DeepSequence, it appeared clear that Theano required a more convoluted and less “pythonic” syntax, making it hard to study and re-purpose the model as intended. Furthermore, Pytorch offers an extremely straightforward integration with GPU training, extensively used within this thesis.

In re-writing the code in Pytorch, it was decided to keep the same structure of the original codebase. The goal, other than making the use of the model more productive for this work, is to provide an easy to use alternative for the computational mutation analysis community.

The DeepSequence codebase is divided in three main Python files: helper.py, model.py and train.py. The first one contains the DataHelper class, which helps in loading and organizing the alignment data, but also contains functions to make predictions about mutations. Containing the definition of the model, model.py specifies the architecture of the Bayesian Variational AutoEncoder, as well as the computation of the loss function to make the model converge during training. The latter file, train.py, contains instead the training loop used to train the model.

As this thesis is focused on establishing the effects of using the AF2's protein representation instead of the more simplistic one-hot encoding representation of the amino acid sequence, the codebase has been simplified by removing functions deemed non necessary, with the intent of slimming down the length of the code and making it also more intuitive. As an example, while the original DataHelper class was designed to also manage RNA sequences, this implementation is instead specific to proteins.

The core model of DeepSequence, a Variational Autoencoder with Bayesian decoder, has been reproduced faithfully based on the DeepSequence paper.

Variational Autoencoders

Variational AutoEncoders (VAEs) are model which aim at learning an identity function, which consist in mapping the input into itself. Contrary to other Neural Network architectures, the goal of a VAE is not to approximate an unknown function, but to find a meaningful representation of the input data: within the identify function, the data is first mapped to an intermediate representation – also known as *latent* or *hidden representation* – which is usually designed as a bottleneck, meaning that the intermediate step has a lower capacity than the input data. In other words, VAEs are compression algorithms for data. In order to succeed in the identity mapping, the model needs to learn how to distill the right information to then reconstruct the input.

Bayesian Decoder

Among the functionalities of DeepSequence that have been kept in the Pytorch implementation, the option to use Bayesian layers for the decoder part of the VAE is the most notable one. Compared to normal layers in which weights take scalar value which are progressively finetuned though the backpropagation of the neural network, Bayesian layers allow to model a probability distribution over the model weights. Having a probability distribution instead of a scalar value makes the neural network – or in this case the decoder of the VAE – equivalent to an ensemble of infinitely many neural networks, which in turns allows to estimate the level of certainty over a particular prediction. On the other hand, non-bayesian neural networks allow only for point estimates. Bayesian neural networks can easily be implemented in Pytorch using the blitz library[8].

8 https://github.com/piEsposito/blitz-bayesian-deep-learning

4. Training Details

This chapter concerns the training details of the baseline and the augmented model. For more details, everything discussed in this chapter is implemented in [9].

VAEs are deep learning architectures trained through backpropagation in a supervised fashion, even though they can be also classified as self-supervised algorithms. The VAE makes predictions on a group of training observations, which is usually called a *batch*. Iteratively, batch by batch, the model adjusts its weights to improve the predictions, which are scored by the *loss function*. As such, the loss function can be used to give the right incentives to the model.

For VAEs, the loss function is usually a sum of two terms: a *reconstruction loss* and a term for the *regularization of the latent dimension*. For this application, the Mean Square Error (MSE) of the predicted one-hot encoding and the ground truth one-hot encoding of the protein is used as the reconstruction loss. The regularization term is the Kullback–Leibler (KL) divergence between the latent distribution and a standard normal distribution, which is set as a prior. The utility of using a standard normal distribution as a prior is to have a compact distribution to sample from. Furthermore, when the Bayesian decoder is used, there is a third term for the regularization of the distribution of the weights, which is also a KL divergence measure.

The parameters of the VAE are optimized to find the minimum of the loss function though an iterative procedure known as Gradient Descent, which works by updating each parameter in the opposite direction of its

9 https://github.com/egruttadauria98/MScThesis_mutation_analysis/blob/main/deepsequence_experiments.ipynb

derivative with respect to the loss function. For this work, Adam[10] was used as the optimizer. Adam is a stochastic gradient descent (SDG) optimizer, meaning it uses a stochastic approximation of the gradient of the loss function. To overcome the computational burden of the gradient of the loss function, SGD performs the update of the parameters based on a single training example and label that is randomly extracted at each iteration.

When summing the two components of the loss (or three for the Bayesian decoder case), it appeared useful to pre-multiply the KL divergence terms by a weight to aid training and give a better balance to the model between reconstruction loss and regularization. In the initial experiments, when the components of the loss where summed without any pre-multiplication, the loss would plateau very fast and the model would stop learning. The parameters used for the training plots below are kl_latent_scale=0.0001 and kl_weights_scale=0.001, where the first value is the scale that multiplies the KL divergence for the latent dimension, and the latter value is the scale of the KL divergence for the weights distribution for when the Bayesian decoder is used. The reconstruction loss is not multiplied by any scale. The scale parameters have been set at the highest value possible such that the reconstruction loss would not get stuck after the very first epochs.

To recapitulate, two different models are trained: a *baseline model* and an *augmented model*. The baseline model aims at reproducing the DeepSequence model but implemented in Pytorch and with slightly less functionalities which are not related to this work. The baseline model gets

10 Kingma D, Ba J. *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION*. Published as a conference paper at ICLR 2015; 2017. https://arxiv.org/pdf/1412.6980.pdf

as input the one-hot encoding for each protein with shape (length of the focus sequence, 20), where the length of the focus sequence equals 286 residues for the BLAT-ECOLX protein family on which the experiments are conducted on. For the augmented model, instead, the input is the MSA representation, and the output is the one-hot encoding of the MSA. Therefore, the augmented model works as an identity function between the two different representations of a protein.

There is no preprocessing for the baseline model, only for the augmented model. As the augmented model uses the MSA representation as input, its shape is (length of the focus sequence, 256) for each protein sequence, where 256 are the channels that AlphaFold uses to characterize each amino acid in the sequence. Because the goal is to compare the different representations without giving any advantage by adapting the architecture to the representation, it was decided to use PCA to reduce the dimensionality of the last dimension of the MSA representation from 256 to only 20. After the PCA, the input is standardized with a min-max scaling to reproduce the distribution of the one-hot encoding which is bounded between 0 and 1.

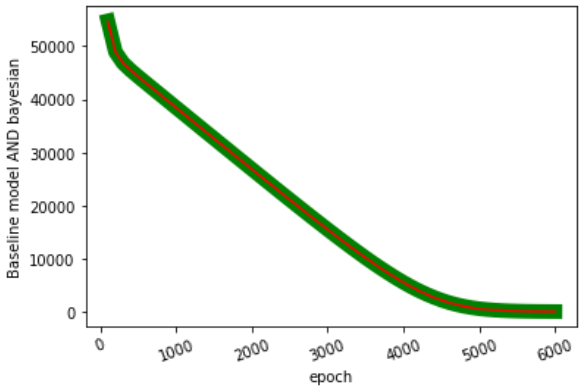
Both the baseline model and the augmented model have the same model architecture. The VAE consists of a 2-layers encoder of size 1500 and 1500 neurons each. The latent variable, which is the bottleneck, is of only size 40. The decoder is again a 2-layers architecture of size 100 and 2000 respectively. The activation functions are all Rectified Linear Units (ReLU) except the last one which is a Sigmoid function to output a probability distribution over the amino acids at each locus.

For each model, when the data-loader is created, it automatically performs a train/test data split with an 80:20 ratio. The test set is used

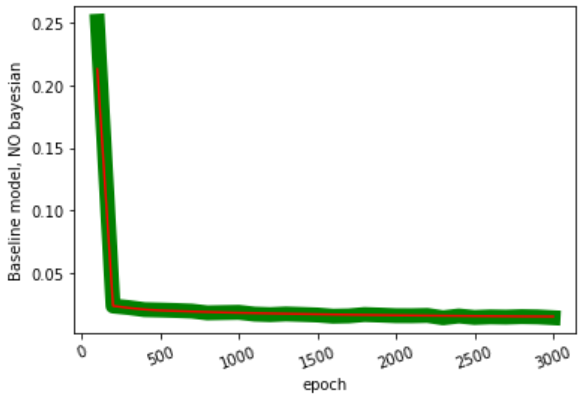
during training to check for overfitting by comparing the loss on the training set and the loss on the test set. Usually, if a model starts to perform better and better on the train set while decreasing its performance in the test set, it means the model is overfitting. Overfitting happens when a model is capturing noise within the training data as actual patterns, which not only do not generalize to the test set, but also imply a decrease in performance on it. Therefore, during training it is important to keep a balance between overfitting and underfitting: the goal is to not interpolate noise that does not generalize outside of the training data, but also not leave important patterns in the data undetected.

As discussed in the DeepSequence paper[11], one of the techniques that the authors deemed very important for regularization is the use of a Bayesian decoder, especially important because it complements the prediction with an uncertainty measure that stems from the distribution of the weights. The less variance the weights have in the decoder, the more the model is sure about the prediction. In fact, in this work the Bayesian decoder was used only on the baseline model and then discarded, as no overfitting can be noticed from the training plots. From the figures below, one can compare the training of the baseline model with and without the Bayesian decoder.

11 Riesselman A, Ingraham J, Marks D. Deep generative models of genetic variation capture the effects of mutations. *Nat Methods*. 2018;15(10):816-822. doi:10.1038/s41592-018-0138-4



Training plot 1: baseline model with Bayesian decoder

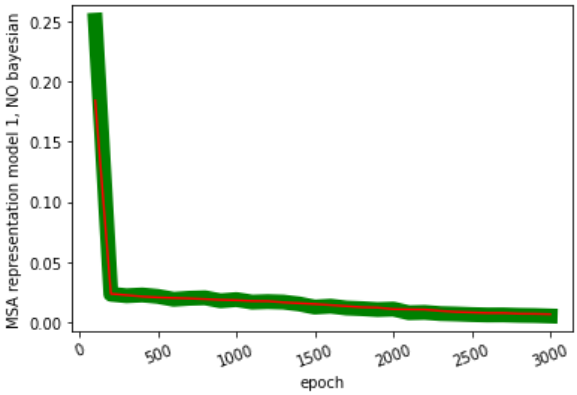


Training plot 2: baseline model without Bayesian decoder

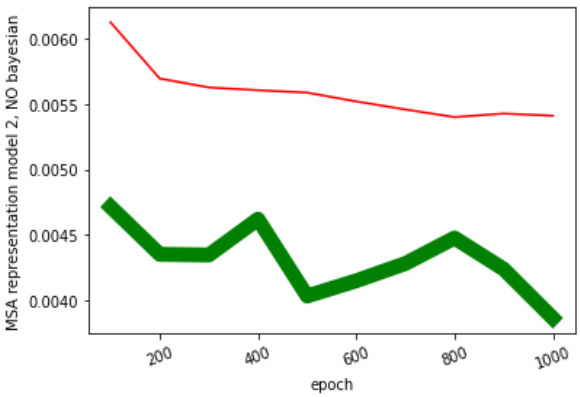
For all the training plots, the green line is the train loss and the red thin like is the test loss. As one can see, there is no overfitting in both cases, but training is much faster without the Bayesian decoder. As one of the reasons why overfitting might not be a concern as much as for DeepSequence, this work is mainly concerned with the BLAT-ECOLX protein family, which is one with the biggest MSA available. Bigger MSA means more homologous sequences, which are the training data that the VAE uses to understand the evolutionary constraints of the protein family. It is possible that the bigger dataset can overcome the need for a stricter regularization, which is needed for smaller protein families. This should not be a surprise, as it is widely known that Deep Learning models are extremely data-hungry and that a bigger dataset can make the different between overfitting and good generalization properties of a model. The lowest losses for the baseline model are around 0.015 for both train and

test loss without a Bayesian decoder, and around 33 with the Bayesian decoder.

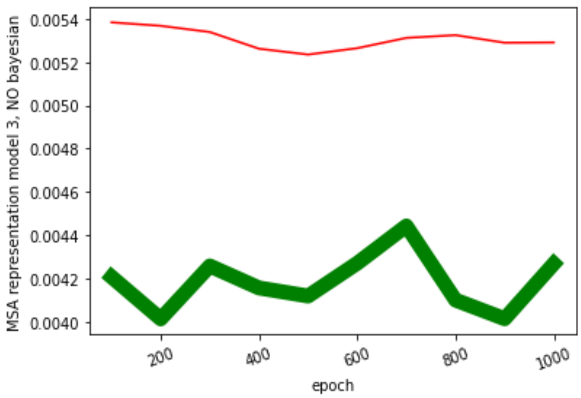
Following below, one can see the training plots of the augmented model. Contrary to the baseline model, it was decided to finetune the model by progressively lowering the learning rate as the model would keep on learning, which is not true for the baseline model for which the the loss would plateau to a minimum hard to decrease even at lower learning rates.



Training plot 3: augmented model 1, lr: 0.001



Training plot 4: augmented model, lr: 0.0001



Training plot 5: augmented model, lr: 0.00001

In the last 2 training plots of the augmented model, it can be seen clearly that the model performs better on the training data compared to the held-out set. This being said, even though the model is picking up signal from the trained set that cannot be generalized to unseen proteins, it is also true that the validation loss seems to go down as the training loss goes down, just by a lower amount. The interpretation of this training plot might be that the model has difficulty separating random pattern in the data from the signal when learning after a certain epoch threshold. The lowest losses for the augmented model are around 0.0042 for the train loss and 0.0053 for the test loss. Therefore, even from just the test loss, it appears clear how the augmented models performs better than the baseline model.

5. Results and Analysis

The goal of this chapter is to analyze the baseline model and the augmented model, as they are defined and trained according to the previous chapters. Furthermore, other experiments are performed on the AlphaFold's MSA representation to complete the analysis and present a unique and coherent interpretation of the results, with also the goal of highlighting further research directions.

Comparison of the models: baseline vs. augmented VAE

As already explained, VAE are bottleneck models trained with the goal of reconstructing the input, while regularizing the hidden dimension. One of the first ways to compare the baseline and the augment model is to analyze the reconstruction quality.

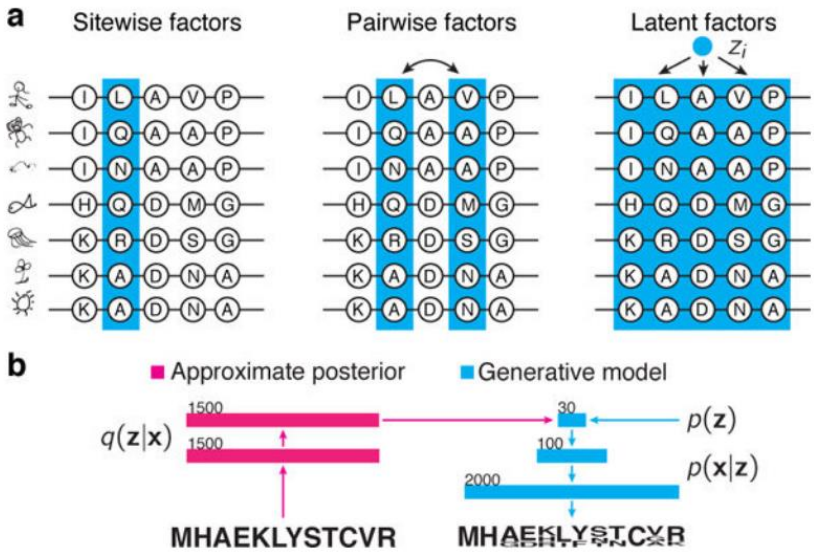


Fig. 1 |. A nonlinear latent-variable model captures higher-order dependencies in proteins and RNAs.

a, Comparison of a nonlinear latent-variable model with site-independent and pairwise models. **b**, The dependency $p(\mathbf{x}|\mathbf{z})$ (blue) of the sequence $\mathbf{x}$ on the latent variable $\mathbf{z}$ is modeled by a neural network, and inference and learning are made tractable by joint training with an approximate inference network $q(\mathbf{z}|\mathbf{x})$ (pink). This combination of model and inference is also known as a variational autoencoder. The size of the latent variables $\mathbf{z}$ and hidden dimensions of the neural network are shown.

Source: DeepSequence paper (Source: Riesselman A, Ingraham J, Marks D. Deep generative models of genetic variation capture the effects of mutations. *Nat Methods*. 2018;15(10):816-822. doi:10.1038/s41592-018-0138-4)

There are multiple ways to quantify the reconstruction quality. A first alternative is to check the prediction at each locus and select the index of the amino acid with the highest value. As the output is always the one-hot encoding of the MSA for both baseline and augmented models, the shape of the output is a matrix of shape (length of the focus sequence, 20) for each sequence predicted. As illustrated by the figure above from the DeepSequence paper, each column of the matrix can be seen as a probability distribution over the possible 20 amino acid. Therefore, given the output matrix, it is possible to check what is the most likely prediction at each position and confront this information with the input to obtain the reconstruction accuracy. With such a reconstruction quality metric, the experiment shows that a random model would guess around 5% of the positions correctly, which is expected as the probability of randomly guessing is one over twenty possible amino acids. On the same tasks, with a sample of 100 sequences, the baseline model achieves 59.65% accuracy rate, while the augmented model achieves 66.78% after the first training and just over 70% accuracy rate after the finetuning with a lower learning rate. Therefore, it is possible to observe a significant difference of more than 10% between the two models.

Alternatively, the models can be compared also from the reconstruction loss on the test set during training. In fact, the augmented test loss is three times smaller than the test loss for the baseline model. Similar patterns emerge if the negative log-likelihood is used instead of the MSE as a reconstruction loss.

Once again, both the baseline and the augmented models have as output the one-hot encoding MSA. Therefore, only the baseline VAE actually implements an identity function, whereas the augmented VAE implements a mapping between AlphaFold’s hidden MSA representation to the one-hot

encoding representation. As such, the AlphaFold's representation has proved to be a better input than the one-hot encoding itself to be compressed and then decompressed to reconstruct the one-hot encoding of the MSA. Such an experiment does not validate per se the use of AlphaFold's MSA representation, but it definitely shows that if one believes the one-hot encoding to be a valid representation of a protein, then AlphaFold encodes at least similar information, and it possibly also represents it in a better way for compression or dimensionality reduction.

Correlation analysis of the MSA representation

To suppose that the AlphaFold's MSA representation can help in mutation analysis means to assume that such a representation can encode information relative to the stability of the protein itself. In other words, the hypothesis that this thesis wants to investigate is if the representation of an unfit protein is different in some dimension from the one of a fit protein. It might seem obvious that, if AF is able to understand the correct folding of a protein, it should be able to tell apart fit and unfit proteins. Unfortunately, one of the downsides of Deep Learning is that such models work mostly as black-boxes and it is not easy to interpret them or know what the behavior will be. As such, one can only plan an experiment and explore the results. In the case of AF, as the model is trained to fold existing proteins, the architecture implicitly assumes that all the sequences that it tries to fold are actually fit proteins, so it is not granted that the MSA representation might contain useful information for mutational analysis. On the other hand, it seems reasonable to hypothesize that AF should struggle more to fold unfit sequences, as they should be fundamentally different from fit sequences it has been trained on. As an example, it could be that AF is on average less certain of its prediction when a protein is unfit compared to when it exists in nature, and that such change in certainty might be encoded somehow in the MSA representation.

Given a data distribution, there are two main cases to separate two labels – in the case of protein stability, the labels are fit and unfit sequences. In the first case, the labels are separable by a linear classifier. This would be the ideal case, such that if the distribution is *n*-dimensional, it is possible to find a *n-1* dimensional linear object to separate, under some conditions, the two labels. If this first condition is not met, then a Neural Network can be used. What a Neural Network does for classification is to take the input space and transform it, layer by layer, to a new space where the labels are easily separable by a linear classifier, which is implemented as the last layer of the Neural Network for classification. Therefore, if the information for mutational analysis and fitness of a protein is embedded in the MSA representation there are two ways to look for it: if a linear classifier is enough, then correlation analysis of the MSA representation matrices should show a different correlation between elements of the two classes. Otherwise, if correlation analysis is not enough, other experiment using a Neural Network – or any other model which performs non-linear transformation to the input distribution – should be designed to find a setting such that the MSA representation performs better than the one-hot encoding, and such that the only way to perform better in the task is by encoding fitness information about different potential mutations. The remaining of this section will discuss the correlation experiment, while the next section will take care of additional experiments using Neural Networks.

Before starting the correlation analysis, it is important to address how correlation is carried out between 2D matrices. Usually, the correlation coefficient is estimated between two 1D vectors. Given two matrices with the same shape, there are two ways to perform correlation: the first method is to correlate, column by column, the 1D vectors of one matrix with the 1D vectors of the other and average all the correlation to have a unique number. Otherwise, the same procedure can be carried out with

the rows instead of the columns. While the absolute value of the correlation coefficient between the MSA representations depends on which one of the two methods one decides to use, all the patterns discussed in the thesis have been found with both methods.

The first correlation analysis carried out was to correlate the MSA representation of a protein with the MSA representation of the same sequence, but with one or two mutations. As shown in the following table, the higher the number of mutations from the original sequence, the lower the correlation. Such a result is good, as it means that at least the MSA representation is not a random point matrix, but instead the correlation diminishes the more mutations occurs to the starting sequence. The next question is harder: is AlphaFold's representation able to distinguish between good and bad mutations?

Number of mutations	**Avg correlation with focus sequence**
One mutation	~ 97 %
Two mutations	~ 90 %

First of all, what is a bad or a good mutation? Using the Supplementary Table 2[12] from the DeepSequence paper, it is possible to find experimental and predicted values for specific mutations to a focus sequence. For the purpose of this thesis, the main protein family taken into account is the BLAT-ECOLX family of proteins. In these settings, a good mutation is one such that the fitness is not reduced, and maybe even improves. On the other hand, a bad mutation is one that disrupt the fitness of the protein.

12 Riesselman A, Ingraham J, Marks D. Deep generative models of genetic variation capture the effects of mutations. *Nat Methods*. 2018;15(10):816-822. doi:10.1038/s41592-018-0138-4

Using the data from the Supplementary Table 2, it is possible to understand what the top 5 best and worst single mutations for the fitness of the original protein are. From these 10 sequences, the MSA representation can be obtained. Unfortunately, all the sequences, irrespectively of the type of mutation, have extremely similar correlation with the MSA representation of the focus sequence, as shown in the following table.

Type of mutation group	**Avg correlation with focus sequence**
Top 5 fittest mutations	~ 98.6 %
Top 5 unfittest mutation	~ 98.5 %

To summarize the findings, the MSA representation is able to characterize different sequences. In general, the more different two sequences are in their amino acid composition, the lower the correlation between their MSA representation will be. This being said, such MSA representation seem not to be able to understand if a mutation is deleterious or not, or at least not with a linear classifier. In the following section, the goal will be to investigate instead if fitness information can be extracted with a series of non-linear transformations.

Experiments using Neural Networks

Finally, two last experiments were performed to find information useful for mutational analysis using non-linear transformations.

The first experiment consisted in comparing the reconstruction accuracy between fit and unfit mutation of the focus sequence. In fact, one could

say that the MSA representation embeds information about the stability of a mutant if a model trained on it is able to reconstruct better fit protein compared to unfit mutations. The motivation is simple: as the VAE works by compressing and decompressing the protein representation, the goal is organizing the latent dimension in order to find a distribution that characterize fit proteins – and not unfit proteins. Therefore, if the model were to perform worse on unfit mutations, it means that they are mapped into the latent space outside of the distribution, which would imply that the MSA representation is able to distinguish between normal and non-normal instances. To give a bit of context, VAEs are sometimes used for anomaly detection. The model is trained by learning to compress only "good" data points. The assumption is that when a "bad" data point is presented, if the model is trained well enough, it should recognize that this latter data point is fundamentally different and it should be mapped outside of the latent distribution, leading to a lower reconstruction accuracy.

Unfortunately, the experiment was not successful because the mutations taken into account, both fit and unfit, were point mutations of the focus sequence. As such, the model classifies them as the original focus sequence obtaining almost 100% reconstruction accuracy in all cases, because only one amino acid is mutated. In the future, it would be interesting to repeat this experiment using, as fit and unfit mutations, proteins that differ from the original focus sequence by multiple mutations. In such settings, where the input is more dissimilar from the majority of the training data, it might be possible that the model could leverage possible fitness information in the MSA.

The second and last experiment, instead, took a completely different approach. As already stated, the output of the VAE is a probability

distribution over the amino acids. From previous experiments, the augmented model has proved to be more accurate than the baseline at predicting the most likely amino acid at each position (59% vs. 70%). So, the question arises naturally: is there any pattern in the distribution of the other amino acids? Or is it just noise? In other words, when predicting the focus sequence, other than the first amino acid at each locus, is it possible that the second and third amino acids predicted are also the most likely to be mutations at that position? These questions motivated the formulation of the last experiments.

The reconstruction accuracy of the focus sequence for both model is exactly 100%. This should not surprise as most sequences in the MSA are probably very similar to the focus sequence, so it is to be expected that the model would overfit on it. What is surprising instead is the correlation between the other predictions at each position, and their relative fitness as mutations.

After sorting the mutations at each locus by fitness, it appears that the second prediction at each locus is 14.4% the most fit mutation for the baseline model and 27.75% for the augmented model. For the third prediction instead, it coincides with the second most fit mutation for 9.12% of cases for the baseline model and 10.27% for the augmented model. For all the further prediction, the prediction for both model is around 5%, which is just random guessing.

The result of this last experiment is exciting in many ways: first of all, it is clear how the MSA representation outperforms the one-hot encoding in prediction of the fitness of point mutation. On the other hand, it is also really interesting how the baseline model not only reconstructs the input, but it is able to predict the first and second fittest mutation at each

position with an accuracy better than random noise, which suggests that the model is somehow able to distill this information on its own.

This last experiment does not necessarily suggest that the MSA representation has additional information not contained in the one-hot encoding, but such a representation is clearly more versatile in the ease of extracting information from it, both for reconstruction accuracy and mutation fitness prediction.

6. Conclusions

This work examined the effectiveness of AlphaFold's MSA representation as a new embedding for proteins, with a special focus of mutation analysis, as it is a problem very closely related to protein folding.

The results of the analysis clearly show that there can be better representations for proteins than a simple one-hot encoding. Most importantly, there is still a lot that can be leveraged from big open-sourced models, and a lot of creative research can be done by finetune existing models to closely related applications. Additionally, it would be very interesting to investigate how to design pretext tasks for self-supervised networks to learn protein embeddings for mutation analysis as the primary goal.

References

1. Jumper J, Evans R, Pritzel A et al. Highly accurate protein structure prediction with AlphaFold. Nature. 2021;596(7873):583-589. doi:10.1038/s41586-021-03819-2

2. Riesselman A, Ingraham J, Marks D. Deep generative models of genetic variation capture the effects of mutations. Nat Methods. 2018;15(10):816-822. doi:10.1038/s41592-018-0138-4

3. Hopf T, Ingraham J, Poelwijk F et al. Mutation effects predicted from sequence co-variation. Nat Biotechnol. 2017;35(2):128-135. doi:10.1038/nbt.3769

4. Jeong C, Kim D. Structure-based Markov random field model for representing evolutionary constraints on functional sites. BMC Bioinformatics. 2016;17(1). doi:10.1186/s12859-016-0948-2

5. Kingma D, Ba J. ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. Published as a conference paper at ICLR 2015; 2017. https://arxiv.org/pdf/1412.6980.pdf. Accessed August 5, 2022.

6. Mikolov T, Chen K, Corrado G, Dean J. Efficient Estimation Of Word Representations In Vector Space. Mountain View, CA; 2013. https://arxiv.org/pdf/1301.3781.pdf. Accessed July 28, 2022.

7. Kingma D, Wellin M. Auto-Encoding Variational Bayes. Amsterdam; 2014. https://arxiv.org/pdf/1312.6114.pdf. Accessed July 28, 2022.

8. Starita L, Fields S. Deep Mutational Scanning: A Highly Parallel Method to Measure the Effects of Mutation on Protein Function. Cold Spring Harb Protoc. 2015;2015(8):pdb.top077503. doi:10.1101/pdb.top077503