Projet 2 - Predicting the effect of a mutation in a protein sequence

Salim Cherkaoui, Maxim Kondracki, Léa Blinière
EPFL, CS - 433

***Abstract*—Proteins are polymers or chain of molecules composed of amino-acids. The type and order of amino-acids in this chain defines the sequence of a protein. Moreover, proteins are molecular machines with a specific role or function. Modification or mutation of specific amino-acids may lead to a disruption or increase of the function.**

I. Introduction

Proteins are large biomolecule that comprises one or more long chains of amino acid sequence. They are vital for the structure, function and regulation of the tissue and organs of the human body. Their various functions depend on the type of amino acid employed in their sequences as well as how they are arranged. A change in the amino acid sequence may affect a protein's structure but not always its function. Identifying mutations that cause human genetic diseases has become a focus of recent research. Knowing the type of deleterious mutation is essential for developing the best molecular diagnosis strategy.

The aim of this research is to understand whether the mutation of an amino acid in the sequence of a protein impacts its function. In this paper, we explore three different methods that learn the relationship between amino acid sequence mutations and protein function. For this purpose, we have at our disposal a database of about 60,000 proteins. Our work gives a comparative analysis of the different classification techniques on data of mutated proteins that are classsified into two different classes : depending on the effect or not of the mutation.
First, we will present the dataset, then we will describe our different strategies to answer the problem, and finally we will compare the performances of these strategies.

II. Data

In this section, we will explore our data.

Sizes of the given sets The data has already been divided into three parts.

- **Training set** : 46392 (80%)
- **Testing set** : 6170 (10%)
- **Validation set** : 5967 (10%)

Features For each dataset there is 7 features :

- **uniprot** : Our data has been extracted from the UniProt Knowledgebase (UniProtKB), it is a comprehensive database resource for protein sequence and its functional information. For each protein, a Uniprot code is assigned.
- **pos** : position in the protein sequence of the mutated amino acid.
- **ori_res** : original amino acid.
- **var_res** : amino acid that replaced the original during the missense mutation.
- **desc** : brief description of the function of the protein
- **length** : size of the sequence
- **no effect** : labels for the model, returns true if the mutation had no effect on the normal function of the protein and false otherwise.

In order to exploit these data, we extracted the sequences corresponding to the uniprot codes from another dataset provided.

Length of the sequences

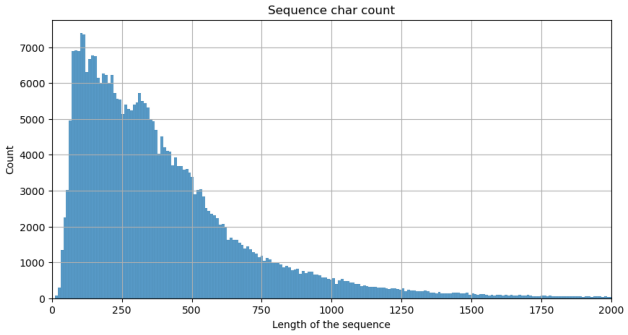

Figure 1. Histogram showing the distribution of the length of the sequence

On average the sequence length is 581 and 75% of our data have a sequence length lower than 741. Due to limited computational power we have limited our study to proteins whose sequence were lower than 600.

Imbalanced data The data provided is very unbalanced, with more than 91% of our protein mutations having an effect on the function of the protein versus 9% of mutations having no impact. The imbalanced data has a significant impact on the performance of our models. To address this problem, several approaches have been developed and will be discussed later.

III. Methods

Three models have been implemented. The first one is a Residual Neural Network (Residual NN) , the second one is a Convolution Neural Network (CNN) and finally a more complex model using a causal approach and a pretrain model to feed a relation network (CRN).

A. Methods 1 and 2

1) Preprocessing:

For the CNN and Residual NN

Feature selection is one of the major tasks for the performance of our models. From the data provided to us we were able to extract for each protein its sequence before and after mutation. The machine cannot process textual data that is why each sequence has been transform into numerical data following this procedure :

- **Integer encoding** : There are 20 different amino acids that can compose our sequences. For each amino acid, an integer has been associated. The sequence of amino acids has been transformed into a sequence of integers.
- **Padding the sequence** : The sequences of our proteins have different sizes. as mentioned above, the study was limited to proteins of length less than 600 for computational reasons. This step consists in removing the data with a longer sequence and adding zeros on each side for the shorter sequences.
- **One Hot Encoding** : It is not a good practice to let a categorical variable be encoded in integer. Indeed, it might even bias our models by assuming a natural order between these variables. One way to overcome this is to use a one hot encoding. This type of encoding consists in transforming each integer into its binary equivalent.

2) Classification techniques:

Convolutional Neural Network : In this approach we process our data as an image with 2 channels. The first channel is the one hot encoded matrix of the non-mutated sequence, the second channel is the one with the mutated sequence. To find the best architecture, we tried different set of convolutional layers. As shown by Ronak Vijay[1], the best way to identify patterns and correlation in a protein sequence is to go as deep as possible in the dimensions of the convolution at each convolutional layer. In order to find even more correlation between each subsequence, we added strides. By this way, the network should be able to extract relevant features which tells information about the influence of the mutation on proteins.

Figure 2. shows the final chosen architecture.

This CNN network consists of 4 layers:

1) **conv1** : A 2D convolutional layer with 90 filters, each kernel has a size of (3,3), and a stride of (2,2). The input to this layer has 2 channels, which could represent the number of color channels (e.g. red and green) in the input images.
2) **conv2** : A 2D convolutional layer with 180 filters, each kernel has a size of (3,3), and a stride of (2,2).

[1]Protein Sequence Classification

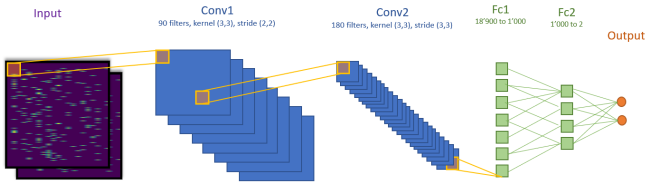

Figure 2. Architecture of the Convolutional Neural Network

3) **fc1** : Fully-connected layer with 1000 units. The input to this layer is a flattened version of the output from the previous convolutional layer, with a size of 18900 (90 filters * 21x21 output size from the previous convolutional layer).
4) **fc2** : Fully-connected layer with 2 units, which is the output layer of the network. The number of units in this layer is 2, the number of classes that the network is meant to classify.

In summary, this CNN takes in an image with 2 channels and processes it through two convolutional layers and two fully-connected layers, finally producing an output of 2 classes.

Residual Neural Network : This models introduces skip connection, which consists in adding to the output of the previous layer to the next one. In short, residual blocks allow memory to flow from initial to last layers of the block. Our input is the same as in the classic convolutional neural network presented above. The model has 3 convolutional layers, a max pooling layer, a dropout layer, and three fully-connected layers. The final layer is a softmax layer that outputs a probability distribution over the classes. The convolutional layers and the max pooling layer are responsible for extracting features from the input image data, while the fully-connected layers use these features for classification. The dropout layer is used to prevent overfitting by randomly zeroing out some of the activations during training.

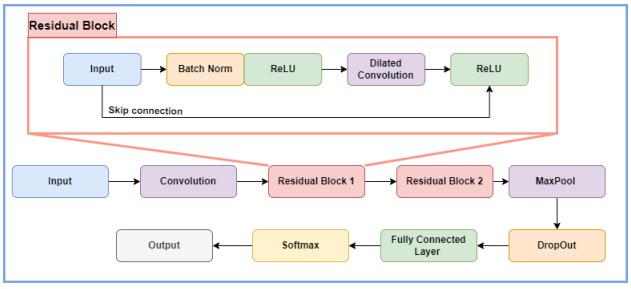

Figure 3. Architecture of the Residual Neural Network

1) **First convolution** : This is a 3D convolutional layer with 16 filters each with a size of 1x1. It is used to extract basic features.

2) **Residual Block 1 and 2** : Residual blocks that consist of a dilated convolutional layer followed by batch normalization and a ReLU activation function. The dilation ratio for the first Residual Block is 2 and for the second 4.
3) **MaxPool** : The max pooling layer reduces the spatial resolution of the feature maps by applying max pooling with a kernel size of 3 and a stride of 3.
4) **DropOut** : The dropout layer randomly zeroes out 50% of the activations during training.
5) **Fully Connected Layer and softmax layer** : The fully-connected layers transform the extracted features into class scores, with the final softmax layer producing a probability distribution over the classes

B. Method 3 : Causal Relation Network (CRN)

We are trying to evaluate the influence of a mutation on a sequence of amino acids. Our data is provided through reel experiments which can allow us to use a causal approach to solve the problem.

1) Causal approach : Causal deep learning is a type of machine learning that focuses on identifying and estimating the causal relationship between variables. In other words, it aims to understand how a particular intervention or treatment affects some outcome of interest, rather than just identifying a correlation between the two. A common practice is to train a model to predict the outcome of experiment with and without a treatment so see if the treatment has an effect.
In our case, the mutation can be consider as treatment, and we will use a pre-train model to extract features from the factual (original sequence) and the counterfactual (sequence after the mutation) environment and use that to feed a relation network.

2) Relation network: In deep learning, a relation network (RN) is a type of neural network architecture that is used to learn the relationships between different objects or entities in a dataset. In its simplest form the RN is a composite function:

$$\text{RN}(O) = f_\phi \left(\sum_{i,j} g_\theta(o_i, o_j) \right),$$

where the input is a set of "objects" $O = o_1, o_2, ..., o_n$, $o_i \in R_m$ is the ith object, and f_ϕ and g_θ are functions with parameters ϕ and θ, respectively. For our purposes, f_ϕ and g_θ are MLPs.
A relation network consists of two main components: a feature extractor and a relation module. The feature extractor is responsible for extracting high-level features from the input data, such as image features or text features. The relation module then takes these features as input and learns to compute the relationships between the objects or entities represented by the features.
The output of a relation network is a scalar value that represents the strength of the relationship between the input objects or entities.
We are using the ProtBert model (discussed in the following) as a feature extractor.

3) ProtBert and transfer learning: Transfer learning is an ML method in which a pretrained model, such as a pretrained BERT model for text classification, is reused as the starting point for a different but related problem.
ProtBERT is a pretrained model on protein sequences using a masked language modeling objective. It's based on the BERT model, which is pretrained on a large corpus of protein sequences in a self-supervised fashion.
BERT (Bidirectional Encoder Representations from Transformers) is a type of transformer-based neural network architecture for natural language processing tasks. One of the key techniques used to train BERT models is masking. The purpose of masking is to force the model to predict the missing tokens based on the context provided by the rest of the input.
In our case, we will extract features from the sequence masked with the [MASK] token on the mutation position and the mutated sequence and try to evaluate the difference.

4) The CRN model: Our model is there a combination of the last three parts, as we reason on the mutation in a causal manner, to extract different features from the ProtBert model to feed a Relation Network to classify correctly our data.

1) **Feature extraction** : We extract the feature of the original masked sequence and mutated sequence (separatly) using the ProtBert model.
2) **Relation network** : We know try to estimate how much the mutation affected the sequence. For that we use a first MLP (g_θ) on each features (one forward pass for each sequence). Than we perform an element-wise sum of the two outputs of this MLP that we use as input for the last MLP (f_ϕ) classifying if the mutation had an effect on the sequence or not.

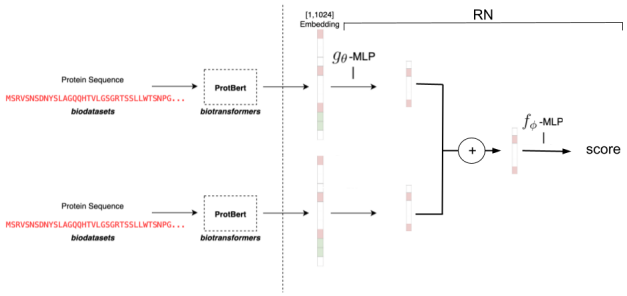

Figure 4. Architecture of the CRN model

IV. Experimental results and analysis

In this study, the following evalutaiton measures are utilized to assess the performance of the previously mentioned classifiers.

A. Performance on sequences of maximum length of 215 and 600

The models were first evaluated on their performance on sequences of maximum length 215 and 600.We take as much sequence of this size with true labels, and we complete our set with ths same amount of sequences with false labels. Thus, 1000 protein data sets with a sequence length less than 215 were randomly selected from the train set (then split 80-20% train-test). Following the same approach, we selected 5585 data for our performance evaluation on sequence lengths below 600. Among this dataset, 4500 were dedicated to training and the rest to the test set. In the training set, 27974 data met our constraint of maximum length 600. The results can be found below.

Size of the sequence is 215					
Model	*Accuracy*	*F1 score*	Precision	Recall	Specificity
Residual NN	0.760	0.750	0.791	0.713	0.808
CNN	0.780	0.758	**0.852**	0.683	**0.879**
CRN	**0.796**	**0.796**	0.809	**0.784**	0.809

Table I
Performance of the 3 models on sequences smaller than 215 amino acids (small dataset)

Size of the sequence is 600					
Model	*Accuracy*	*F1 score*	*Precision*	*Recall*	*Specificity*
Residual NN	0.852	0.830	0.854	0.807	0.888
CNN	**0.882**	0.855	0.901	**0.827**	0.927
CRN	0.875	**0.856**	**0.960**	0.772	**0.970**

Table II
Performance of the 3 models on sequences smaller than 600 amino acids (larger dataset)

B. Performance of models on sequences cut around the position of the mutation

An interesting evaluation of our models was to know how the position of the mutation in the sequence influenced the performance. The sequences were cut around the mutated amino acid. Different lengths of sequences around this amino acid were taken. The objective is to understand to what extent our models are influenced by the proximity of the mutated amino acid, an how much does it understand about the logic of a sequence composition. (see Table III in the appendix).

C. Performance of models on the different type of mutation

To have a better understanding of our models' performances, we compute the accuracy of their predictions for each type of encountered mutations. This allow us to see more how the model reasons. Does it only make correlation between the original and the mutation amino acids and the distribution of the data, or does it reason on the structure of a sequence.

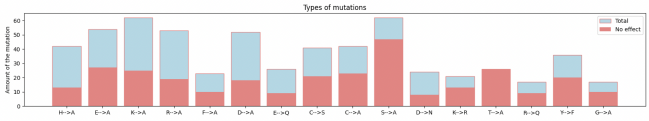

Figure 5. Amount of each type of mutation in the dataset

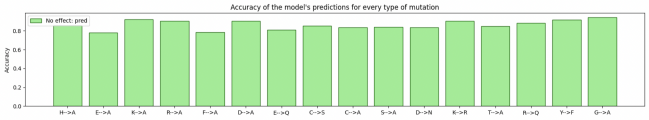

Figure 6. Accuracy of the Causal Relation Network for each mutation

We can see that the accuracy vary depending on which kind of mutation we are dealing with, but very much. The model seems there to be able to deal with any mutation, and is not overfitting to certain types. This is a very good information to have, since, for certain distribution of the data, with can get pretty good accuracies, but without the model learning anything meaningful, just by having a shallow understanding of the distribution (exemple : if A becomes C than 'no effect').

D. Analysis

In A. we can see that CRN outperforms both RNN and CNN on our sequences composed of less than 215 amino acids and on those composed of less than 600 amino acids. Also in B. (Table III) we can see that both RNN and CNN models are not really affected but the size of the cut of the sequence that we take into account for the prediction. Whereas the CRN model improves the more complete the sequence is. Those results and the one got in C. allow to presume that the CRN has a better reasoning on the composition of a sequence and better understanding on how the amino acids should be associated, making the CRN the most robust model. This makes sens since the CRN is using a pre-trained model that captures the logic behind a sequence composition, which is transfered to our model.

V. Further works

We could try to combine the CNN with the CRN. One could encode the input data from the sequence using the pre-processed model from Amazon, and follow the same causal approach. It surely would be interesting to continue this work with more computational power and try to reproduce the results with more data. Also it would be interesting to add a more biological aspect and reasoning to the project, and try to see how much the model develops a biological understanding of the dataset.

VI. References

(1) Timothy Lillicrap (Deep Mind), A simple neural network module for relational reasoning
(2) Timothy Lillicrap (Deep Mind), Measuring abstract reasoning in neural networks
(3) Jiayao Zhang (Cognitive Computation Group, University of Pennsylvania, USA.), ROCK: Causal Inference Principles for Reasoning about Commonsense Causality
(4) Ahmed Elnaggar, ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Learning
(5) How to use Amazon ProtBert
(6) Amazon ProtBert GitHub

VII. Appendix

Model	*Size*	*Accuracy*	*F1 score*	*Precision*	*Recall*	*Specificity*
Residual NN	10	**0.570**	**0.626**	**0.558**	**0.713**	0.424
	20	**0.500**	**0.558**	0.504	**0.624**	0.374
	40	0.535	**0.597**	**0.531**	**0.683**	0.384
	100	0.530	0.453	0.549	0.386	0.583
CNN	10	0.510	0.500	0.516	0.485	**0.535**
	20	**0.555**	0.553	**0.561**	0.545	**0.566**
	40	0.465	0.473	0.471	0.475	0.455
	100	0.460	0.426	0.460	0.396	0.525
CRN	10	0.497	0.492	0.506	0.478	0.510
	20	0.481	0.440	0.487	0.402	0.562
	40	0.564	0.521	0.589	0.467	**0.663**
	100	**0.624**	**0.580**	**0.671**	**0.511**	**0.742**