
Learning Unknown from Correlations: Graph Neural Network for Inter-novel-protein Interaction Prediction

Anonymous Author(s)

Affiliation

Address

email

Reproducibility Summary

1

2 *This is a report of reproducibility of paper (1), submitted to ML Reproducibility Challenge 2021.*

3 **Scope of Reproducibility**

4 In the paper the authors propose a new evaluation that respects inter-novel-protein interactions, and also a new method,
5 that significantly outperforms previous PPI methods, especially under this new evaluation. Therefore we will first
6 inspect if this kind of evaluation is objectively better, and secondly, we will try to reproduce the results of the proposed
7 model in comparison with previous state-of-the-art, PIPR (2).

8 **Methodology**

9 For the reproduction we used authors code, slightly changing the pipeline for automatization. We also used PIPR code,
10 where we completely changed the pipeline, to be able to use it on the same datasets as GNN-PPI, but used their function
11 for building the model. The experiments were run on Nvidia Titan X GPU, using around 250 GPU hours altogether.

12 **Results**

13 We reproduced the papers results within standard deviations of our repeated experiments. But in some cases, this still
14 means there is a big difference between the performances, which is coming from different train-test splits of the newly
15 proposed splitting schemes. Even with these discrepancies we still managed to (at least partially) confirm all authors
16 claims. The proposed model GNN-PPI performed better than PIPR overall and for inter-novel-protein interactions,
17 evaluation on their proposed schemes predicted the generalization performance better, and their model is also robust for
18 predictions for newly discovered proteins – here our results were surprising, they were even better when the network
19 was built knowing fewer proteins.

20 **What was easy**

21 It was easy to run GNN-PPI code on different datasets and with different parameters, as their repository is nicely
22 organized and the code is clearly structured. It was also easy to understand their idea of the problem, the reasons for
23 new evaluation and the framework of their proposed model.

24 **What was difficult**

25 In both models used in this reproduction, the environment setup was harder than expected. There was no documentation
26 or comments in the code, which made it hard at first to understand it. Some debugging was needed for GNN-PPI and a
27 lot of code changes for PIPR to train well.

28 **Communication with original authors**

29 We communicated with authors through email. They provided some useful clarifications of the method and pipeline.

30 1 Introduction

31 The study of multi-type Protein-Protein Interaction (PPI) is fundamental for understanding biological processes from
32 a systematic perspective and revealing disease mechanisms. Existing methods suffer from significant performance
33 degradation when tested on different dataset, that was not used for training (in comparison to only dividing one dataset
34 into train and test set). In this paper, authors investigate the problem and find that it is mainly attributed to the poor
35 performance for inter-novel-protein interaction prediction. However, current evaluations overlook the inter-novel-protein
36 interactions, and thus fail to give an instructive assessment.

37 As a result, they propose to address the problem from both the evaluation and the methodology. Firstly, they design a
38 new evaluation framework that fully respects the inter-novel-protein interactions and gives consistent assessment across
39 datasets. Secondly, they propose a graph neural network based method (GNN-PPI), that uses correlations between
40 proteins for better inter-novel-protein interaction prediction. Experimental results on real-world datasets of different
41 scales demonstrate that GNN-PPI significantly outperforms state-of-the-art PPI prediction methods, especially for the
42 inter-novel-protein interaction prediction.

43 2 Scope of reproducibility

44 Since authors propose a new evaluation framework, our first task will be to critically inspect their methodology, that is
45 based on different construction of train and test sets. We need to confirm that this kind of dataset splitting is objectively
46 better and mimics the real world problem that they are trying to solve.

47 Secondly, we will try to reproduce experimental results on real-world datasets of different scales, that were shown in
48 the paper. The authors compare their model to a series of baseline algorithms, some Machine Learning based models
49 (Support Vector Machines, Random Forest and Linear Regression) and some Deep Learning based models (PIPR (2),
50 DNN-PPI (3) and DPPI (4)). Since most of the author claims were made on comparisons of GNN-PPI with PIPR, we
51 will try to reproduce those. The main claims of the original paper, that we will test are the following:

- 52 1. GNN-PPI has **higher micro-F1 score** than PIPR on SHS27k, SHS148k and STRING datasets, using *random*,
53 *BFS* or *DFS* testset construction strategies.
- 54 2. GNN-PPI predicts **inter-novel-protein interaction** better than PIPR on the same datasets and testset construc-
55 tion strategies as in claim above.
- 56 3. Test performance on trainset-homologous testset (trained and tested on same set: SHS27k or SHS148k) under
57 *BFS* and *DFS* partition schemes **reflects the performance of generalizing knowledge** to unseen testset
58 (trained on SHS27k or SHS148k and tested on STRING) better than using *random* partition scheme – micro-F1
59 scores are more similar, when training and testing GNN-PPI on data split in such way.
- 60 4. If we construct the PPI network in GNN-PPI only from the trainset, the micro-F1 score is still better than the
61 one of PIPR. This shows that the trained model is **robust to newly discovered proteins** and their interactions.

62 3 Methodology

63 For the reproduction we used authors code, slightly changing the pipeline for automatization of multiple runs with
64 different seeds. We also used PIPR code, where we completely changed the pipeline, to be able to use it on the same
65 datasets as GNN-PPI. We only used their function `build_model`, in which we changed last layer activation function
66 from *softmax* to *sigmoid*, because otherwise the model failed in predicting multiple interaction types.

67 3.1 Model descriptions

68 In our experiments we used the author’s model, GNN-PPI and the previous state-of-the-art for multi-type protein
69 interaction prediction, PIPR.

70 3.1.1 GNN-PPI

71 **GNN-PPI** is a **graph neural network** based method that uses **correlation** between two protein features to predict
72 multiple types of their interaction. Pairwise interaction data are firstly assembled to build the graph, where proteins

73 serve as the nodes and interactions as the edges. The model is developed by constructing an embedding for each
74 protein to obtain predefined features, then processed by Convolution, Pooling, BiGRU and FC modules (it is called
75 GIN network) to extract protein-independent encoding (PIE) features, which are aggregated by graph convolutions and
76 arrive at protein-graph encoding (PGE) features. Embeddings are pretrained for each amino-acid and combined for the
77 proteins by their amino-acid sequences. The last is Multi-label PPI prediction. For unknown PPIs, we combine their
78 protein feature encoded by the previous process with a dot product, and then use a fully connected layer as classifier for
79 multi-label PPI prediction. For optimization the authors use Adam optimizer.

80 3.1.2 PIPR

81 **PIPR** employs a **Siamese** architecture of residual **RCNN** encoder to better apprehend and utilize the mutual influence
82 of two sequences. It uses the same pretrained embeddings as GNN-PPI, which are then send through the RCNN, from
83 which we get sequence embedding vectors. This are multiplied with a dot product to form a sequence pair vector.
84 Finally, this sequence pair vector is fed into a multi layer fully connected network with categorical cross-entropy loss
85 function, to predict the multi-label PPI prediction.

86 3.2 Datasets

87 We trained and tested this two models on three different databases (and their combinations):

- 88 (1) **STRING**: this database collected, scored, and integrated most publicly available sources of protein-protein
89 interaction information and built a comprehensive and objective global PPI network, including direct (physical) and
90 indirect (functional) interactions. In this paper, we focus on the multi-type classification of PPI by **STRING**. It
91 divides PPI into 7 types: reaction, binding, post-translational modifications (ptmod), activation, inhibition, catalysis,
92 and expression. Each pair of interacting proteins contains at least one of them. We use all PPIs of Homo sapiens,
93 which contains 15,335 proteins and 593,397 PPIs.
- 94 (2) **SHS27k**: randomly selected 1690 proteins of Homo sapiens subset of **STRING**, that have 7624 PPIs between them.
- 95 (3) **SHS148k**: randomly selected 5189 proteins of Homo sapiens subset of **STRING**, that have 44488 PPIs between
96 them.

97 The interactions from the datasets were combined into labels, so that multiple lines that represent different types of
98 interactions between two proteins are combined into one datapoint, where label is a vector of length 7 (number of
99 interaction types) with ones on the indices of interactions that are present and zeros elsewhere.

100 The authors split datasets so that the test set contained 20% of the interactions. Splitting schemes will be described in
101 the experimental setup more thoroughly, as their evaluation approach was one of their main contributions to the field.
102 Whole datasets are available on the authors GitHub repository, and the combined ones, used for PIPR training can be
103 found on our repository.

104 3.3 Hyperparameters

105 For GNN-PPI we used the hyperparameters that authors described in Table 10 in the paper. The only change was setting
106 batch size to 2048 when training on **STRING** dataset, as that was much faster, and the authors also used that in the code
107 on their repository.

108 For PIPR we did a manual hyperparameter search, to find the parameters where the model works best. We tried different
109 batch sizes, 64, 128, 256, 512, 1024 and 2048; different learning rates, 0.1, 0.01, 0.001 and 0.0001; numbers of epochs,
110 100, 200 and 300; and RMSprop and Adam optimizers. As the training takes a lot of time, we didn't train the model
111 until the end for all options. With batch sizes, we fully trained it only on smaller datasets and compare the results. We
112 saw that smaller batch size slows the training time a lot, so when using it, we needed to lower the number of epochs.
113 For **SHS27k**, we saw that it is better to train for 200 epochs with batch size 128, than to do less epochs with batch size
114 64 (or more with bigger batch size), so we chose this combination. For **SHS148k** batch size 256 gave better results
115 than 128, and with bigger performance fell. On **STRING**, the only option was to take batch size 1024, as others were
116 too slow, and with 2048 training also slowed, because we ran out of memory on GPU. Here we only did 100 epochs,
117 because after that, we couldn't use the GPU anymore. We set optimizer to Adam after it proved better on couple runs
118 on the smallest dataset. Learning rate was set just based on first couple epochs (10-20), as we quickly saw that with

119 0.0001 the training loss hardly even falls, and that with 0.01 and 0.1 it only falls at the beginning, and after couple of
120 epochs, the model stops learning. So we set it to 0.001 for all datasets.

121 3.4 Experimental setup and code

122 One of the main authors claims is that the usual evaluation scheme (randomly splitting the interactions into train and
123 test set) is not correct from the protein interaction point of view. The result of such split is, that the majority of the
124 proteins were already seen during training, and it is much easier to predict the interactions for such proteins, then for
125 some completely new. To show this, they separate the test set into three subsets: X_{BS} denotes interactions where both
126 proteins were already seen during training, X_{ES} denotes those where one protein was seen, and X_{NS} those interactions,
127 where both proteins are first seen in test phase. In the random testset construction, the X_{NS} set is almost empty (which
128 can be also seen in Table 2), meaning that the testing is not representative for new proteins.

129 To solve this, authors propose two new strategies, Breath-First Search (BFS) and Depth-First Search (DFS), where the
130 testset is constructed by firstly selecting the root node and then performing the proposed BFS or DFS strategy to select
131 other nodes for the test set (0.2 of the whole dataset in our experiments). As they more thoroughly explained in the
132 paper, these strategies seem to mimic the real world case, where some new cluster of proteins, that tightly interact with
133 each other, is found (BFS) or we just have some new proteins distributed around the previously known network (DFS).

134 As the authors did in the paper, we then compared the models based on their micro-averaged F1 score (which is the
135 same as the accuracy giving each sample the same importance). We repeated the experiments 5 times where it wasn't
136 too computationally expensive, to get the uncertainty and see how reliable our results are. We were not able to do
137 multiple runs on STRING dataset for each setting, as each run was very expensive, but we believe that the uncertainty
138 here would be very small, as we have a very big dataset. The code for all our experiments is available on this repository.

139 3.5 Computational requirements

140 The experiments were run on Nvidia Titan X GPU, using around 250 GPU hours altogether. We used different
141 environments for each model – to easily use them we uploaded environment files on our repository.

142 In Table 1 you can see separate times of training on 300 epochs for GNN-PPI (which was the default setting described
143 in the paper) in minutes and prediction on the test set (0.2 of each dataset) in seconds. We can see that training takes
144 quite some time, especially for bigger datasets (18 hours for STRING dataset, but this is not unexpected for a deep
145 learning based approach), but once the model is learned, the prediction is very fast: in range of 10^{-4} of a second per
146 one protein pair.

147 Learning and prediction times are much slower for PIPR model, especially for smaller datasets, where we also took
148 smaller batch size, as it substantially improved the performance. For this reason we only did 200 epochs of training for
149 SHS27k and SHS148k, and it was still much slower than with GNN-PPI. On the STRING set we only did 100 epochs,
150 as we ran out of GPU time we could afford. Long training time also made it harder for us to thoroughly check all
151 hyperparameters and find the best values, as we couldn't possibly do 200 epochs for each parameter choice combination.
152 We can also see that prediction of PIPR is slower, but that is probably because of model using default predict batch size,
153 which is 32 (GNN-PPI uses 256) and could be improved. Even if not, we have around 500 predictions per second, so
154 prediction time shouldn't be a problem.

Table 1: Training and prediction times.

	Training time [min]			Testing time [s]		
	SHS27k	SHS148k	STRING	SHS27k	SHS148k	STRING
GNN-PPI	11	63	1064	0.2	1.2	44
PIPR	36	135	602	5	23	291

155 4 Results

156 In this section we will present the results of the experiments that we did to support the claims above. Our results were
157 in some parts far from the authors results, but even so, they support the main claims of the original paper.

158 **4.1 Comparison of GNN-PPI and PIPR**

159 Results in this subsection refer to our first two claims. In Table 2 we can see the comparison of the models in question
 160 on all three datasets, trained and tested with all three partition schemes. For first two datasets we ran the experiments 5
 161 times, with 5 different random seeds, to see how the performance changes on different sets. We can see that especially
 162 for SHS27k the standard deviation for *bfs* and *dfs* splits is very big, which also explain why our results are at some
 163 points quite far from the authors. For STRING, multiple runs were too expensive, so we only trained and tested once.
 164 But this should be enough to asses the authors claims, because the dataset is so much larger, that the randomness of the
 165 split effects the performance less.

166 The comparison of general performance, that refers to the first claim, can be seen if we look at the micro-F1 score
 167 averaged across whole dataset (column *average*). Except for BFS mode on the smallest dataset, where uncertainty is
 168 too big, we **can confirm** that in all other cases **GNN-PPI performs better**.

169 The second claim is that GNN-PPI predicts inter-novel-protein interactions better. The subsets we are observing are
 170 denoted with *bs*, standing for both proteins being seen during training, *es*, either of the proteins seen during training and
 171 *ns*, neither of the proteins seen before. So for the second claim, we need to compare the performance on subsets *es* and
 172 *ns*. The results in the table are **in bold where** uncertainty is small enough that **we can confirm it**. We can also observe
 173 that even where it is not, mean is never bigger for PIPR.

Table 2: **Results on test subsets for GNN-PPI (G) and PIPR (P)** on all datasets and splitting schemes. The results show mean F1-micro scores and their standard deviations, based on 5 runs on SHS datasets, and one run on STRING.

Dataset	Scheme		Average	Test subsets			Test subset sizes		
				X_{BS}	X_{ES}	X_{NS}	$\#BS$	$\#ES$	$\#NS$
SHS27k	rand	G	0.88 ± 0.01	0.89 ± 0.01	0.72 ± 0.04	0.44 ± 0.16	1429 ± 11	91 ± 10	4 ± 2
		P	0.81 ± 0.01	0.82 ± 0.01	0.6 ± 0.01	0.25 ± 0.24			
	BFS	G	0.54 ± 0.12	/	0.58 ± 0.12	0.37 ± 0.16	0	1180 ± 103	368 ± 9
		P	0.49 ± 0.05	/	0.52 ± 0.05	0.37 ± 0.08			
	DFS	G	0.66 ± 0.12	/	0.68 ± 0.12	0.5 ± 0.15	0	1353 ± 35	191 ± 25
		P	0.48 ± 0.06	/	0.5 ± 0.06	0.35 ± 0.08			
SHS148k	rand	G	0.92 ± 0.0	0.92 ± 0.0	0.75 ± 0.02	0.19 ± 0.29	8649 ± 26	245 ± 26	3 ± 3
		P	0.85 ± 0.0	0.85 ± 0.0	0.63 ± 0.03	0.12 ± 0.21			
	BFS	G	0.57 ± 0.05	/	0.59 ± 0.05	0.39 ± 0.05	0	7699 ± 352	1265 ± 332
		P	0.5 ± 0.01	/	0.51 ± 0.01	0.37 ± 0.02			
	DFS	G	0.79 ± 0.03	/	0.8 ± 0.03	0.66 ± 0.06	0	8247 ± 82	707 ± 83
		P	0.54 ± 0.03	/	0.55 ± 0.03	0.36 ± 0.07			
STRING	rand	G	0.95	0.95	0.72	0.99	118410	268	2
		P	0.87	0.87	0.57	0.33			
	BFS	G	0.85	/	0.86	0.75	0	107018	11689
		P	0.58	/	0.59	0.51			
	DFS	G	0.89	/	0.89	0.83	0	112328	7058
		P	0.62	/	0.63	0.47			

174 **4.2 Generalization performance**

175 In this subsection we will inspect the third claim, that says that with the newly proposed evaluation protocol, we are
 176 better assessing the models generalization abilities. To test that, authors trained both models (we will reproduce results
 177 for GNN-PPI) on SHS27k and SHS148k datasets and tested them on the bigger STRING dataset. The test sets of both
 178 smaller datasets were now used as validation sets during training, to determine the model from which epoch should be
 179 taken as best. The authors then compared these to the results on trainset-homologous testset (here authors don't use
 180 validation during training, as they use that set for testing).

181 The results in Table 3 **confirm claim 3**. As we can see generalizing accuracy severely drops when using random
 182 partition scheme (~ 0.2). With newly proposed schemes we get similar, sometimes even better performance when
 183 testing on STRING set.

Table 3: **Results of GNN-PPI generalization.**

Method	Trainset	Testset	Partition Scheme		
			<i>random</i>	<i>BFS</i>	<i>DFS</i>
GNN-PPI	SHS27k-Train	SHS27k-Test	0.88 ± 0.01	0.54 ± 0.12	0.66 ± 0.12
		STRING	0.62 ± 0.01	0.56 ± 0.14	0.61 ± 0.04
	SHS148k-Train	SHS148k-Test	0.92 ± 0.0	0.57 ± 0.05	0.79 ± 0.03
		STRING	0.72 ± 0.01	0.7 ± 0.01	0.7 ± 0.02

184 4.3 Robustness for unknown proteins

185 In Table 4 we have the comparison of different PPI graph construction methods - *GCA* means that the PPI graph was
 186 constructed from all proteins in the dataset, and *GCT* means that PPI graph was constructed only from the proteins
 187 in the train set. This results **confirm claim 4**, but in a different way as those in the original paper. It is true that the
 188 performance with *GCT* construction method is still better than PIPR method, but contrary to authors results, we get
 189 even better performance from the *GCT* construction method than with *GCA*.

Table 4: **Performance of GNN-PPI with different graph construction method.**

Scheme	Graph	Dataset	
		<i>SHS27k</i>	<i>SHS148k</i>
BFS	<i>GCA</i>	0.54 ± 0.12	0.57 ± 0.05
	<i>GCT</i>	0.66 ± 0.1	0.66 ± 0.04
DFS	<i>GCA</i>	0.66 ± 0.12	0.79 ± 0.03
	<i>GCT</i>	0.71 ± 0.09	0.8 ± 0.02

190 5 Discussion

191 New evaluation protocol which proves superiority of a newly proposed model could be questionable, because the
 192 authors would only select a new evaluation that speaks for their model. But in this case, they also compared the models
 193 by random strategy, which is the previously used evaluation method. The new evaluation also has empirical studies to
 194 support it and is based on the domain knowledge, so we think that the **authors proposed a good framework** that can
 195 be very useful in future research in this field as such performance prediction is more on par with real world situation.

196 When comparing GNN-PPI and PIPR model, we confirm that **GNN-PPI has better average performance than PIPR**.
 197 We should also observe, that the standard deviation for new partition scheme BFS on the small dataset is too big for us to
 198 make certain claims. But we see that on the bigger dataset, it becomes smaller, so we can rely more on this evaluations.
 199 There is a bit less certainty when assessing second claim – comparing results in columns X_{ES} and X_{NS} from Table 2.
 200 First let’s observe that the size of X_{NS} is too small in random partition scheme for all datasets, so we won’t compare
 201 models on it. Next, we see that with BFS split scheme, training useful models is much harder than for other two. We
 202 need a lot more data for models to perform well, and if we have it, then we can claim that GNN-PPI performs better
 203 at least when one of the proteins in the pair is seen in training. For both unseen before, we can’t claim with enough
 204 certainty. So let’s say that we **partially proved the second claim** - it holds for random and DFS scheme, and for BFS
 205 on bigger datasets. There is one more shortcoming of our approach. We ran out of time to run 200 epochs of training
 206 PIPR on STRING dataset. If we had the time, the result could be better, and could potentially beat GNN-PPI. We
 207 propose this to be checked in further reproduction.

208 If we compare absolute scores of GNN-PPI with those that authors described in the paper, they vary quite a bit. This
 209 happens because the standard deviations are so big, and the authors might (could be unintentionally) chose the better
 210 results of their runs. Their in-depth analysis on test subsets also didn’t include uncertainty, which is a problem, as we
 211 can see from our results that show that uncertainty is very big and so their results are not reliable.

212 **We confirmed that the new evaluation protocol is much better in assessing the models generalization abilities,**
213 which was the main problem of the field that the authors were trying to solve. An important thing to observe here is
214 also that when using any model for prediction of PPI, we can use this evaluation protocol and predict its performance in
215 much more detail. We know how many of the proteins that we are interested in are completely new, and how many
216 were already known when training the model, so we can predict the performance separately for any new set, based on
217 performances on test subsets BS, ES and NS.

218 When comparing the robustness of predictions for unknown proteins, we need to ask ourselves why can we even use
219 the GCA method, that actually uses test data in network construction during training. In most cases that would be a
220 big mistake, but if we look from a protein perspective, this has a practical explanation. We could know which protein
221 exist, what amino-acids they are built of, so we could put them into the network, but we wouldn't know anything about
222 their interactions yet. If we are interested into such proteins, than GCA evaluation will tell us more about our models
223 performance. If we want to know how our model will perform for some newly discovered proteins, we need to inspect
224 evaluation using GCT. In Table 4 we can see a big discrepancy with authors results. They were showing that with GCT
225 the performance falls, which would be expected because we have less information, but in our case, the mean results are
226 better with GCT construction. If we look at the standard deviations, we again see that the differences between results of
227 multiple runs are big, so we can't say with certainty that this results show that GCT is better than GCA. The reason for
228 this could also just be different root node at splitting schemes, and therefore different clusters of unknown proteins. But
229 in any case, it shows that the performance of GNN-PPI with graph construction only from the trainset is much better
230 than the performance of PIPR, which **confirms the fourth claim** and further proves the usefulness of GNN-PPI.

231 In this reproduction, we left out some parts of the paper that we think should be further verified. We only took PIPR
232 model, and didn't inspect the others for the same tasks. Except for overall performance, also authors didn't compare
233 themselves with other methods, but we believe that for saying that their model is state-of-the-art in inter-novel-protein
234 interaction prediction, they should also inspect other models more thoroughly. With a lot of trouble with setting up
235 PIPR and long training times, we ran out of time for this additional experimentation. We also just superficially grasped
236 the hyperparameter selection for PIPR, and with proper grid search (that would require a lot of computing resources),
237 we could also find better settings for it, or in other case, we could more confidently say that GNN-PPI is better. We
238 also left out the separate results for labels, and ablation study, for which there was no code, but since it could be easily
239 implemented, we suggest it to be done in future. We didn't do it because of time restrictions, and because the authors
240 said in our communications, that it was not the part of original paper, only an addition for requirements of the reviewer,
241 included in arXiv version.

242 **5.1 What was easy**

243 The easiest part of this reproduction was to understand the paper and it's ideas. The authors motivate and describe
244 the problem in an easy to understand way, supported by expressive graphics. They support the ideas with domain
245 knowledge, which makes the paper much more insightful. The algorithm for GNN-PPI is clearly described in the paper,
246 so we believe that it would be possible to put it in code without major problems. But their code is also nicely structured
247 and easy to run with various parameters, so we suggest you use it when in need for their method.

248 **5.2 What was difficult**

249 For GNN-PPI there was also no environment file, and in the environment descriptions, versions of some libraries were
250 left out, so it took some time to set the environment correctly. You can load the environment file from our repository, to
251 avoid this problem. There was also no documentation of the authors code, so when inspecting the implementation, it
252 was difficult to recognize use of some variables. There was some minor debugging needed before the code ran smoothly.
253 We couldn't use the already trained models for any of our tasks. If we are least judging from their names, they were
254 trained only using the train set, so only useful for checking the accuracy on test part of the same dataset. But this can't
255 be done, because they didn't upload the split information, so we can't build this exact test set. If we wanted to use the
256 models on some other dataset, they would need to upload the model with best validation accuracy, as that one would
257 probably be better.

258 This were all minor problems compared to the difficulty of running the PIPR model. Their environment file was not
259 useful, and the versions they described on repository were not compatible with each other, which meant that we spent
260 a lot of time finding out which libraries we should load to run their code. The code was not documented and it was

261 completely unreadable, so it was hard to even know what they were doing. In the end we only used the model building
262 part, and wrote other parts ourselves. Even the building function needed some changes, because they used wrong
263 activation function on the last layer for the model to be able to learn multi-type prediction. Even its description in their
264 paper was not very explanatory, but the reason for that was probably that it wasn't the main part of their research.

265 **References**

- 266 [1] G. Lv, Z. Hu, Y. Bi, and S. Zhang, "Learning unknown from correlations: Graph neural network for inter-novel-
267 protein interaction prediction," 2021.
- 268 [2] M. Chen, C. Ju, G. Zhou, X. Chen, T. Zhang, K.-W. Chang, C. Zaniolo, and W. Wang, "Multifaceted protein-protein
269 interaction prediction based on siamese residual rcnn," *Bioinformatics*, vol. 35, no. 14, pp. i305–i314, 07 2019.
- 270 [3] Y. Gui, R. Wang, Y. Wei, and X. Wang, "Dnn-ppi: A large-scale prediction of protein–protein interactions based on
271 deep neural networks," *Journal of Biological Systems*, vol. 27, no. 01, pp. 1–18, 2019.
- 272 [4] S. Hashemifar, B. Neyshabur, A. A. Khan, and J. Xu, "Predicting protein–protein interactions through sequence-
273 based deep learning," *Bioinformatics*, vol. 34, no. 17, pp. i802–i810, 2018.