Final Project Report

CS 184A, Fall 2022

Project Title: Novozymes Enzyme Stability Prediction

Student Names:

Eesha Tur Razia Babar, 45369117, ebabar@uci.edu

Calvin Yeung, 88763753, chyeung2@uci.edu

Po-Chu Hsu, 80441969, pochuh@uci.edu

1. Introduction and Problem Statement

Enzymes have many applications, including in agriculture, human health, nutrition, and cleaning. However, many enzymes are marginally stable and hence cannot perform well under harsh conditions. Currently, empirical methods have been used to confirm an enzyme's thermal stability, which can be costly in terms of time and money. Hence, it is necessary to develop an efficient model to predict protein thermal stability. Thus, in this project, we aim to develop models that predict/rank an enzyme's relative thermal stability given its amino acid sequence and the pH of the environment. As the goal is to rank the relative thermal stability of enzymes, the output of the model does not matter in absolute terms; rather, we require that the predicted rank order matches the relative order of the target values. To solve this problem, we use an LSTM model as a baseline as well as a Transformer encoder model.

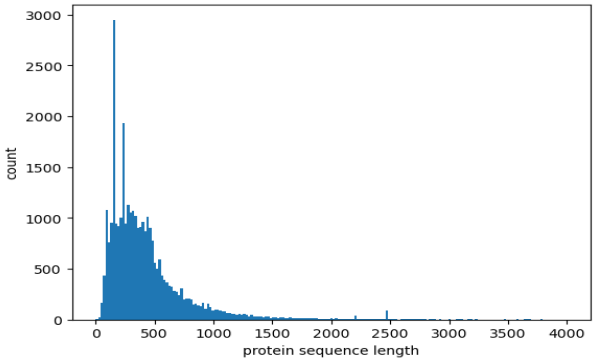
2. Related Work

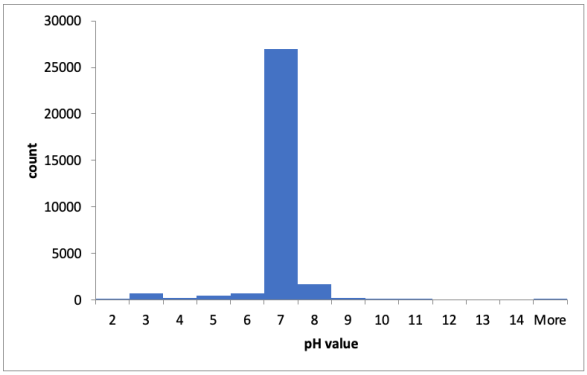
Much work has been done in the past related to protein thermal stability based on physics principles at the quantum level. For instance, methods such as Rosetta ΔΔG[1] and FoldX[2] are force-field-based methods that make predictions based on the change in folding energy. There have also been machine learning approaches to this problem, such as ELASPIC[3], an ensemble-based method, DeepDDG[4], a neural network-based method, BoostDDG[5], an extreme gradient boosting-based method, and ProS-GNN[6], a graph neural network-based method. Our project aims to extend upon existing machine learning-based methods by using machine learning models specialized for handling sequences, i.e., LSTM[7] and Transformer encoders, to encode protein sequences, and make protein thermal stability predictions.

3. Data Sets

We used the Novozymes enzyme stability prediction dataset from Kaggle for our project. The training data consists of 31,398 samples. The features are unique identifiers, protein sequence and pH. A protein sequence is a sequence of characters, where each character corresponds to an amino acid. There are 20 characters in total. pH is a measure of acidity of a solution, ranging from 0 to 14. The pH feature refers to the pH of the solution in which the stability of the protein was measured. The target label Tm is a continuous number, which is the thermostability (melting temperature) of the enzyme. A higher value of Tm means higher thermal stability. The correctness of a prediction is the relative order of thermal stability

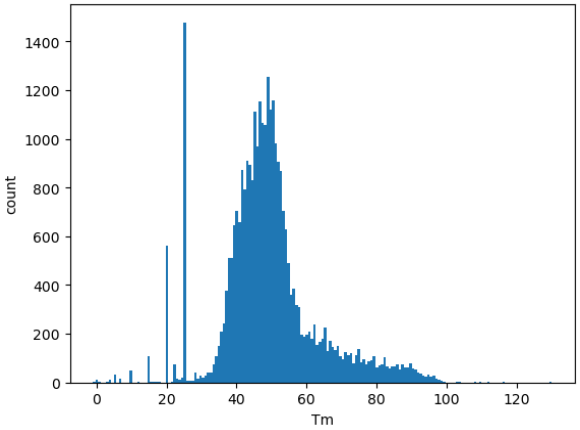
instead of the exact magnitude. The training data consists of natural enzymes (wild type) as well as engineered sequences that are single- or multiple-point mutations of the natural enzymes. Thus, all enzymes in the training dataset can be grouped according to their corresponding natural enzyme. The test set contains 2,414 samples and consists of unique identifiers, protein sequences, and pH. The enzymes in the test dataset are all single-point mutations of an enzyme. Thus, the sequence lengths in the test set are either 220 (from amino acid deletion) or 221. Moreover, since the test data is provided solely by Novozymes A/S, the measurement conditions are standardized. As such, all the pH values are 8 in the test data. Since the training dataset consists of many different types of enzymes originating from different wild types, there is a huge variety in sequence lengths. The figure to the left below shows the distribution of sequence lengths in the training dataset.

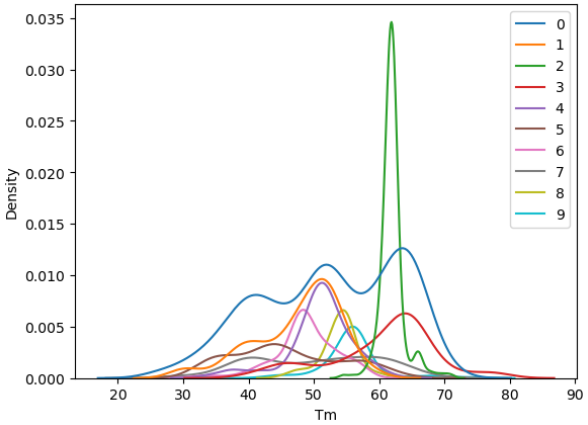




The training data also comes from various data sources, so the measurement conditions are not standardized, as indicated by differences in the pH of the aqueous solution in which the measurement was performed. The figure to the right above shows the distribution of pH values in the training dataset.

The left figure above shows the distribution of Tm values in the training dataset. In the figure to the right, we plot the Tm KDE density for groups of enzymes in the training dataset grouped by their wild type. We plot the density for the 10 largest wild type enzyme groups. The Tm distributions of some groups are rather concentrated, which means it would be difficult to rank the thermal stability within the group. Instead, it is easier to rank thermal stability across groups since the distributions across groups are quite varied, making it easier to distinguish. Both the train and test data set also provide links for data sources.





4. Description of Technical Approach

4.1. Data Preprocessing

The original dataset provided has some errors such as swapping Tm and pH values, so we used a script provided in Kaggle to fix this error. We also dropped the rows with NaN values. We also drop sequences with more than a set amount of tokens, which we control using a hyperparameter called MAX_LEN. This is because longer sequences require larger models, which increases computational cost. Moreover, the sequences in the test dataset are only 220/221 in length, so long sequences might not contribute much to the performance of the model. We preprocess the data into two versions: one where the sequences are converted into a one-hot encoding for the LSTM, and another where the sequences of tokens are converted to sequences of numbers for the Transformer-based models.

We cluster the provided training data into groups based on wild type. There are 1002 unique lengths for protein sequences. The minimum group size is 5 and the mutation threshold (insertion, deletion) is 1. The algorithm groups different protein sequences into different subgroups and plus the sequences with 1 mutation away from the basic sequence. Then each sequence in the subgroup is divided into three parts considering the fact that since the mutation threshold is 1, all the sequences in one subgroup must have 1/3rd of the similar consecutive substring. Hence three lists of different lists are obtained, one with starting part of each sequence, another with the center and the last with the ending part. The most common substring is obtained for each list but counting the occurrences. Hence the most common substring and the count of that substring is obtained. The occurrences of starting substring, middle substring, and ending substring is compared. Then the proteins are reduced to the substrings they match closely. The reduced list of substrings is used to find the wildtype through the mode function to find the most common substring and then is identified as wild type of that group. However, only a tiny portion of the training data consists of enzymes that can be grouped. We label the training data by their group IDs: non-negative integers for wild-type groups, and -1 for all enzymes that cannot be grouped.

4.2. LSTM Model

We used an LSTM as a baseline model. We choose LSTM as a baseline as it is a classic neural network model designed to handle sequential data. The LSTM model works sequentially, taking the input sequence one token at a time and updating its hidden state. The next hidden state is determined by the previous hidden state as well as the current input. An LSTM model also makes use of various techniques such as keep tracking of the cell state and using the forget mechanism to increase the range of its memory.

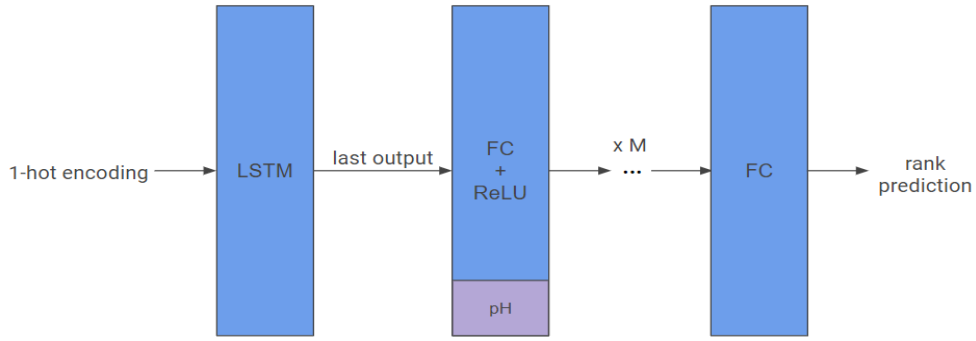
We use the LSTM by taking in a batch of sequences that are dynamically padded with zeroes such that they are all of the same lengths. We take the output of the final hidden state and concatenate the pH to it. We then pass it through M fully connected layers to get the final output. The figure below gives a high-level description of the model architecture.

4.3. Transformer-based Model

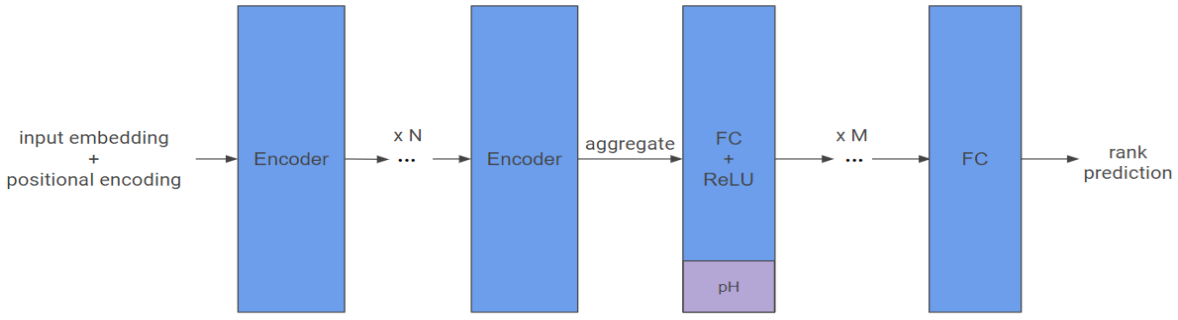
We chose a Transformer-based approach as it can capture longer-range dependencies compared to LSTMs, and as shown in the dataset description, many of the sequences are long. Thus, we believe a Transformer-based model is appropriate for this situation.

A Transformer encoder module consists of a self-attention layer with a skip connection and normalization, which then passes its output to a feedforward layer also with a skip connection and normalization. It is the self-attention layer that enables the learning of long-range dependencies. The encoder also adds positional information to the input embeddings via positional encoding, which ensures the sequential nature of the learned representation. The encoder module is shown in the figure to the right. Thus, our encoder consists of a positional encoder with N Transformer encoder modules stacked on top of each other. When we pass in the input padded batch, we also compute a mask to perform mask self-attention so that the encoder ignores the padding. The padding is only used so the sequences of different lengths can be aggregated into a batch.

The Transformer encoder outputs a sequence of embeddings for each token in the input sequence. We aggregate the sequence by taking the mean, giving us a single representation for the entire sequence. Then, like in the LSTM model, we take the output of the Transformer encoder and concatenate the pH, then pass it through M fully connected layers to get the final output. The figure below gives a high-level description of the model architecture.



The Transformer encoder outputs a sequence of embeddings for each token in the input sequence. We aggregate the sequence by taking the mean, giving us a single representation for the entire sequence. Then, like in the LSTM model, we take the output of the Transformer encoder and concatenate the pH, then pass it through M fully connected layers to get the final output. The figure below gives a high-level description of the model architecture.



4.4. Loss Functions

The loss function we use has two components: (1) an MSE loss component, which minimizes the expected squared distance between model prediction of Tm and the target Tm; and (2) an intra-group L1 loss, which minimizes the expected absolute difference between the predicted difference in Tm between two protein sequences in the same group, and the actual difference in Tm.

The MSE loss component is used to learn the general Tm values for a given protein sequence. The intra-group L1 loss component is used to force the model to learn to rank protein sequences within a group, a much more difficult task that we ultimately want to learn as it is the task given in the test set. The loss function we use is a sum of the two components. We do not use the intra-group L1 loss by itself because most of the training data are not in a group, so it does not make sense to use intra-group L1 on it. Therefore, we would be wasting a lot of data if we only used the L1 loss. The hope is that using the MSE loss on the ungrouped data will lead to better internal model representations, which would help with the final predictions.

5. Software

4.1. Code that we have written

- Data Exploration: finds the number of unique length of protein sequences and their counts
- Protein grouping: groups the different protein sequences based on length with threshold 1
- Preprocessing: converts data to one-hot encoding/numeric encoding using NumPy.
- LSTM model: written in PyTorch; specifies the model architecture.
- Transformer-encoder model: written in PyTorch; specifies the model architecture.
- Model training: training loop written in PyTorch.
- Model validation: computes Spearman coefficient for a data subset given a model using PyTorch and scipy.
- Dataset/DataLoader: load dataset and pads sequences in batches with zeros dynamically using a custom pad_collate function, written in PyTorch.
- Train/validation split: splits dataset based on group IDs.

4.2. Code by others

- Script used to fix errors in dataset[1]: fixes errors in original Kaggle dataset.
- Positional encoder[2]: adds positional information to input embedding in the Transformer encoder.
- Wild type group clustering[3]: computes wild type group for all enzymes in the dataset and labels each datapoint with the corresponding group ID. SOURCE

[1] https://www.kaggle.com/competitions/novozymes-enzyme-stability-prediction/discussion/356251

[2] https://pytorch.org/tutorials/beginner/transformer_tutorial.html

[3] https://www.kaggle.com/code/roberthatch/novo-train-data-contains-wildtype-groups/notebook

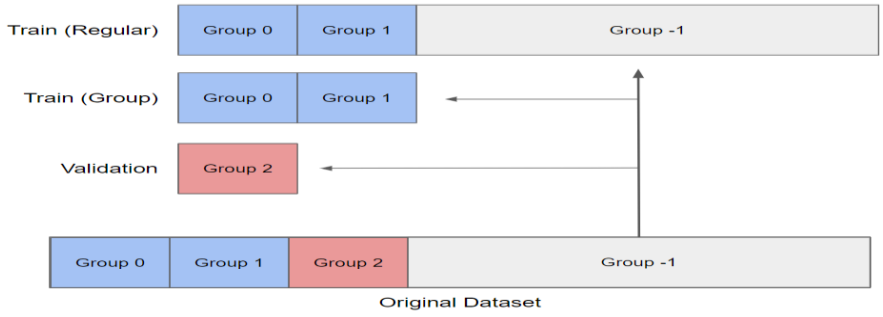
6. Experiments and Evaluation

6.1. Train/Validation Datasets

We split the data into training and validation datasets based on their group IDs. A portion of the groups are placed in the validation dataset. The rest of the groups and all data with a group ID of -1 are placed in the training dataset. All enzymes in the training dataset are not part of any group in the validation dataset. We do this because the no enzyme in the training dataset is part of the wild-type group in the test dataset.

We have two types of training datasets: (1) regular training dataset, where all training data with a group ID of -1 and groups not in the validation dataset are mixed together; and (2) group training dataset, which contains groups not in the validation dataset. We make this distinction as the two types of training datasets are used for the two components of our loss function, which we will discuss in a later section.

The figure below illustrates our train/validation dataset split.



6.2. Evaluation Metric

We evaluate the models using Spearman's correlation coefficient, which measures how similar the two rankings are. Thus, the magnitude does not matter. Only the ordering does.

The validation dataset consists of a set of enzyme groups. We use the validation dataset to evaluate the model by computing the Spearman correlation coefficient between the model's ranking and the target ranking for each group within the validation dataset. We then average the correlation coefficients obtained for each group to obtain what we call the Group Spearman score. We use this method to validate the model instead of computing the overall Spearman score over the entire validation dataset at once because the test set consists of only a single enzyme group. Thus, our approach evaluates the model's performance on a single group, for multiple groups, then takes the average. This better imitates the model's performance on the test set compared to the naive method.

To evaluate the model performance on the test dataset, we upload our model's generated output to Kaggle, which then gives us the corresponding Spearman correlation coefficient.

6.3. Experiment and Results

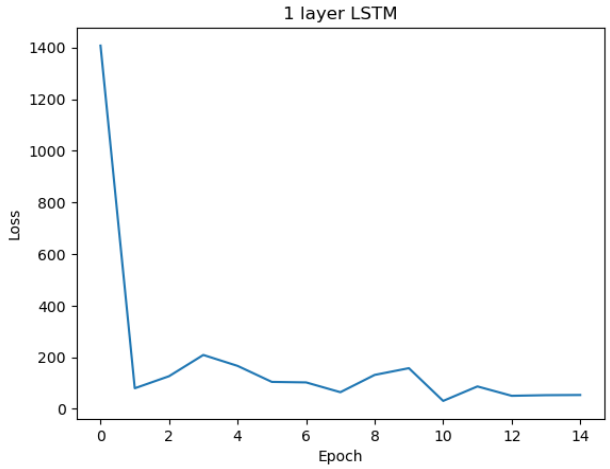
In our experiments, we train and evaluate both the LSTM model, which we treat as a baseline, and the Transformer encoder model. We use the Adam optimizer to train the models, with a batch size of 16

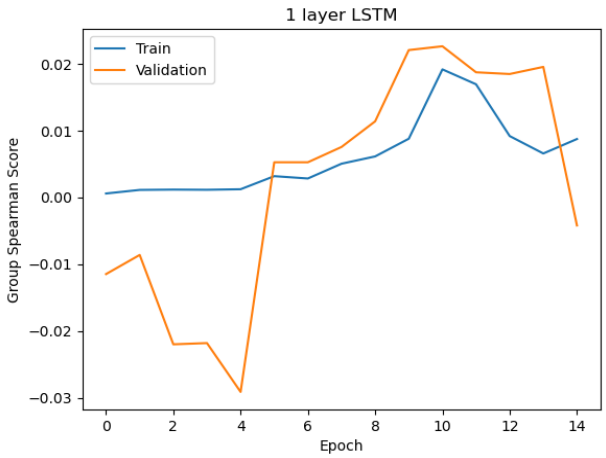
and a learning rate of 0.001. We use a max length of 1024. We train both models for a total of 15 epochs for the 1 layer case and 8 for the 2 layer case. At the end of each epoch, we save the model if it achieves a higher validation score than the last best scorer.

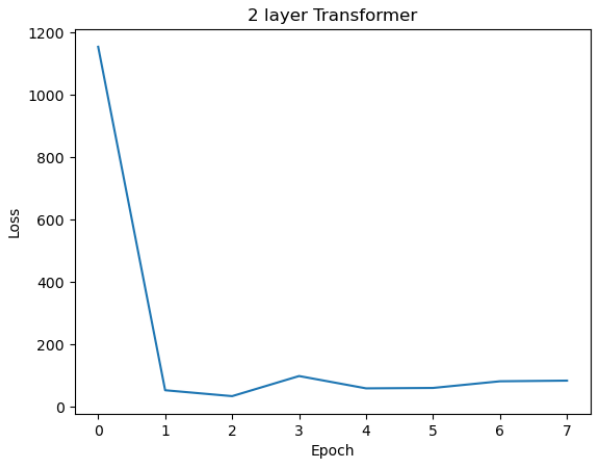
We evaluate two variants of the LSTM model, one with one layer, and the other with two layers, both with a hidden size of 128. We use two fully connected layers with ReLU activation to get the output. We plot the results for the LSTM models below.

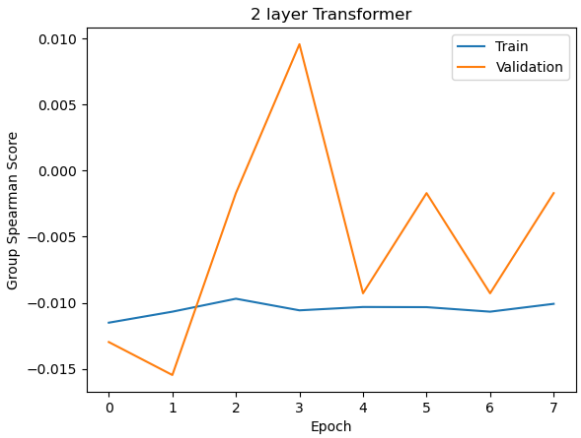
The top left figure shows the training loss versus the number of epochs for the 1-layer LSTM. As expected, it generally decreases over time, with some variation due to stochasticity. The top right figure shows the group Spearman score for both train and validation datasets. In this case, higher is better. Surprisingly, the validation performance surpassed the training performance in epoch 5 and after, only dropping below in epoch 15. The validation performance is maximized in epoch 10.

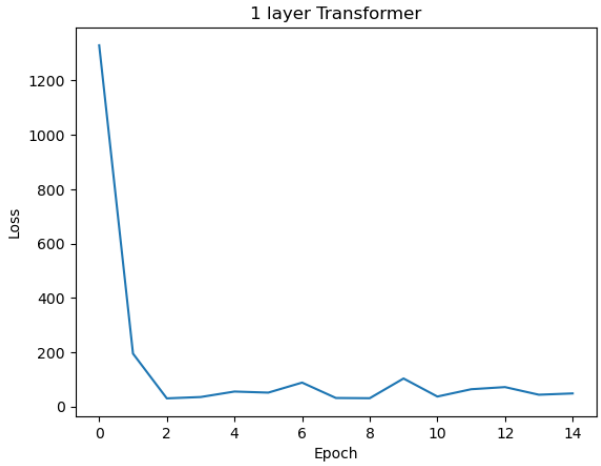
As the results above show, the 2-layer LSTM performs worse than the 1-layer LSTM in terms of validation score. For the Transformer encoder model, we evaluate one with one encoder module. We did not do two layers due to insufficient computational resources. We use 16 attention heads, an embedding dimension of 256, a hidden dimension of 256, and dropout rate of 0.5. We plot the results for the Transformer encoder models below.

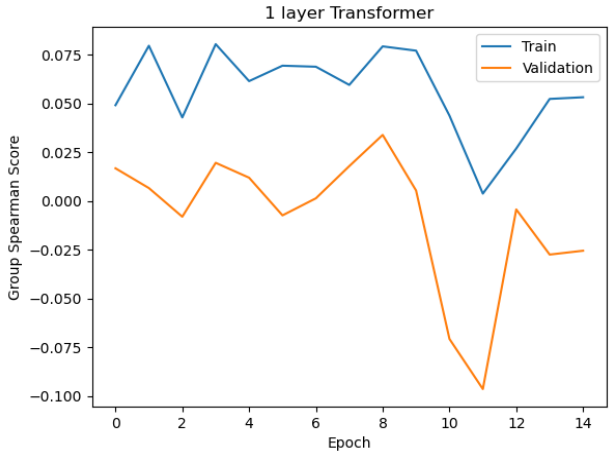












Similarly, as above, the training loss for the Transformer encoder model generally decreases with the number of epochs. The train score is consistently above the validation score, which is expected. The validation score is maximized in epoch 8. In terms of maximum validation score, the Transformer encoder model performs better than the LSTM model.

In the table below, we record the different models' performance on the test set as determined by Kaggle.

Model	Spearman Correlation Coefficient
LSTM (1-layer)	-0.038
Transformer Encoder (1-layer)	-0.045
LSTM (2-layer)	-0.038

As above, the results we get are poor. This is because the train and test distributions are different, and our model did not successfully learn useful statistics needed to learn the connection between protein

sequences and their thermal stability. More advanced techniques and data will need to be used to achieve this goal.

7. Discussion and Conclusion

The problem of predicting the thermostability of protein sequence is challenging and interesting at the same time. In the beginning, the problem looked like a straightforward supervised learning problem that could be easily solved with LSTM or Transformers, however, that is not the case.

First, the training and test distributions are very different, as the training data has many wild type groups and other ungroupable enzymes while the test data only has one wild type group. The wild types present in the training data are different from the one in the test set. Hence, the use of expert knowledge and relevant biological features is the most important section of this project. These features can be obtained readily from the provided PDB file generated by AlphaFold regarding B-factors, 3D coordinates, etc. However, we lack computational resources and biological knowledge to make use of it.

The results, although not ideal, were to some extent expected as we did not have many features and our models were not sufficiently sophisticated due to time and computational limitations. The project can be improved by adding features related to 3D structures predicted by AlphaFold. Pretrained models for solving similar problems like Thermonet[8] can also be used through transfer learning. Specialized Python libraries like Rosetta[9] can also be used for rank calculation. These features will likely help to increase the performance of the model considerably.

8. A separate page on *Individual Contributions*

Calvin: wrote part of the preprocessing code for one-hot and numeric encodings; wrote the train/validation split; wrote the training loop and loss function; wrote the LSTM and Transformer encoder models; worked on the report.

Eesha tur razia : worked on research, data exploration, visualization, preprocessing, testing, and training basic LSTM and report writing

Po-Chu Hsu: worked on the data preprocessing, feature engineering, report writing

Reference

[1]T. Kortemme and D. Baker, “A simple physical model for binding energy hot spots in protein–protein complexes,” *Proceedings of the National Academy of Sciences*, vol. 99, no. 22, pp. 14116–14121, Oct. 2002, doi: 10.1073/pnas.202485799.

[2]R. Guerois, J. E. Nielsen, and L. Serrano, “Predicting Changes in the Stability of Proteins and Protein Complexes: A Study of More Than 1000 Mutations,” *Journal of Molecular Biology*, vol. 320, no. 2, pp. 369–387, Jul. 2002, doi: 10.1016/s0022-2836(02)00442-4.

[3]D. K. Witvliet, A. Strokach, A. F. Giraldo-Forero, J. Teyra, R. Colak, and P. M. Kim, “ELASPIC web-server: proteome-wide structure-based prediction of mutation effects on protein stability

and binding affinity," *Bioinformatics*, vol. 32, no. 10, pp. 1589–1591, Jan. 2016, doi: 10.1093/bioinformatics/btw031.

[4]H. Cao, J. Wang, L. He, Y. Qi, and J. Z. Zhang, "DeepDDG: Predicting the Stability Change of Protein Point Mutations Using Neural Networks," *Journal of Chemical Information and Modeling*, vol. 59, no. 4, pp. 1508–1514, Feb. 2019, doi: 10.1021/acs.jcim.8b00697.

[5]X. Lv, J. Chen, Y. Lu, Z. Chen, N. Xiao, and Y. Yang, "Accurately Predicting Mutation-Caused Stability Changes from Protein Sequences Using Extreme Gradient Boosting," *Journal of Chemical Information and Modeling*, vol. 60, no. 4, pp. 2388–2395, Mar. 2020, doi: 10.1021/acs.jcim.0c00064.

[6]S. Wang, H. Tang, P. Shan, and L. Zuo, "ProS-GNN: Predicting effects of mutations on protein stability using graph neural networks," Cold Spring Harbor Laboratory, Oct. 2021. Accessed: Dec. 06, 2022. [Online]. Available: http://dx.doi.org/10.1101/2021.10.25.465658

[7]S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[8] Gersteinlab, Gersteinlab/Thermonet: ThermoNet is a computational framework for prediction of the impact of single-point mutations on protein thermodynamic stability.., GitHub. Available at: https://github.com/gersteinlab/ThermoNet (Accessed: December 9, 2022).

[9] The rosetta software, RosettaCommons. Available at: ://www.rosettacommons.org/software (Accessed: December 9, 2022).

[10] Novozymes enzyme stability prediction Kaggle. Available at: https://www.kaggle.com/competitions/novozymes-enzyme-stability-prediction (Accessed: December 9, 2022).