DeepDDG-reconstruction

Anowarul Kabir
Department of Computer Science
George Mason University
akabir4@gmu.edu

I. Source paper

The source paper [1], titled as *DeepDDG: Predicting the Stability Change of Protein Point Mutations Using Neural Networks*, can be found at https://pubs.acs.org/doi/10.1021/acs.jcim.8b00697. In addition, the authors made the train-test set and feature computation procedure publicly available in the supporting information section at the same link.

II. Source code

The code is available in https://github.com/akabiraka/DeepDDG_reconstruction.git with the instructions of setting up the project, computing features, running the model and evaluation process. I run the code partially in GMU Argo cluster [2].

III. Methodology

A. *Problem Formulation*

Assuming, there are N pairs of wild-type and mutant-type proteins and their experimental $\Delta\Delta G$ values are available, represented as (x_i^w, x_i^m, y_i) for $i = 1, 2, ...N$ where x_i^w, x_i^m and y_i denotes i-th wild-type protein, mutant protein and their corresponding $\Delta\Delta G$ value. Assuming the thermodynamic stability of a mutant protein is likely to change with any degree of mutations compared to its wild-type, the data generation distribution of wild and mutant proteins should be distinctly different. Therefore, the task of estimating $\Delta\Delta G$ values can be defined as a regression problem. To do that, features as abstract embeddings for each residue or some neighbor residues of the mutation position of the protein need to be computed which can be defined as:

$$em(x_i) = em(r_1^{(i)}), em(r_2^{(i)}), ..., em(r_{n_i}^{(i)}) \tag{1}$$

where x_i denotes a wild or mutant protein and r_j indicates j-th residue of the protein. Given the model parameters θ_{model}, the model, g, predicts $\Delta\Delta G$ value for a pair of wild-mutant protein embeddings, $em(x_i^w)$ and $em(x_i^m)$, which can be defined as follows:

$$\Delta\Delta\tilde{G}_i = g(y_i | em(x_i^w), em(x_i^m); \theta_{model}) \tag{2}$$

B. *Data cleaning*

The source dataset is divided into train and test set where the train and test set contains 5444 mutations of 211 proteins and 276 mutations of 37 proteins, respectively, and there are no common proteins between them. However, the dataset contains different $\Delta\Delta G$ values for the same mutation of the same protein. For instance, hiv-1-capsid protein (PDBID (Protein Data Bank id):1A43) contains $\Delta\Delta G$ of -2.3 and -2.8 for the same mutation C-218-S (where 218th residue is substituted from Cysteine to Serine in mutant type) at the same environment condition, such as pH and temperature. To solve this issue, I take the average over the same mutation of a protein following Potapov *et. al* [3]. Yet, same PDBID is referred to as different proteins, for instance, PDBID:1AG2 is reported as PrP and cellular_prion protein. After cleaning these, the final train and test set contains 4344 mutations of 209 proteins and 253 mutations of 37 proteins. Several other protein specific problems are faced throughout the implementation, which are reported in the README document.

C. *Data distribution analysis*

Figure 1 shows train and test set distribution from different perspectives. The train set consists of mutations from all general kinds of 20 residues. In the ideal case, the number of mutations for each amino acid should be similar. However, the training set shows a great compromise among that. Following that, the test set consists of examples from all amino acid types, except aspertic acid (ASP), which is summarized in Figure 1(a).

The source paper also provides the same data distribution analysis with respect to $\Delta\Delta G$ values which is our target variable as in Figure 1(b). Additionally, the authors mentioned that the dataset is biased towards destabilizing mutation where the value of $\Delta\Delta G$ is negative. However, the test set is more balanced compared than train set but no value is out of range [-5, 3] whereas the train set mostly ranges from [-10, 10]. Additionally, the number of examples for each value range is not well distributed. Therefore, the train and test set fail to completely resemble the problem.

Figure 1(c) shows mutation site (the residue number where the mutation occurred) versus the number of mutations. This shows that the mutation site is confined from 0 to 300th position. Compared to this distribution there are some mutations occurred approximately at 850th position which can be thought of as outliers. Since proteins can be of different types and of different lengths and not confined within 1000 residues,

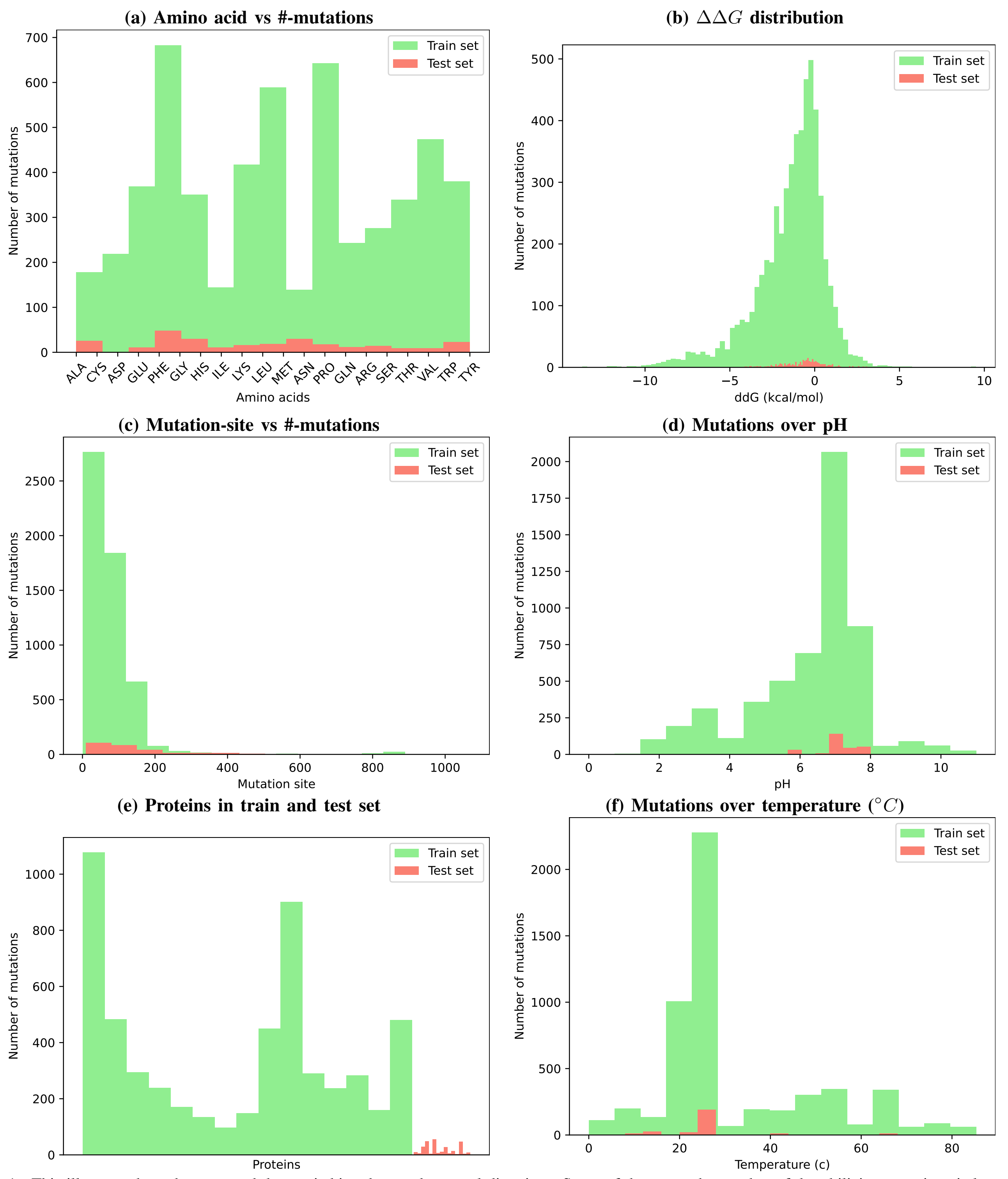

Fig. 1. This illustrates how the proposed dataset is biased towards several directions. Some of them are: the number of destabilizing mutations is larger than stabilizing mutation (b, c), the $\Delta\Delta G$ over mutation is not well distributed for learning task (b), the mutations of particular proteins and residues dominate the dataset (a, c), the environmental condition may induce error in the learning task (d, f).

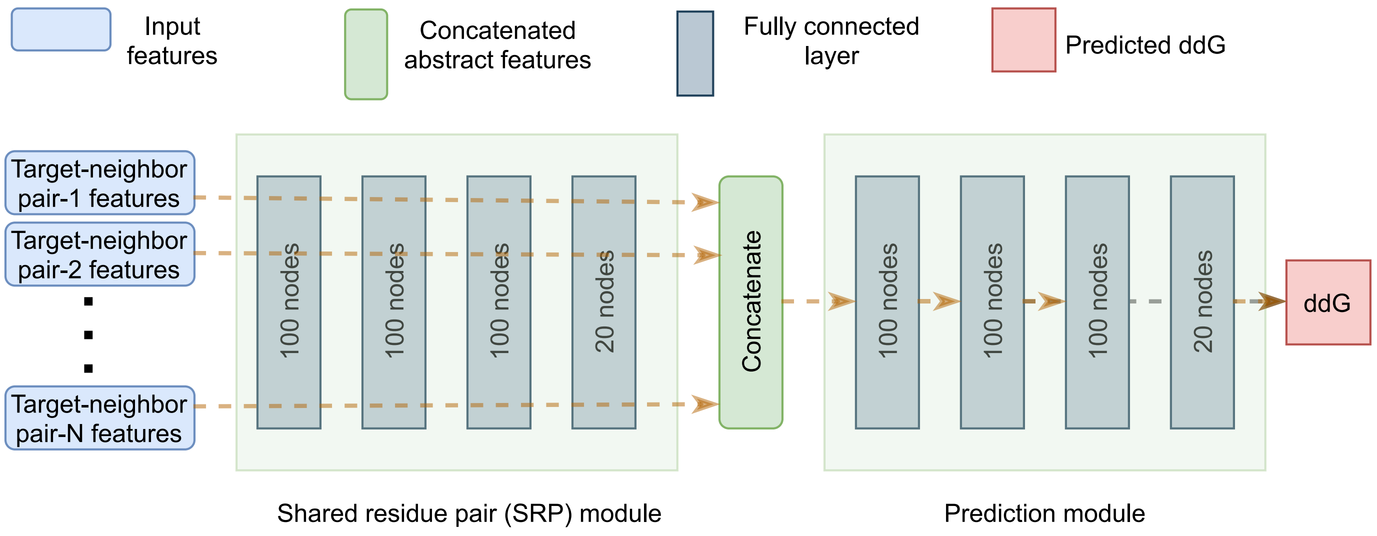
Input features
Concatenated abstract features
Fully connected layer
Predicted ddG
Target-neighbor pair-1 features
Target-neighbor pair-2 features
Target-neighbor pair-N features
100 nodes
100 nodes
100 nodes
20 nodes
Concatenate
100 nodes
100 nodes
100 nodes
20 nodes
ddG
Shared residue pair (SRP) module
Prediction module

Fig. 2. The neural network architecture proposed by the authors.

therefore this dataset provides a fairly small train and test set for the learning task.

Figure 1(d, f) shows the mutation data in different environmental condition of pH and temperature ($^\circ C$). Contessoto *et al.* [4] mentioned that the most important parameter for calculating the $\Delta\Delta G$ value was the pH in their prediction method. Thiltgen and Goldstein [5] argued that the value of $\Delta\Delta G$ may depend on the experimental method used, as well as on the temperature, pH, ionic strength, presence of denaturants, redox state of co-factors, method of protein preparation, etc. But most learning-based methods do not take this environment specification into account. Additionally, mutations not in room temperature can be very unstable. The authors did not consider the environment variable in the learning process and the mutations outside of normal temperature could be an intrinsic error of the learning process in this problem domain.

Finally, Figure 1(e) shows no common proteins between training and testing set. Additionally, the authors mentioned that the sequence similarity between training and testing set is less that 25% which testifies the training and testing set has no overlapping, consequently, independent. Furthermore, a large number of mutation data is dominated by specific proteins, which may cause the model biased towards that protein.

D. Input feature computation

The input feature vector of the network is comprised of target residue features and its neighbor residue features. Following the source paper, the neighbors are found based on Ca-Ca atom distance and the number of neighbors is set to 15. The target residue features are comprised of wild and mutant residue features in the mutation site. Table I shows the summary of the feature computation process. Solvent accessible surface area (SASA), secondary structure, hydrogen bonds, position-specific scoring matrix (PSSM) are computed by Naccess [6], Stride [7], HBPLUS [8] and PSI-BLAST [9], respectively. The other features, such as backbone dihedral angles, distance and orientation of the neighbor residue and residue type encoding are computed from wild protein structure directly, consequently, there are not predicted. Following the authors, PSI-BLAST is run with 3 iterations based on the rp-seq-15 [10] sequence set instead of rp-seq-55 [10] which is a much larger database compared to the previous one because of computational structure shortage. Final feature vector is 51 dimensional where 22 values come from target residue and other 29 corresponds to a neighbor residue.

E. Neural network architecture

The model architecture is a fully connected neural network which is divided into two phases. The network parameters of the first phase is shared among target-neighbor residue pair for N neighbors, called shared residue pair (SRP) network. The SRP network has 27,420 parameters to learn. Then the second phase of fully connected neural network is deployed to make the estimation of $\Delta\Delta G$ values, which has 50,401 learnable parameters. Figure 2 shows the full architecture of the proposed model. The model is trained as end-to-end fashion using mean-squared error (MSE) function, optimized by Adam optimizer. The learning rate is set to 0.001 and the training is performed for 100 epochs. The authors proposed a batch size of 1000. To reduce overfitting, a dropout value of 0.15 at each layer and a L2 kernel regulation value of 0.0008 were used. Rectified Linear Unit (ReLu) activation was used for all layers except the output layer, where a softsign function was used. Since the output of a softsign function is between -1 and 1, the experimental $\Delta\Delta G$ values were normalized using a multiplier of 10 during training. During testing the output of the network was multiplied by 10 to make the final prediction. Figure 3 shows the training (green) and validation (red) MSE loss versus the number of epochs which clearly illustrates an overfitting scenario. Note that, the authors did not mention the training and validation loss. In particular, the SRP and

TABLE I

INPUT FEATURES USED BY THE NEURAL NETWORK. THIS DESCRIBES THE PROPERTY OF EACH FEATURES, THEIR COMPUTATION METHOD AND THE NUMBER OF DIMENSIONS.

Category	**Property**	**Process**	**Implementation**	**#-dimensions**	**Target residue**	**Neighbor residue**
Backbone dihedral	*sin* and *cos* of phi, psi and omega angles		*BackboneDihedral.py*	6	Y	Y
SASA	SASA of a residue	Naccess [6]	*SASA.py*	1	Y	Y
Secondary structure	One-hot encoding	Stride [7]	*SecondaryStructure.py*	3	Y	Y
Number of hydrogen bond	#-backbone-backbone hydrogen bond	HBPLUS [8]	*HydrogenBond.py*	1		Y
	#-backbone-sidechain hydrogen bond			1		Y
	#-sidechain-backbone hydrogen bond			1		Y
	#-sidechain-sidechain hydrogen bond			1		Y
Distance and orientation	Ca-Ca distance from neighbor to target residue		*DistanceAndOrientation.py*	1		Y
	Ca-Ca unit vector from neighbor to target residue			3		Y
	Ca-C unit vector of the neighbor residue			3		Y
	Ca-N unit vector of the neighbor residue			3		Y
PSSM	Softmax PSSM value of wild and mutant residue	PSI-BLAST [9] rp-seq-15 [10]	*PSSM.py*	2	Y	
Residue type encoding	Wild target residue		*TargetResidue.py*	5	Y	
	Mutant target residue		*TargetResidue.py*	5	Y	
	Neighbor residue		*NeighborResidue.py*	5		Y
		Total feature dimension			22	29

the prediction module has 77,821 parameters to learn in total. Additionally, the train and validation set has 3,998 and 345 mutations where each mutation is represented as 15 vectors of size 51. Therefore, there are 2,03,898 input variables for training which is only 2.6 magnitude more. Therefore, this large network is highly prone to overfitting. However, the best model is saved when the best validation loss is found.

IV. EXPERIMENTS AND ANALYSIS

Following the authors, I compute the Pearson's correlation coefficient (PCC) between predicted and expected $\Delta\Delta G$ values. The reconstructed DeepDDG (DeepDDG-recon) achieved PCC of 0.81, 0.29 and 0.21 at train, validation and test set. Additionally, mean absolute error (MAE) and MSE are computed to evaluate the models and they are summarized in Table II. Moreover, Spearman's correlation coefficient (SCC) is computed and it achieved 0.75, 0.39 and 0.19 at train, validation and test set. All of these results confirm the overfitting instance. Comparing with the authors result, the train and test result differs 0.10 which is also overfitted to some

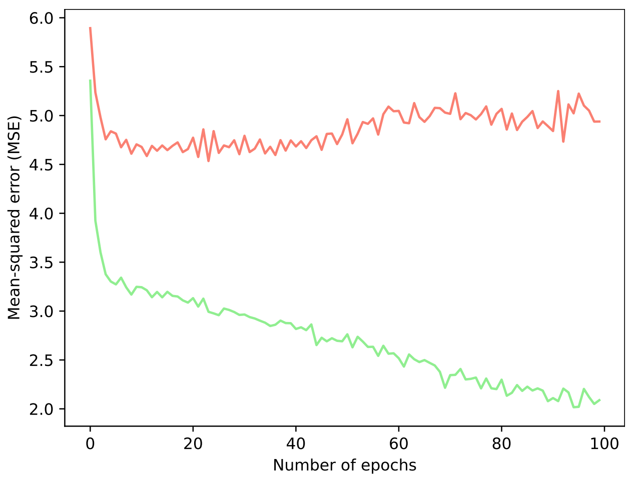

Fig. 3. This depicts mean-squared error loss versus the number of epochs at training time using the exact model and training parameters the authors proposed where the green and red line corresponds to train-loss and validation-loss. It clearly shows an overfitting scenario.

TABLE II
PERFORMANCE COMPARISON

Method	**Train set**				**Validation set**				**Test set**			
	PCC	SCC	MAE	MSE	PCC	SCC	MAE	MSE	PCC	SCC	MAE	MSE
DeepDDG	0.681±0.019		1.04±0.02						0.557±0.015		0.86±0.04	
DeepDDG (After removing pairwise fitness score feature)	0.664		1.06						0.539		0.87	
DeepDDG (After removing protein design probability feature)	0.658		1.08						0.543		0.86	
DeepDDG-recon	0.81	0.75	0.80	1.21	0.29	0.39	1.56	4.93	0.21	0.19	1.09	1.95
DeepDDG-recon (overfitting reduction)	0.38	0.37	0.17	0.04	0.24	0.28	0.20	0.07	0.24	0.24	0.29	0.15

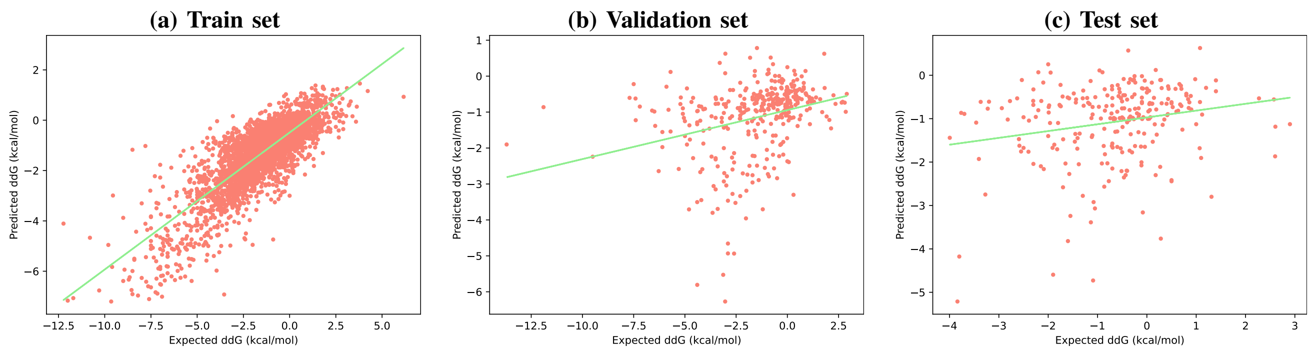

Fig. 4. Comparison between experimental and predicted $\Delta\Delta G$ values obtained by DeepDDG-recon network. The green line shows the linear fitting of all corresponding dataset. The model did not fit well for the validation and test set.

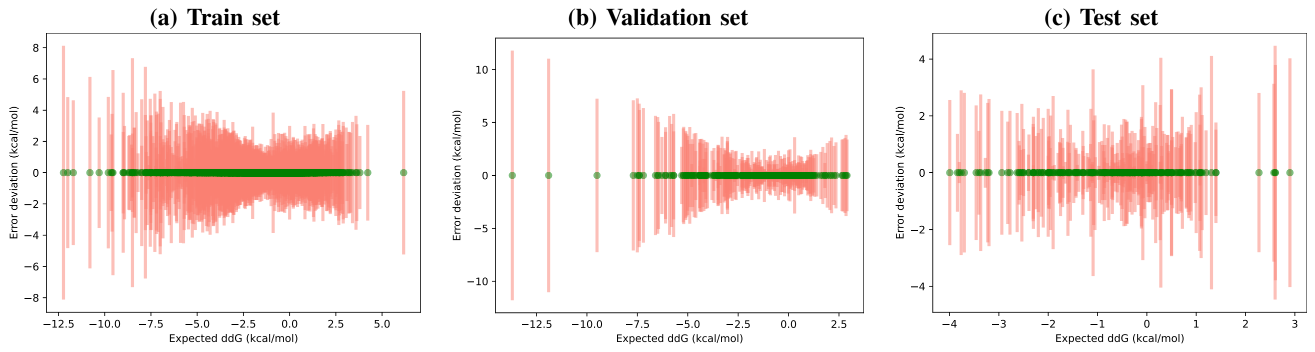

Fig. 5. Deviation of estimated $\Delta\Delta G$ from expected result. The green dots represent the ground-truth $\Delta\Delta G$ values and red bars show the deviation of the predicted ones. The model performs better in case of the ground-truth $\Delta\Delta G$ values being in range [-1, 1].

extent. Additionally, I could not leverage two mutant residue specific features, namely PSSM and pairwise fitness score (PFS). PSSM is computed using rp-seq-15 instead of rp-seq-55 dataset which is 5 magnitude larger than the used one. Additionally, PFS was computed using another database which is size of 20GB, which could not be leveraged because of resource constraints. Therefore, overall input feature dimension is less than the authors.

To understand the rationale of overfitting, I compare the predicted $\Delta\Delta G$ values with the ground-truth. Figure 4 shows that the model is able to learn the pattern for the training set but performs poor for the validation and test set. This may represent, if the validation and test set are the well representative of the training set which are not, then the method may perform better than reported. From Figure 1(b), we can see that the train set is highly biased towards destabilizing mutations where ground truth $\Delta\Delta G$ values are negative. Consequently, the prediction of the models are mostly negative for validation and test set. A further approach can be to train separate models for destabilizing and stabilizing mutations and then combine them together. In another experiment, figure 5 illustrates the deviation of the predicted scores from the expected ones, which clearly pictures that the model performs better if the expected values are in between [-1, 1].

A. Reducing overfitting

There are couple of issues I detected here: 1) the training set is not large enough compared to the proposed model

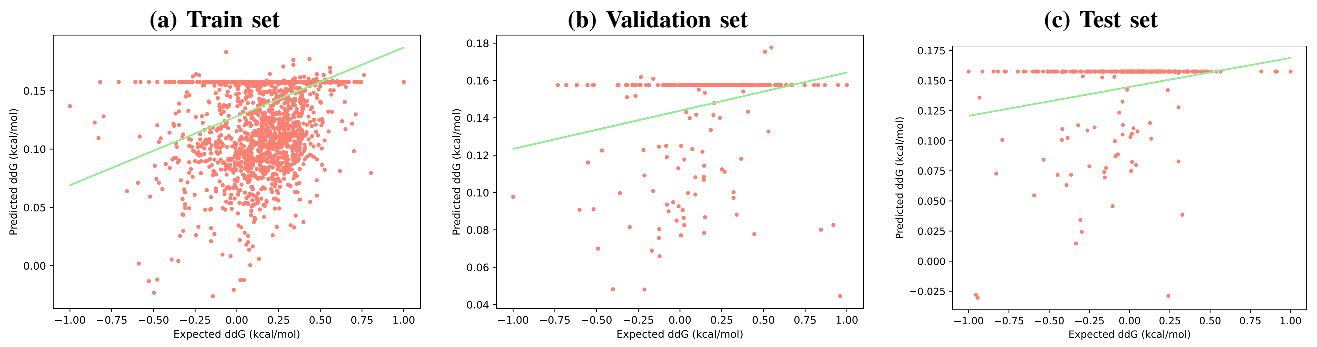

Fig. 6. Comparison between experimental and predicted $\Delta\Delta G$ values obtained by DeepDDG-recon network. The green line shows the linear fitting of all corresponding dataset. The model did not fit well for the validation and test set.

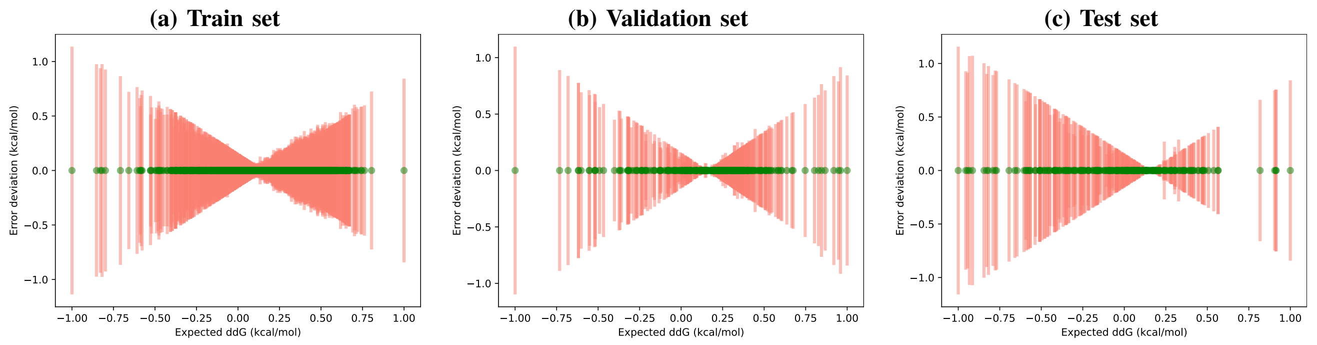

Fig. 7. Deviation of estimated $\Delta\Delta G$ from expected result. The green dots represent the ground-truth $\Delta\Delta G$ values and red bars show the deviation of the predicted ones. The model performs better in case of the ground-truth $\Delta\Delta G$ values being in range [-1, 1].

architecture 2) the $\Delta\Delta G$ distribution over mutation is biased towards destabilizing mutations 3) the training set is dominated by particular protein's mutations 4) the dataset did not consider environment variables, such as pH or temperature 5) the number of learnable parameters is very large, consequently very complex model 6) the feature extraction process may induce additional error towards model training.

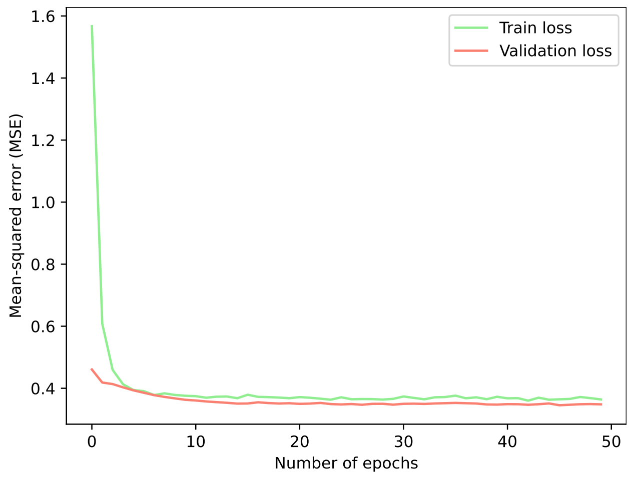

Fig. 8. Train and validation loss and no overfitting.

Therefore, I take some measures to solve the overfitting issue by some extent. First, I reduce the model complexity by reducing the number of layers. Now, SRP and prediction module has 7,220 and 2,302 parameter; second, scale the the ground truth $\Delta\Delta G$ values within a fixed range, such as [-1, 1]; third, add classification (stabiling or destabiling) loss to enforce the model learn for stabilizing mutation category. This model achieved PCC of 0.38, 0.24 and 0.24 for train, validation and test set. Additionally, it achieved SCC of 0.37, 0.28 and 0.24 for train, validation and test set. Figure 8 depicts the training and validation loss which shows no overfitting. The result of PCC and SCC also confirm that the new model achieved better performance than the overfitted model.

References

[1] Huali Cao, Jingxue Wang, Liping He, Yifei Qi, and John Z. Zhang. Deepddg: Predicting the stability change of protein point mutations using neural networks. *Journal of Chemical Information and Modeling*, 59(4):1508–1514, Apr 2019.

[2] Argo wiki. Argo Cluster. http://wiki.orc.gmu.edu/index.php/Main_Page, 2021. [Online; accessed 7-November-2021].

[3] V. Potapov, M. Cohen, and G. Schreiber. Assessing computational methods for predicting protein stability upon mutation: good on average but not in the details. 22(9):553–560, June 2009.

[4] Vinícius G. Contessoto, Vinícius M. de Oliveira, Bruno R. Fernandes, et al. TKSA-MC: A web server for rational mutation through the optimization of protein charge interactions. *Proteins: Structure, Function, and Bioinformatics*, 86(11):1184–1188, September 2018.

[5] Grant Thiltgen and Richard A. Goldstein. Assessing predictors of changes in protein stability upon mutation using self-consistency. *PLoS ONE*, 7(10):e46084, October 2012.

[6] S. Hubbard and J. Thornton. Naccess, computer program. 1993.

[7] Dmitrij Frishman and Patrick Argos. Knowledge-based protein secondary structure assignment. 23(4):566–579, December 1995.

[8] Ian K. McDonald and Janet M. Thornton. Satisfying hydrogen bonding potential in proteins. 238(5):777–793, May 1994.

[9] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, Sep 1997. 9254694[pmid].

[10] Chuming Chen, Darren A. Natale, Robert D. Finn, Hongzhan Huang, Jian Zhang, Cathy H. Wu, and Raja Mazumder. Representative proteomes: a stable, scalable and unbiased proteome set for sequence analysis and functional annotation. *PloS one*, 6(4):e18910–e18910, Apr 2011. 21556138[pmid].