Protein-Protein Interactions

Adrian Sager La Ganga, Maximo Cravero, Kushagra Shah
Supervisors: Lucien Krapp, Prof. Dal Peraro
CS-433, EPFL, Switzerland

***Abstract*—This project aims to contribute to the ongoing research in understanding and predicting protein interactions. The quantitative changes in the binding affinity between protein complexes are crucial in characterizing these interactions. We begin exploring the protein interaction data, followed by feature engineering to clean the data and generate meaningful features for the machine learning models. Finally, we implement some standard machine learning models such as Linear Regression, Multi-Layer Perceptron and XGBoost, as well as a novel HydraNet model.**

I. Introduction

Understanding protein interactions is an essential part in medical research since it provides deeper insights into the inner-workings of biological systems and can help combat widespread diseases. Mutations are particularly relevant, since they are a fundamental mechanism for development, positive and negative, in nature, and are linked with different human diseases [1], such as cancer [2].

The effects of missense and multiple mutations are usually quantified through the changes in binding affinity ($\Delta\Delta G$) between the mutant and wild type interacting protein pairs, also called complexes.

Using the latest SKEMPI v2.0 dataset [3], $\Delta\Delta G$ predictors can be trained on a dataset containing labeled examples. Previous methods have used a small number of features for $\Delta\Delta G$ prediction using this dataset [4], [5]. In our approach, more descriptive feature matrices of the molecular system are used, which include the Lennard-Jones potential, the electrostatic potential, and a distance matrix of the relevant atoms at the interface between the interacting proteins.

II. Feature Engineering

The SKEMPI v2.0 dataset [3] provides the atomic structures and experimentally measured binding affinity values for various interacting protein complexes. The structures contain spatial coordinates and types of the atoms, which can be used to extract relevant features for our prediction models. Before computing the features, PDBFixer from the OpenMM tool [6] is used to prepare the structures by adding missing residues, atoms, and replacing non-standard residues. Next, we use OpenMM to create and parameterize the topology of bonded atoms using the Amber 14 force field.

Potential energies and distance matrices were chosen as features since they provide relevant information about the position and interaction between particles. In order to compute accurate potential energies, we run an energy minimization of the structures. The closest interacting atoms between the sub-units are found and, since it is impractical to store all interactions, sub-matrices with a restricted size of 256x256, centered at the smallest pairwise distance, are extracted. Using this information, the distance matrices can be calculated along with the Lennard-Jones potential (U_{LJ}) and the electrostatic potential (U_{elec}) matrices. These three matrices, combined, are used as features for the prediction models (Figure 1). The calculations are done for both the mutant and wild type protein pairs. Figure 1 provides a visualization thereof.

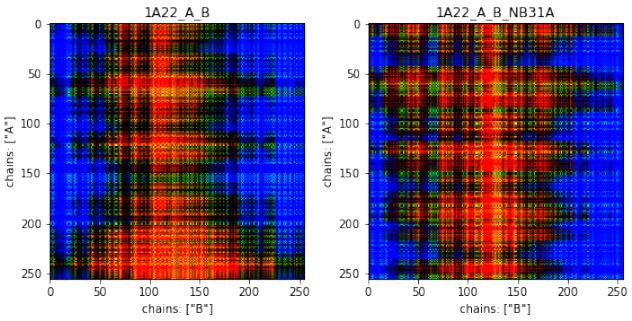

Figure 1. 3-Channel feature matrices: Clipped 'RGB' values corresponding to pairwise Lennard-Jones potentials (R), Electrostatic potentials (G), and distances (B)

Data Processing

Upon some data exploration, the very wide range of the Lennard-Jones potential (-1 to 1e14) compared to the other matrices was discovered. These high interaction potentials are due to atomic clashes created by some of the mutations.

The data was augmented by mirroring the matrices about the vertical and horizontal axes to quadruple the data. Finally, a random train-test split of 80-20 % (a varying proportion was squeezed out for the validation set) was applied to the dataset. The data is split into the same training and testing sets throughout to ensure a fair basis for comparison.

The matrices were standardized into a more uniform distribution. Further, in order have usable data for some models, features were added by calculating the mean and standard deviation of the matrices, as well as storing the

temperature at which the SKEMPI experimental data was retrieved. To handle high-magnitude values for the Lennard-Jones potential of the mutant complexes, the points were clipped between relevant lower and higher thresholds. As with the matrices, these different features where standardized as well.

III. Models and Methods

Baseline: Linear Regression

We consider linear regression as a baseline performance on our dataset in order to gauge the efficacy of the training and preprocessing methods we investigate in this paper. The main metrics used throughout will be RMSE and the Pearson R score, allowing us to compare our methods with other recent papers [4] [5].

Multi Layer Perceptron

The first advanced method considered is the multilayer perceptron (MLP). By selecting the appropriate architecture, one can leverage the representational capability of a multi-layered network containing non-linearities.

The initial consideration concerning the hyperparameter tuning process was the specific parameters and ranges that were used for tuning. Table I outlines the grid of hyperparameters that was used to select a model.

ReLU and Adam with weight decay were used as the default activation and optimizer due to both their high performances and efficiencies. Additionally, K-fold validation with 5 folds was performed in order to ensure a higher degree of robustness. The parameters are initialized randomly according to a uniform distribution which depended on the number of neurons per hidden layer.

Due to long train times, the batch size was kept at 256, which demonstrated similar robustness to lower sizes, but with much lower run times.

The grid search was adapted according to previous runs, so that the initial run had sparse points over the hyperparameter space, and each successive run looked more locally at the points with the best validation score. This, in addition to an overfitting discussion unrolled in Section IV, lead to the final model parameters seen in Table I.

Table I
Hyperparameter Ranges

Parameters	Options	Final
Hidden Layers	{2, ..., 16}	8
Neurons per layer	{2, ..., 256}	128
Activation	ReLU	
Dropout	0% – 30%	25%
Batch-norm	Yes/No	No
Initial weights	Uniform$\left([-N_{\text{neurons}}^{-1/2}, N_{\text{neurons}}^{-1/2}]\right)$	
Optimizer	AdamW	
Learning Rate	$[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]$	10^{-3}
L2 Regularization	$[0, 10^{-3}, 10^{-2}, 10^{-1}]$	None
Batch Size	[16, 32, 64, 128, 256, 512]	256

XGBoost

XGBoost [7] is implemented as an alternative approach to the MLP. It is an efficient gradient tree boosting approach whose underlying mechanism consists of sequentially generating decision trees, each of which acts to rectify the previous tree's errors. It uses the idea of combining multiple weak learners to generate one strong learner, i.e. it is an ensemble tree learner. Contrary to the MLP, it requires a large number of features to make the necessary splits at each node and ensure a good performance. Therefore we need to perform feature expansions to make full use of the algorithm.

Another reason to consider this model is that it tends to outperform deep NNs in the context of smaller datasets. This requires proper tuning of its hyperparameters, which is very challenging due to the high dimensional search space. In order to address this efficiently, we make use of the Bayesian Optimization [8] algorithm, which is a black box optimizer. This requires a model function, that is, a mapping from the hyperparameter space to the evaluation metric of choice of the form $f : \Theta \mapsto x, \quad x \in \mathbb{R}$. Bayesian optimization works by setting a prior distribution for the evaluation metric given by the input model parameters and updating it iteratively. The selected metric is the average RMSE over 10 validation folds, providing a robust set of hyperparameters.

Hydra Net

One issue which both previous models fail to address is that of extracting more detailed information from the feature matrices. Basic statistical parameters provide a decent representation of the data, but by using convolutional layers we can attempt to extract more meaningful implicitly contained details. HydraNet is a double headed convolutional network which takes as input the wild type and mutated protein feature matrices, applies convolutional filters to extract key information, and follows up with fully connected layers to predict changes in binding affinity.

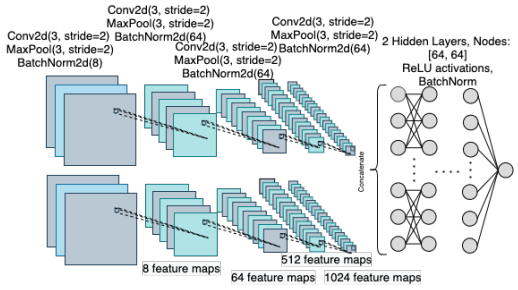

Figure 2. Hydranet Architecture: Separate CNNs and an MLP architecture for the concatenated CNN outputs

IV. Results and Discussion

MLP overfitting

As a baseline for MLP, a configuration was chosen to be 8 hidden layers and 32 nodes without regularization by localizing the grid search, as stated before. Then, another grid search fixing these hyperparameters was performed in order to find the regularization method with the best validation score, which resulted to be 25% dropout, no batch-normalization and no L2 regularization, as well as a learning rate of 10^{-3}.

Since excessively large number of nodes per layer where also demonstrating high R scores and low RMSE losses in validation and test, we performed 10 fold cross-validation with different number of neurons per layer to demonstrate that, although they perform better on average, there is a higher variance in the results, as well as low training losses. The model in these cases is heavily overfitting. Each fold training, in this research scenario, was performed with early stopping, so that if after 300 consecutive epochs the model was not achieving better validation RMSE, the best results were chosen. This discussion is briefed in Figure 3. Additionally, Figure 7 compares the Pearson R scores.

To avoid overfitting, higher probability dropout, batch-normalization and L2 regularization could be used. However, this would overly complicate the MLP model and result in higher training and inference times.

As from the plot, the model with 8 hidden layers and 128 neurons per layer finds a balance between robustness and validation loss. Studying the number of layers in the same manner as for the nodes, all of them showed similar RMSE losses in validation (see Figure 4); however, we kept 8 layers since higher number of layers had higher train loss.

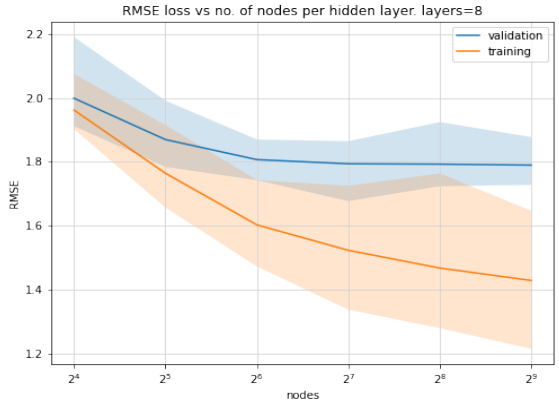

Figure 3. MLP validation and training RMSE losses, using 25% dropout, batch size 512 and 10^{-3} learning rate. The number of layers was fixed to 8. The translucent filled area represents the range of values and the opaque line is the mean.

As a final remark, MLP is not the most suited for this dataset since, for all hyperparameters, R is not above 0.55 and RMSE is not bellow 1.7 in the test set after cross-validation.

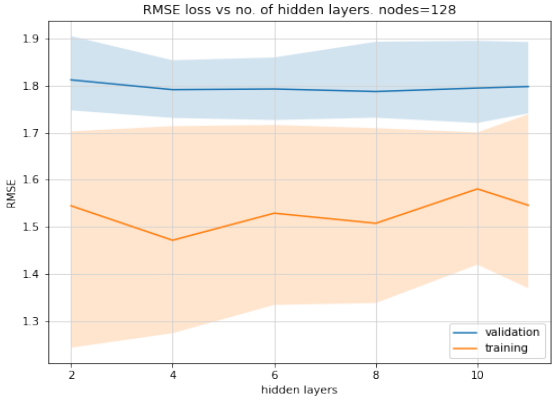

Figure 4. MLP validation and training RMSE losses, using 25% dropout, batch size 512 and 10^{-3} learning rate. The number of nodes per layer to 128. The translucent filled area represents the range of values and the opaque line is the mean.

XGBoost

Given that the dataset is not particularly large, XGBoost is an interesting alternative to explore in order to obtain a higher performance. It has a smaller tendency of overfitting and is far more computationally efficient to train.

XGBoost contains a wider variety of hyperparameters, which make it more challenging and potentially more relevant to our problem. The most critical ones include the L1 and L2 regularizers α and λ respectively, learning rate η, minimum loss reduction required to justify an additional split γ, the maximum depth of each tree, subsampling ratio, and the maximum delta step. The default parameters are used as a baseline to quantify improvements due to data preprocessing and hyperparameter tuning. Table II summarizes the results.

Table II
XGBoost improvements via feature expansions and hyperparameter tuning

	Test R score	Test RMSE
Baseline	0.50	1.78
Feature Expansion	0.53	1.75
Bayesian Optimization	0.55	1.72

Contrary to the MLP model, which does not require feature expansions as they are implicitly learned, XGBoost benefits from additional data transformations. Concatenating the logarithm of the original dataset (applied to strictly positive columns) and adding the cross-channel correlations increased Pearson's R score by 6%.

Figure 5 visually depicts the correlation between the predicted and actual binding affinities, and the resulting model parameters are summarized in Table III.

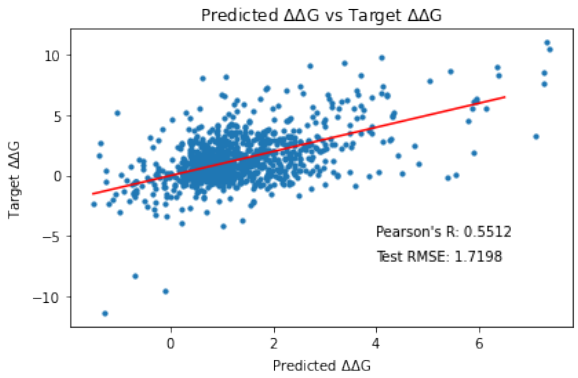

Figure 5. Scatter plot for predicted and target relative binding affinities for tuned XGBoost model

Table III
FINAL MODEL PARAMETERS FOR XGBOOST

α	λ	η	γ	Max Depth	Subsample	Max Delta Step
4.48	0.25	0.04	11	10	0.94	11

Hydra-Net

The first iteration of HydraNet consisted solely of convolutional and fully connected layers with interspersed ReLU activations and max pooling layers. This was unsuccessfully trained as the training RMSE grew exponentially, using the Adam optimizer. To address this, the input matrices were standardized by channel for each mutation and wild type separately, and batch normalizations were implemented to standardize the intermediate outputs and provide training stability. This led to a severely overfitting model, generating a train RMSE of 0.9, and a corresponding validation RMSE of 9.0. The main challenge was thus to reduce the complexity of the model in order to reduce the degree of overfitting. Due to the time consuming training process, it was decided to only consider one validation split. This does not provide the most reliable result, but aids in obtaining a working architecture.

To address the overfitting, the model was regularized via means of dropout in the fully connected layers and randomized augmentations (specifically rotations to preserve the spatial information). Artificially increasing the size of the dataset and reducing over-reliance on specific weights via dropout led to a more robust predictor, with a training RMSE of 1.0, and a decrease in the validation RMSE to 1.9.

Noting the improvements in overfitting through these techniques, we then iteratively tweaked the number of convolutional and fully connected layers, followed by testing variations of the regularization terms to incrementally improve the model. By testing different configurations and slight variants thereof, the following training losses in Figure 6 was obtained.

It is still overfitting clearly, but the simpler models that were tested led to far higher validation errors. The resulting validation error was 1.71 with a Pearson R score of 0.46. The final model architecture is outlined in Figure 2, which further used a batch size of 16 and a learning rate of $1e-4$, as well as a weight decay of $1e-4$.

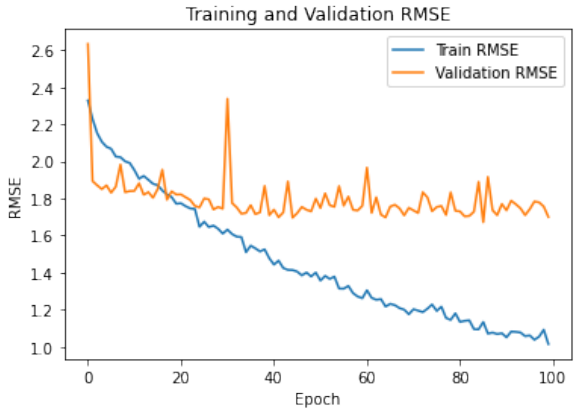

Figure 6. Training and testing error HydraNet

Overview

An overview of our final test results and those of other papers are outlined in the following table.

Model	Test R Score	Test RMSE
MutaBind2	0.76	1.34
iSEE	0.43	1.18
Lin Reg	0.19	2.03
MLP	0.53	1.73
XGboost	0.55	1.72
HydraNet	0.45	1.74

Note that the test sets being used differ from the other papers and thus only provide a rough overview of the relative performance of each approach.

V. CONCLUSION

Overall, the two simpler methods (MLP and XGBoost) performed similarly to the HydraNet model. However, this does not necessarily highlight that the methods themselves are better, but rather emphasizes the importance of feature processing. In such a complex problem where the relation between mutations and changes in binding affinities is not at all explicit, it really emphasizes the importance of being able to extract the most crucial pieces of information from the feature matrices.

Whilst the convolutional approach may not have worked as well as hoped, it seems that the key in achieving higher accuracy lies in gathering more features, and extracting more information from the features we already have. On this note, future projects investigating this topic should seek to make this the focus, as the models' representational capabilities are limited by this factor.

References

[1] S. Stefl, H. Nishi, M. Petukh, A. R. Panchenko, and E. Alexov, "Molecular mechanisms of disease-causing missense mutations," *J Mol Biol*, vol. 425, no. 21, pp. 3919–3936, Nov 2013.

[2] F. Zhao, L. Zheng, A. Goncearenco, A. R. Panchenko, and M. Li, "Computational Approaches to Prioritize Cancer Driver Missense Mutations," *Int J Mol Sci*, vol. 19, no. 7, Jul 2018.

[3] J. Jankauskaitė, B. Jiménez-García, J. Dapkūnas, J. Fernández-Recio, and I. H. Moal, "Skempi 2.0: an updated benchmark of changes in protein–protein binding energy, kinetics and thermodynamics upon mutation," *Bioinformatics*, vol. 35, no. 3, pp. 462–469, 2019.

[4] N. Zhang, Y. Chen, H. Lu, F. Zhao, R. V. Alvarez, A. Goncearenco, A. R. Panchenko, and M. Li, "Mutabind2: Predicting the impacts of single and multiple mutations on protein-protein interactions," *iScience*, vol. 23, no. 3, Mar 2020. [Online]. Available: https://doi.org/10.1016/j.isci.2020.100939

[5] C. Geng, A. Vangone, G. E. Folkers, L. C. Xue, and A. M. J. J. Bonvin, "isee: Interface structure, evolution, and energy-based machine learning predictor of binding affinity changes upon mutations," *Proteins: Structure, Function, and Bioinformatics*, vol. 87, no. 2, pp. 110–119, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/prot.25630

[6] P. Eastman, J. Swails, J. Chodera, R. McGibbon, Y. Zhao, K. Beauchamp, L. Wang, A. Simmonett, M. Harrigan, C. Stern, R. Wiewiora, B. Brooks, and V. Pande, "Openmm 7: Rapid development of high performance algorithms for molecular dynamics," *PLoS Comput Biol*, vol. 13, no. 7, 2017. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1005659

[7] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785

[8] F. Nogueira, "Bayesian Optimization: Open source constrained global optimization tool for Python," 2014–. [Online]. Available: https://github.com/fmfn/BayesianOptimization

Appendix
Figures

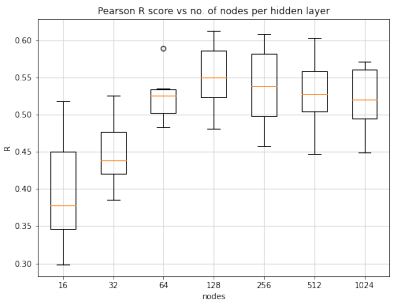

Figure 7. Boxplot of MLP 10-fold cross-validation R scores for validation set, using 25% dropout. The number of layers was fixed to 8.

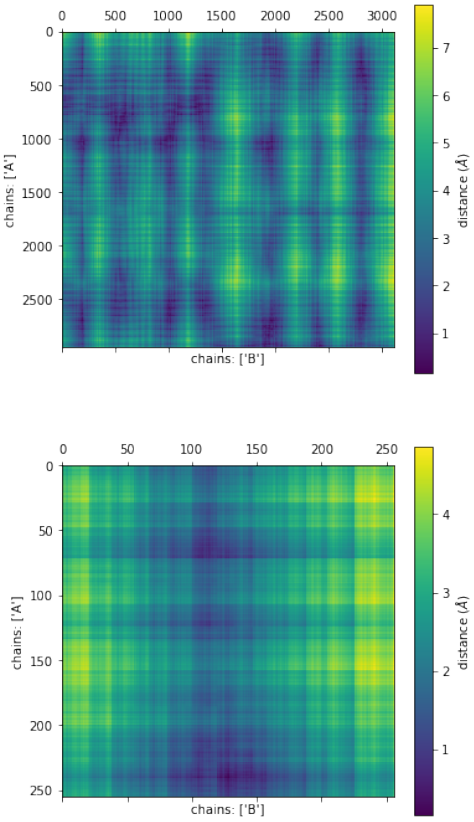

Figure 8. Distance matrix of the 1A22 complex. Bottom plot corresponds to the extracted 256x256 sub-matrix with lowest pair-wise distance.