COMP551 Assignment 4

Yijie Zhang[1]*, **Yiping Liu**[1]*, **Tian Bai**[1]*
[1]McGill University
{yj.zhang,yiping.liu,tian.bai}@mail.mcgill.ca

Abstract

In this assignment, we reproduced the results from an ICLR 2023 paper, CDCnov [1], which utilizes a 3+1 Dimensional Graph Neural Network to encode the protein features from sequences and structures, then makes predictions on the protein functional and structural features. To better reproduce the result and further develop this topic, we changed the coding framework, transferring it from Pytorch to Pytorch Lighting, which greatly improved the readability of the code. Also, the re-implemented code maintained a similar performance to the source code. Additionally, since some datasets are not publicly available, we manually collected and cleaned original structure data from the PDB database, and made their format aligned with the original data preprocessing programs. Furthermore, based on available datasets, we added protein single-point mutational effect prediction task to the original model, using Protein Language Models (ESM2) to enhance the extraction of sequential features and tested the performance on the commonly used test datasets. Also, we conducted a thorough ablation study on the new task, illustrating the effect of each component of the model.

1 Introduction

1.1 Original Paper

The paper we are going to reproduce is CDConv [1] published in ICLR 2022. This paper introduces a novel convolution operation named Continuous-Discrete Convolution (CDCnov), which addresses the challenge of modeling the structures of proteins that involve both 3D geometric coordinates and 1D peptide chain sequences. Proteins, being essential biomolecules, exhibit irregular 3D structures due to the uneven distribution of amino acids in space and their continuous coordinate values. In contrast, the 1D sequence structure is regular, with amino acids arranged uniformly and sequentially.

CDConv uniquely combines both irregular and regular modeling approaches. It uses independent learnable weights for different regular sequential displacements, while directly encoding geometric displacements due to their irregularity. This methodology significantly enhances protein modeling by mitigating the impact of geometric irregularities on sequence modeling. The paper proposes and formulates CDConv, then implements it in a (3+1)D structure, creating a hierarchical (3+1)D GNN for geometry-sequence modeling in proteins. This network is applied to tasks such as protein fold classification, enzyme reaction classification, gene ontology term prediction, and enzyme commission number prediction, demonstrating state-of-the-art accuracy.

In conclusion, the paper highlights CDConv as a pioneering convolution class that effectively models irregular and regular data structures. It showcases the potential applications of CDConv beyond protein modeling, such as recognizing non-coding RNAs, and suggests future improvements like integrating self-attention mechanisms for more adaptive local structure capturing.

*Equal contribution.

Preprint. Under review.

1.2 Selected Downstream Tasks

Proteins are crucial biomolecules in organisms [2], essential for catalyzing metabolic reactions [3, 4], DNA replication [5], responding to stimuli [6], providing cellular structure [7], and transporting molecules [8]. Over the past decades, advancements in electron microscopy have allowed researchers to study protein structure and functions at the molecular level [9, 10]. Several typical functional and structural features have been proposed to measure the stability, functionality and other affinities of protein molecules. In this assignment, we selected three of them to evaluate the performance of our re-implemented model.

- **EC**: Enzyme commission (EC) number prediction. Different from enzyme reaction classification, this task aims to predict the three-level and four-level 538 EC numbers. We use the training/validation/test splits of Gligorijevic et al. (2021)[11], which contains 15,550/1,729/1,919 proteins in total. EC number prediction is also a multi-label classification task. We use F_{max} as the evaluation metric.
- **GO**: Gene ontology (GO) term prediction. This task aims to predict the functions of a protein via multiple Gene Ontology (GO) terms, which can be seen as a multi-label classification task. Following Gligorijevic et al. (2021)[11], we classify proteins into hierarchically related functional classes organized into three ontologies: biological process (BP) with 1,943 classes, molecular function (MF) with 489 classes and cellular component (CC) with 320 classes. The dataset contains 29,898/3,322/3,415 proteins for training/validation/test, respectively. The F_{max} accuracy is used as the evaluation metric.
- $\Delta\Delta G$ **prediction**: Since any change in the components of protein molecules could be considered a chemical reaction, the thermostability could be interpreted as the tendency that which a mutation is about to happen, usually denoted as the subtraction of Gibbs folding free energy ΔG for a pair of wild and mutated proteins 1 [12]. A higher $\Delta\Delta G$ value indicates a more stable protein [13]. Therefore, predicting the $\Delta\Delta G$ becomes a numerical prediction task, differing from the previous two classification tasks. In this way, the model's performance could be better evaluated.

$$\Delta\Delta G_{MW} = \Delta G_M - \Delta G_W \tag{1}$$

where M denotes mutated type and W denotes wild type.

1.3 Pytorch and Pytorch Lightning

PyTorch and PyTorch Lightning are both popular frameworks in the world of deep learning, but they serve slightly different purposes. PyTorch, developed by Facebook's AI Research lab, is a comprehensive, flexible, and powerful deep learning framework that provides a rich set of tools and libraries for research and development in machine learning. On the other hand, PyTorch Lightning, built on top of PyTorch, aims to simplify the training process. It abstracts much of the boilerplate code required in PyTorch, making the codebase more readable, maintainable, and less error-prone. This change not only increases efficiency but also makes it easier to scale and reproduce experiments. The reason for transitioning from PyTorch to PyTorch Lightning in our project is to leverage these advantages, particularly for streamlining the training process and focusing more on the model's architecture and experiments rather than the underlying engineering complexities. Also, through this transition, it is easier for us to further develop this model.

2 Method

2.1 Data Collection and Preprocessing

In this assignment, we manually collected the EC and GO datasets since the original code did not provide the downable coordinates. According to the PDB ID provided, we downloaded the original structural data and extracted $C\alpha$ coordinates as the representation of each residue. Then, we aligned them with the corresponding sequence, making it a suitable input for the model.

For the mutational effect prediction task, We derived our initial dataset from PROSTATA [14], an aggregation of various datasets that have been utilized in preceding methodologies. This aggregation

allowed for the selection of samples under unified experimental conditions. The dataset we propose encompasses both wild and mutated sequences, along with corresponding $\Delta\Delta G$ values, comprising 10544 samples, including reverse mutations. To preclude any potential information leakage between the training and test datasets, we employed the TM-Score, a measure signifying structural similarity amongst proteins, and analyzed it between each sample in both datasets. We adhered to the threshold and set it as an average score of 0.5 [15]. Consequently, samples exhibiting a score exceeding this threshold, in comparison to any structures within the testing dataset, were excluded from our dataset. We have provided a comparative analysis in the Appendix A to facilitate our structure selection.

Upon acquiring the refined training dataset, we extracted the backbone coordinates from the corresponding PDB files and aligned them with the sequences. This alignment aimed to generate coherent structure-sequence pairs, streamlining the feature extraction process. For the purposes of our experiment, we adopted the coordinates of the C_α atom to represent the structure of each residue. However, discrepancies in alignment were observed during this process. To address these misalignments, we devised a checking algorithm to discern and select coherent and applicable sequence-structure pairs for subsequent experimentation. The specifics of this algorithm are delineated in the Appendix A.1. Finally, we maintained two subsets for training purposes, with regards to evaluating the model performance on S669.

2.2 CDConv Model Reimplementation

In our implementation, we followed the original structure of the model, which can be found in the models.py file. The hyperparameters for EC and GO tasks are maintained the same as the original source code. In our novel code framework, the dataloader has been wrapped into data_interface.py, which performs as the interface between the model input and originally processed data. The model has been wrapped into model_interface.py. In this way, the general interface could be easily adapted to many other models, ensuring an aligned training and evaluation paradigm.

2.3 Mutational Effect Prediction Model

We separate the process of predicting $\Delta\Delta G$ for single-point mutations into two steps. As per Equation 1, we can predict the ΔG value for each protein and then find the difference between these predictions to get $\Delta\Delta G$. Building on the approach from CDConv [1], we have integrated both sequence and structural information to improve our predictions. Given the lack of mutated structures, we have used the structures of wild proteins as our structural inputs, an approach supported by SCONES [16], showing minimal structural changes in unstable protein mutations. Also, we have added the sequential embedding from the ESM-2 650M UR50D model to our input, enhancing our feature extraction capabilities. More details on the proposed strategies are provided below:

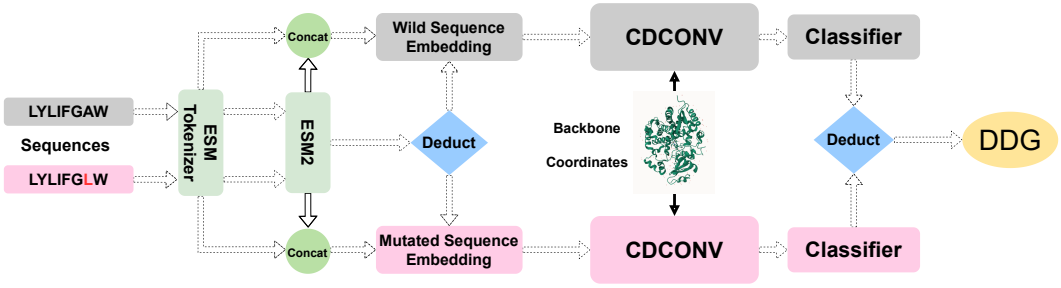

Figure 1: The architecture of our proposed model

System architeture Our model receives sequence-structure pairs for both wild-type and mutated-type as input. The ESM-assisted Continuous Graph Convolutional Network is independently employed on each input to assimilate sequential and structural features into a coherent embedding. Subsequently, we deduct one embedding from the other, transforming it into a one-dimensional prediction via a nonlinear classifier. The detailed system pipeline is shown in Figure 1.

ESM augmented variance enhancement To streamline the extraction of sequence features, we employed and froze the ESM-2 model to derive a sequential representation matrix for each mutation

pair. Subsequently, the obtained embedding was concatenated with the sequence embedding, serving as the input for the graph neural networks. Given the scarce feature variances induced by single-point protein mutations, we also incorporated the difference between the mutated and wild sequence embeddings into both inputs to enhance the feature variance. We have presented the sample input generation process for either mutated or wild type in Algorithm 1. Consequently, this integration aids in accentuating the subtle alterations caused by mutations, allowing the model to effectively discern and learn from the nuanced differences between the mutated and wild-type sequences.

Algorithm 1 Model Input Generation

$SeqInput \leftarrow \text{Concat}(SeqEmbed_{ESM}, SeqEmbed_{CDConv})$
$DiffEmbed \leftarrow SeqInput_{mut} - SeqInput_{wild}$
$FinalSeqInput \leftarrow \text{Concat}(SeqInput, DiffEmbed)$

Loss function Through calculation, the distribution of $\Delta\Delta G$ values reveals that the vast majority of samples are at a range between $[-5, 5]$. Therefore, in order to better conceive the effect of major cases, we adopted the concept of Huber Loss, as denoted in Equation 2

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \leqslant \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \tag{2}$$

Where L_δ represents the Huber Loss, y is the true value, $f(x)$ is the predicted value, and δ is a threshold set to 1.0.

3 Experiment Details

Test Dataset for $\Delta\Delta G$ prediction We assess the effectiveness of our method utilizing widely-used test dataset: the S669 dataset [17].

- **S669 Dataset** is a novel test dataset that is less similar to conventional training sets, including 669 mutations. In order to better analyze the pattern of protein point mutation, we compiled S669_r dataset, which is a reverse in $\Delta\Delta G$ value and the mutation points, based on the assumption that such reaction is reversible [18]. For better evaluation, we employed a refined training dataset, encompassing 6951 samples. These samples exhibit no structural similarity to those within the S669 dataset, ensuring the robustness and reliability of our assessment.

Metrics We chose to employ the Pearson Coefficient (PCC), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE) as metrics to provide a comprehensive assessment of our model's performance. The Pearson Coefficient offers insights into the linear correlation between predicted and true values, RMSE accentuates larger errors providing a focus on substantial inaccuracies, and MAE presents a clear picture of the model's predictive capability by measuring the average magnitude of the errors.

3.1 Implementation Details

For the CDConv experiement, In the experiments, we trained the model for 25 epochs with a batch size of 64 and using the OneCycle optimizer with a learning rate of 0.001. The hyperparameters are obtained by conducting 5-fold cross-validation for best-averaged model performance in the validation datasets. The model was implemented based on the standard PyTorch Geometric [19] library and Pytorch Lighting Framework. We ran the models on Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz CPU and NVIDIA A100 GPU.

3.2 Result of Reproducing CDConv

For the experiment result of reproducing CDConv , as seen in Table 1, our re-implementation outperforms the original method in the GO-cc, Go-bp, and EC tasks, leading to 0.098, 0.134, and 0.047 respectively. However, the original method performs better at the GO-mf task, showing a 0.076

Table 1: Comparision of Our Re-implemented Results and Original Results

95%	Go-cc	Go-mf	Go-bp	EC
Original	0.479	0.654	0.453	0.820
Ours	0.577	0.578	0.587	0.867

advantage. The results indicate that CDConv itself is a robust method that could be reproduced on different machines/frameworks, and get even better results.

3.3 Evaluation on S669 dataset

In this assignment, we trained the model using the training datasets proposed, specifically focusing on the S669 and S669_r datasets. We employed a 5-fold cross-validation process to ensure equitable use of all data samples during training. The experiments were conducted with various hyperparameter combinations to determine the optimal settings for our model, which are detailed in Appendix 3.1. We then assessed the model's performance on the testing dataset by evaluating its performance in each fold and computing the average of these five results. The outcomes from both the original and reversed S669 datasets are presented in Table 2, offering a thorough evaluation of the model's effectiveness.

Table 2: The evaluation results on the S669 and S669_r datasets.

Method	S669			S669_r			Runtime(minutes)
	PCC ↑	RMSE ↓	MAE ↓	PCC ↑	RMSE ↓	MAE ↓	
Ours	0.51	1.46	1.07	0.51	1.46	1.07	1.9

From the results revealed, it could be concluded that for both testing datasets, the PROSTATA method with reduced samples shows a slight increase in RMSE and MAE compared to its counterpart, indicating a minor compromise in prediction accuracy. Notably, our proposed method consistently performs on par with the PROSTATA method with similar sample sizes, while drastically improving the runtime efficiency, since our method freezes the gradient of the ESM model thus it does not need to update the parameters in it. In both testing datasets, our method achieves nearly 15 times faster processing than PROSTATA without compromising much on the accuracy metrics. These results highlight the efficacy of our method in balancing accuracy and computational efficiency.

3.4 Ablation Study

To elucidate the contribution of each component of our model, we performed a detailed ablation study. From Table 3, our proposed model demonstrates the best performance across both datasets in all the metrics. When we remove the ESM embedding, there is a drastic decline in performance, especially in PCC values, which drop to 0.25, respectively. This suggests that ESM embedding plays a pivotal role in capturing the intricate patterns in the data. On the other hand, excluding the CDConv embedding results in a more subtle performance degradation. Although the decline is not as pronounced as in the ESM ablation, it is still noteworthy, particularly in the PCC metric. This indicates that while CDConv embedding contributes to the model's performance, the ESM embedding appears to be more critical for the model's efficacy.

Table 3: The ablation study of our model on S669 dataset.

Method	Metrics		
	PCC ↑	RMSE ↓	MAE ↓
Ours	0.51	1.46	1.07
w/o ESM embedding	0.25	1.90	1.44
w/o CDConv embedding	0.48	1.54	1.13

4 Conclusion and Discussion

In our assignment, we reproduced the results in CDConv in a different code framework and conducted thorough research on predicting the $\Delta\Delta G$ of protein single-point mutation. We assembled a dataset of aligned sequence-structure pairs, ensuring no information leakage with prevalent testing datasets to maintain fairness in model comparison and evaluation. Additionally, we introduced an ESM-augmented prediction model, leveraging both sequence and structure features. In a majority of the tasks, including Go-cc, Go-mf and EC, our re-implemented results are better, compared to the original results.

In our approach, the ESM model is frozen during the acquisition of input embeddings, which are subsequently stored locally, enhancing the model's computational efficiency. This approach relies solely on authentic PDB files, avoiding the use of computational protein structure predictors like AlphaFold2, making it more accessible for researchers to conduct experiments independently.

References

[1] Hehe Fan, Zhangyang Wang, Yi Yang, and Mohan Kankanhalli. Continuous-discrete convolution for geometry-sequence modeling in proteins. In *The Eleventh International Conference on Learning Representations*, 2022.

[2] Martin Karplus and John Kuriyan. Molecular dynamics and protein function. *Proceedings of the National Academy of Sciences*, 102(19):6679–6685, 2005.

[3] Alessandro Michelucci, Thekla Cordes, Jenny Ghelfi, Arnaud Pailot, Norbert Reiling, Oliver Goldmann, Tina Binz, André Wegner, Aravind Tallam, Antonio Rausell, et al. Immune-responsive gene 1 protein links metabolism to immunity by catalyzing itaconic acid production. *Proceedings of the National Academy of Sciences*, 110(19):7820–7825, 2013.

[4] D Michael Gill and Roberta Meren. Adp-ribosylation of membrane proteins catalyzed by cholera toxin: basis of the activation of adenylate cyclase. *Proceedings of the National Academy of Sciences*, 75(7):3050–3054, 1978.

[5] Stephen J Benkovic, Ann M Valentine, and Frank Salinas. Replisome-mediated dna replication. *Annual review of biochemistry*, 70(1):181–208, 2001.

[6] Nelo Eidy Zanchi and Antonio Herbert Lancha. Mechanical stimuli of skeletal muscle: implications on mtor/p70s6k and protein synthesis. *European journal of applied physiology*, 102:253–263, 2008.

[7] E Hesper Rego, Lin Shao, John J Macklin, Lukman Winoto, Göran A Johansson, Nicholas Kamps-Hughes, Michael W Davidson, and Mats GL Gustafsson. Nonlinear structured-illumination microscopy with a photoswitchable protein reveals cellular structures at 50-nm resolution. *Proceedings of the National Academy of Sciences*, 109(3):E135–E143, 2012.

[8] Thomas Zeuthen. Water-transporting proteins. *Journal of Membrane Biology*, 234:57–73, 2010.

[9] Bozhen Hu, Jun Xia, Jiangbin Zheng, Cheng Tan, Yufei Huang, Yongjie Xu, and Stan Z. Li. Protein language models and structure prediction: Connection and progression, 2022.

[10] Paul J Carter. Introduction to current and future protein therapeutics: a protein engineering perspective. *Experimental cell research*, 317(9):1261–1269, 2011.

[11] Vladimir Gligorijevic, P Douglas Renfrew, Tomasz Kosciolek, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C Taylor, Ian M Fisk, and Hera et al. Vlamakis. Structure based protein function prediction using graph convolutional networks. *Nature communications*, 2021.

[12] Nobuhiko Tokuriki and Dan S Tawfik. Stability effects of mutations and protein evolvability. *Current opinion in structural biology*, 19(5):596–604, 2009.

[13] Angel L Pey, François Stricher, Luis Serrano, and Aurora Martinez. Predicted effects of missense mutations on native-state stability account for phenotypic outcome in phenylketonuria, a paradigm of misfolding diseases. *The American Journal of Human Genetics*, 81(5):1006–1024, 2007.

[14] Dmitriy Umerenkov, Tatiana I Shashkova, Pavel V Strashnov, Fedor Nikolaev, Maria Sindeeva, Nikita V Ivanisenko, and Olga L Kardymon. Prostata: Protein stability assessment using transformers. *bioRxiv*, pages 2022–12, 2022.

[15] Yang Zhang and Jeffrey Skolnick. Tm-align: a protein structure alignment algorithm based on the tm-score. *Nucleic acids research*, 33(7):2302–2309, 2005.

[16] Yashas BL Samaga, Shampa Raghunathan, and U Deva Priyakumar. Scones: self-consistent neural network for protein stability prediction upon mutation. *The Journal of Physical Chemistry B*, 125(38):10657–10671, 2021.

[17] Corrado Pancotti, Silvia Benevenuta, Giovanni Birolo, Virginia Alberini, Valeria Repetto, Tiziana Sanavia, Emidio Capriotti, and Piero Fariselli. Predicting protein stability changes upon single-point mutation: a thorough comparison of the available tools on a new dataset. *Briefings in Bioinformatics*, 23(2):bbab555, 2022.

[18] Jorge A Vila. Proteins' evolution upon point mutations. *ACS omega*, 7(16):14371–14376, 2022.

[19] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

A Dataset details

A.1 Training dataset

We collected samples from PROSTATA [14], subsequently executing TM-Score calculations between each sample pair in the original dataset and those within the S669 dataset, intended for evaluative comparisons. Subsequently, samples from the original dataset exhibiting a TM-Score exceeding 0.5 with any sample in the testing datasets were deliberately omitted since such a score indicates a notable structural similarity. This task was performed separately for each testing dataset, ensuring more equitable comparisons without compromising the richness of useful samples. The original dataset contains 10544 samples, after filtering similar structures, the subset for testing the S669 dataset has 6951 samples. From this, it is evident that the original dataset revealed a substantial number of similar samples, which have been excluded from the filtered datasets.

A.2 Data Cleaning

We first aligned the original sequences and mutated sequences separately with the sequences extracted from PDB file using the Needleman-Wunsch algorithm. However, inconsistency often appears. In order to make strict alignment, we designed a cleaning procedure that deals with the following situations in the aligned sequences. The situations are simplified and visualized in Figure 2.

- The provided sequence has missing residues. As situation 1 in Figrue 2 denotes, we amend the position with the corresponding residue in the PDB extracted sequence.
- The PDB extracted sequence has missing residues. As situation 2 in Figure 2 denotes, we following the PDB extracted sequence has the standard and thus omitted the residue from the sequence provided.
- The mutation normally appears as situation 3 denotes in Figure 2. Under these circumstances, we reserve the mutated residue from the provided sequence.

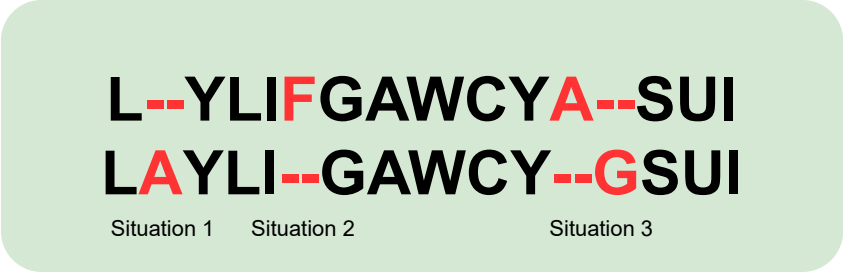

Figure 2: The situations considered in the data cleaning process

It is worth noting that when the neighborhood residue of the mutated position is missing in PDB extracted structures, we are unable to tell if the mutant type is single-point or multiple-point. Therefore, we omitted the sequences under certain circumstances.