Using Convolutional Autoencoders for Predicting Change in Protein Stability

Aum Khatlawala *IIIT Hyderabad*
Hyderabad, India
aum.k@research.iiit.ac.in

Sarthak Aggarwal *IIIT Hyderabad*
Hyderabad, India
sarthak.aggarwal@students.iiit.ac.in

May 9, 2023

Abstract

Predicting protein stability change upon single point mutations through Machine Learning Models has proven to be a valuable tool in unveiling the mechanisms of mutation-induced drug failure and helping the development of immunotherapy strategies. To this end, we propose a novel method for predicting the effects of mutations on protein stability using Convolutional Autoencoders. Our method was evaluated on a dataset of protein structures and mutations, and it was able to predict the effects of mutations on protein stability with high accuracy. We believe that our method has the potential to be a valuable tool for understanding the impact of mutations on protein function.

1 Introduction

Proteins are essential biomolecules that perform a wide range of biological functions in living organisms. For a protein to function properly, it must be able to maintain its three-dimensional structure under a variety of circumstances. Protein stability can be significantly impacted by mutations, which are modifications to a protein's DNA sequence. Understanding the effects of mutations on protein stability is a challenging topic with significant consequences in areas like drug discovery, protein engineering, and personalised medicine to comprehend and forecast the effects of mutations on protein stability.

In this study, we propose a novel framework to predict the impact of mutations on protein stability using Convolutional Autoencoders (CAEs) in conjunction with Fully Connected Neural Networks (FCNNs). An artificial neural network called a CAE may be trained to learn how to represent the structural data of

proteins in a low-dimensional latent space. We can utilise encoding abilities of CAEs to our advantage to capture the crucial characteristics of non-mutated proteins by training a dataset of wild-type proteins with known stability values.

The proposed method involves training a CAE to produce latent space representations of both wild-type and mutant proteins. Then, an FCNN framework that has been trained to anticipate the change in stability caused by the mutation will be fed the difference between the latent space vectors of the wild-type and mutant proteins. Once trained, the FCNN can be used to predict how changes in protein stability will affect previously unknown mutant proteins.

The relationship between protein mutations and stability may be better understood through the use of this research, which also has potential applications in the fields of protein engineering, drug development, and personalised medicine. The methodology, results, and discussion of our proposed approach's implications and limitations are covered in more detail in the following sections.

2 Data

Dataset creation The dataset was created by collecting protein structures from the Protein Data Bank (PDB) and mutations from the UniProtKB database. The dataset was then cleaned and filtered to remove incomplete or inaccurate data.

Dataset Description	
Feature	**Description**
Protein structure	3-D coordinates of the atoms in the protein
Mutation	Location and type of the mutation
Number of protein structures	2160
Number of mutations	2160
Protein types	Enzymes, structural proteins, signaling proteins
Mutation types	Point mutations, insertions, deletions

Dataset limitations The dataset is relatively small, which could limit the accuracy of machine learning models trained on the dataset. The dataset is

not representative of all proteins, as it only contains protein structures from the PDB. This means that the results of studies using the dataset may not be generalizable to all proteins.
Dataset value Despite these limitations, the dataset is a valuable resource for researchers studying the effects of mutations on protein stability. The dataset can be used to train machine learning models to predict the effects of mutations on protein stability. The dataset can also be used to identify mutations that are likely to have a significant impact on protein stability.

3 Baseline Model

We used two papers as baseline models for rigorous testing of our model.

- ProS-GNN: The baseline model.
- ThermoNet: The baseline model for the baseline model.

In the ProS-GNN paper, Wang et al. introduce a new method for predicting the effects of mutations on protein stability. Their method is based on graph neural networks, which are a type of machine learning model that can be used to learn the relationships between the atoms in a protein. They trained their method on a dataset of protein structures and mutations, and they showed that it could predict the effects of mutations on protein stability with high accuracy.
Wang et al.'s method could be used to identify mutations that are likely to have a significant impact on protein stability, and it could be used to develop new therapies for diseases that are caused by protein misfolding.

3.1 Dataset Used

The use of the below mentioned datasets allowed the authors to train and test their model on a variety of protein structures and mutational effects. This allowed them to develop a model that is more generalizable and robust than previous models. There are four dataset used in the baseline which are:

1. S2648: This dataset contains 2648 protein structures with their corresponding mutational effects on protein stability. The mutational effects are measured by the change in Gibbs free energy ($\Delta\Delta$G).
2. S3412: This dataset is similar to S2648, but it contains 3412 protein structures.
3. S^{sym}: This dataset contains 684 protein structures with their corresponding mutational effects on protein stability. The mutational effects are measured by the change in the relative Gibbs free energy ($\Delta\Delta$G).
4. Myoglobin: This dataset contains 134 protein structures with their corresponding mutational effects on protein stability. The mutational effects are measured by the change in the fractional occupancy of the native state (ϕ native).

3.2 Code Availability

The computational framework presented in the paper, pretrained neural network weights, and data used for model training are available in a GitHub repository from URL: `https://github.com/shuyu-wang/ProS-GNN`

3.3 Methods

The basic flowchart of the baseline model is as follows:
The steps involved in the architecture of the baseline model are as follows:

1. Input: The wild-type protein structure and the mutated protein structure.
2. Extracting the atomic coordinates of the protein structure near the mutation site: The protein structure near the mutation site is represented as a set of 3D coordinates for each atom in the protein. These coordinates are used to construct a graph where the nodes represent atoms and the edges represent the bonds between atoms.
3. Constructing a graph where the nodes represent atoms and the edges represent the bonds between atoms: The graph is constructed using the atomic coordinates of the protein structure. The nodes in the graph represent the atoms in the protein, and the edges in the graph represent the bonds between the atoms.
4. Calculating the features of each atom, such as its atomic number, hybridization and charge: Each atom in the graph is assigned a set of features (one hot encoded vector). These features are used to describe the properties of the atom, such as its atomic number, hybridization and charge.
5. Passing the atom features to the graph neural network: The atom features are passed to the graph neural network. The graph neural network is a machine learning model that is able to learn the relationships between the atoms in the graph.
6. The graph neural network will learn the relationships between the atoms and how they affect protein stability: The graph neural network will learn the relationships between the atoms in the graph by training for around 300 epochs. This information will be used to predict the change in protein stability that is caused by the mutation.
7. The graph neural network will output a prediction for the change in protein stability: The graph neural network will output a prediction for the change in protein stability that is caused by the mutation. This prediction can be used to identify mutations that are likely to destabilize the protein.
8. Output: Predicted change in protein stability.

3.4 Results for the Baseline Model

Model	RMSE (test)	PCC (test)
ProS-GNN (Claimed)	1.23	0.61
ProS-GNN (Running locally)	1.2	0.56
ThermoNet (Baseline for ProS-GNN)	1.56	0.47

1. RMSE - Ideal value is 0. The previous best was 1.56 and the baseline achieved 1.15.
2. PCC - Ideal value is 1. The previous best was 0.47 and the baseline achieved 0.6.

4 Methodology

4.1 Problem Formulation

The task at hand is to predict the change in Gibbs Free Energy ($\Delta\Delta$G) between mutant protein and wild-type protein.

4.2 Approach

The input data received is in the form of PDB files, which contain elemental and structural information about the area around the mutation site in both, the wild-type protein and the mutant protein. From this, we extract the one hot encoded feature vectors for each atom in the PDB file and this has information about the following features about the atom:

1. Atom Symbol: C, N, O, S, F, P, Cl, Br, B, H.
2. Degree (Number of Adjacent Atoms): 0 to 6.
3. Adjacent Hydrogen Atoms: 0 to 4.
4. Implicit Valence: 0 to 6.
5. Aromatic Bond: If aromatic, then the 30^{th} entry is set to 1, else it is set to 0.

After this, the feature vectors are trimmed in such a way that only the first 50 atoms are considered (if less than 50 atoms, then the vector is padded

accordingly). The reason behind doing so is due to the large number of entries which have around 30 to 50 atoms in them and the very few entries which have more than 50 atoms. Thus, we finally obtain a uniform-length 2-D feature vector of size (50, 30).
We send this as input to a trained CAE to obtain a latent space vector of size (8, 30) and this essentially reduces the dimensionality of the data from 1,500 dimensions in the input space to 240 dimensions in the latent space.
The latent space dimensions for the mutant protein and the wild-type protein are then subtracted into a vector and this process is repeated for all the entries in the dataset and a new dataset called $latent\ space\ diff$.
The $latent\ space\ diff$ dataset is then split into training and testing subsets and passed to a trained FCNN to obtain the predicted change in stability.
Thus, to summarise, the approach is as follows:

- Input: Wild Protein Structure and Mutated Protein Structure.
- Getting the atom features: One hot encoding is used for storing the feature vector of the atoms into the dataset on which the model will be trained upon. These features are used to describe the properties of the atom, such as its atomic number, hybridization and charge.
- Passing the atom features to the CAE: The atom features thus obtained are passed to the convolutional autoencoder. The convolutional autoencoder architecture is a type of neural network that is used to learn latent representations of data by applying convolutional layers while encoding the data. The convolutional autoencoder architecture consists of two parts: an encoder and a decoder. The encoder takes the input data and compresses it into a latent representation. The decoder then takes the latent representation and reconstructs the input data. Thus, the model performs unsupervised learning.
- Output: Predicted change in the stability of the input protein upon mutation.

4.3 Machine Learning Model Architecture

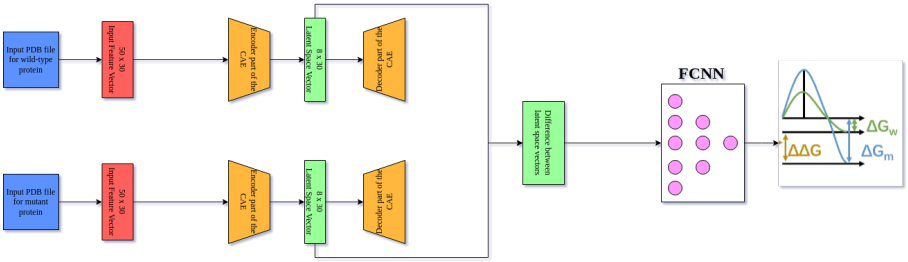

Figure 1: Model Architecture

The input is a 2D array with shape (50, 30).
The **encoder** part of the CAE is composed of a series of convolutional layers, which are followed by a max pooling layer and a fully connected layer. The convolutional layers extract features from the input data, and the max pooling layer reduces the size of the feature maps. The fully connected layer projects the feature maps to a lower-dimensional space.
The **decoder** is composed of a series of convolutional layers, which are followed by a max pooling layer and a fully connected layer. The convolutional layers reconstruct the feature maps from the lower-dimensional space, and the max pooling layer increases the size of the feature maps. The fully connected layer projects the feature maps to the original input space.
The **loss function** used is the MSE loss function. The MSE loss function is a loss function that is used to measure the mean squared error between the predicted output and the ground truth output.
The **optimizer** used in the code is the Adam optimizer. The Adam optimizer is a gradient descent algorithm that is used to update the model parameters to minimize the loss function. The Adam optimizer is a stochastic gradient descent algorithm, which means that it updates the model parameters using a small subset of the data at each iteration. The Adam optimizer is used to update the parameters of the CNN to minimize the MSE loss function. The Adam optimizer is a popular optimizer for deep learning models because it is efficient and effective.
The **learning rate** used in the code is 0.001. This is a small learning rate, which will cause the model to learn slowly. However, a small learning rate will help the model avoid overfitting the training data.
The number of **epochs** used in the code is 20. A small number of epochs will help the model to avoid overfitting the training data.

4.4 Training

Parameter	CAE	FCNN
Optimizer	Adam	Adam
Training Datapoints	4320	1944
Epochs	20	10
Learning rate	0.001	0.001
Loss	0.026 (after 20 epochs)	0.824 (after 10 epochs)
Testing Datapoints	4320	216
Input Dimensions	50 x 30 (1500)	8 x 30 (240)
Output Dimensions	8 x 30 (240)	1

4.4.1 CAE

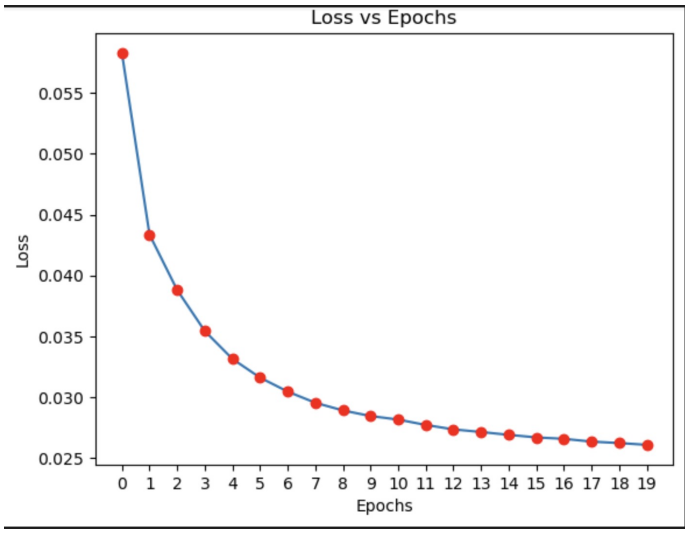

Figure 2: Loss vs Epoch curve for the Convolutional Auto-Encoder

4.4.2 FCNN

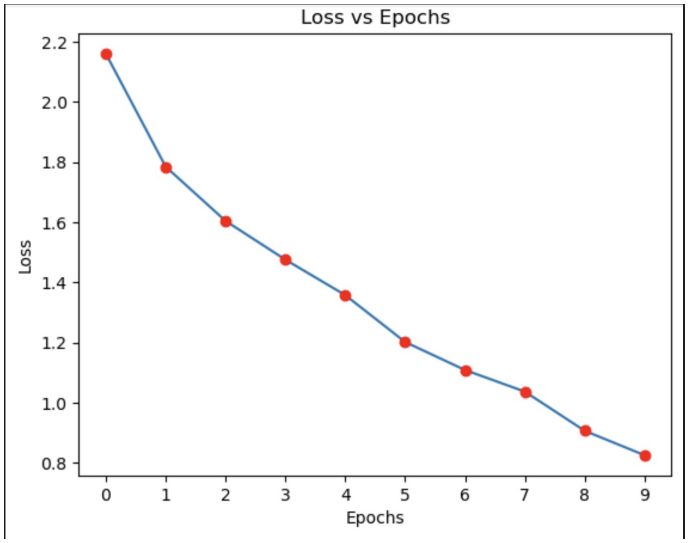

Figure 3: Loss vs Epoch curve for the Fully Connected Neural Network

4.5 Model Performance Assessment

The Pearson correlation coefficient and the RMSE can be used to assess the performance of the model.
The Pearson correlation coefficient is a measure of the linear correlation between two variables. The Pearson correlation coefficient can be used to assess the relationship between the predicted stability values and the ground truth stability values. The Pearson correlation coefficient can be used to assess the strength of the relationship between the predicted stability values and the ground truth stability values.
The root mean squared error (RMSE) measures the average error between the predicted stability values and the ground truth stability values. The RMSE can be used to assess the accuracy of the model.

5 Results

We obtained Pearson Correlation coefficient value of 0.5353 and root mean squared error value of 1.1877 on unseen/test dataset.

Model	RMSE (test)	PCC (test)
CAE+FCNN (Running Locally)	1.18	0.53
ProS-GNN (Running locally)	1.2	0.56
ThermoNet (Baseline for ProS-GNN)	1.56	0.47

We can clearly observe that our model architecture has outperformed the ThermoNet model and is close to the ProS-GNN model.
But, the model is able to outperform both the models in terms of the time taken to run. ProS-GNN is significantly slower than our model in terms of time taken to train and run the model. ProS-GNN took around 30 seconds for 300 epochs (150 minutes) to reach the desired results while our model's CAE took around 30 minutes to train for 20 epochs and our model's FCNN took 1 minute to train for 10 epochs to receive the desired output.
To conclude, our model is able to achieve results similar to the state-of-the-art models in the field while taking significantly lesser time to train.

6 References

[1] Li B, Yang YT, Capra JA, Gerstein MB (2020). Predicting changes in protein thermodynamic stability upon point mutation with deep 3D convolutional neural networks. PLoS Comput Biol.
[2] Wang S, Tang H, Shan P, Zuo L (2021). ProS-GNN: Predicting effects of mutations on protein stability using graph neural networks.
[3] Chris Kuo (2019). Convolutional Autoencoders for Image Noise Reduction (Medium Article).
[4] Kulshreshtha S, Chaudhary V, Goswami GK, Mathur N (2016). Computational approaches for predicting mutant protein stability. Journal of Computer-Aided Molecular Design, 30, 401-412.
[5] Rodrigues CH, Pires DE Ascher DB (2018). DynaMut: predicting the impact of mutations on protein conformation, flexibility and stability. Nucleic Acids Research, 46, W350 - W355.
[6] Capriotti E, Fariselli P, Casadio R (2005). I-Mutant2.0: predicting stability changes upon mutation from the protein sequence or structure. Nucleic Acids Research, 33, W306 - W310.