Classification Task Over Protein-Protein Interactions

Emre Kucuk

July 2020

1 Introduction

Protein - Protein Interactions (PPI) cause vital processes such as growing, division or maintenance of a cell. It is an important topic in order to understand the functioning of any cell with itself and others. Even though the most reliable techniques are experimental for investigating the properties of different structures, numerical methods are much faster with negligible error. Nevertheless, predictions of PPI needs improvements in order to derive more accurate results faster. In this project, we are testing the capability of deep learning algorithms and their performances over protein protein interaction predictions. By using different optimization algorithms and Convolutional Neural Networks, we are tackling the PPI prediction problem as a classification problem and measure performance. We are aiming to compare the performances of different optimization algorithms, which can be considered as a challenge for sparse data. In addition, we are implementing a heuristic algorithm for structural matching in a fast manner. Main motivation is to calculate similarity scores in order to determine relevant protein interfaces for a given arbitrary protein. Doing that in a fast way provides docking methods to work far faster by negligible error.

2 Related Work

2.1 Deep Learning

Deep learning models are studied immensely and more and more studies are conducted as time goes. They are widely used in different fields such as image recognition, sentiment analysis, anomaly detection, and so on. In recent years, deep learning algorithms are used in bioinformatics field extensively, to find answers to different problems such as protein folding [11]. In this project, we are trying to train a model for protein protein interaction prediction, which is another vital problem for bioinformatics.

To tackle the PPI prediction problem, we used Convolutional Neural Networks (CNN) and acted prediction problem as an image classification problem.

Normally, protein structures are kept as PDB files [4]. To make a neural network understand protein files, we used a prepared data set as three dimensional data for atom coordinates. More detailed information can be obtained from "Data" and "Experiments" section.

2.2 Structural Comparison

QuickRet is a hashing algorithm for protein interfaces. Main point is to determine which cluster a certain given protein belongs to. In order to understand the effectiveness and the motivation behind QuickRet, it is vital to understand some concepts.

There are thousands of experimentally determined structures of proteins. Some of them contains a protein itself, some contain different proteins interacting. These interactions are similar in various ways, and in literature there are studies that clustered all interactions in order to prevent redundancy [5]

In PIFACE [5], there are clusters in order to prevent redundancy amongst protein interfaces. All of Protein-Protein Interactions (PPI) are clustered, thus redundancy is prevented. Each cluster has a representative, that represents the members of its cluster. A useful idea might be to check for similarities for an arbitrary protein with these clusters. Let's call this arbitrary protein as query protein. That way, we can determine the specifications of a query protein and investigate its properties, possible interaction candidates. This way, we can computationally calculate its binding properties or possible strong interactions much accurate.

Computational methods are by far faster, and knowing a certain proteins interaction candidates may lead more precise experimental results. Moreover, cruciality of computational methods are undeniable. Template-based PPI Prediction methods are proven to be successful [3] [13]. Though experimental methods are the most reliable methodologies for obtaining new information on a certain structure, computational methods can lead way to experimental methods since they can be done faster and produce bulk results. However, computational methods have a long way, they can be more faster, accurate.

QuickRet aims to decrease the calculation time of PPI's and remaining the accuracy same. As discussed above, template-based PPI prediction algorithms are successful. Template-based method means that to predict an interaction between two arbitrary protein P_a and P_b, we can use another interaction let's say between P_c and P_d as a reference. By that, PRISM tries to dock P_a and P_b each other as if they are interacting similar like P_c and P_d. Algorithm first makes structural alignment to find best possible template[12]. Considering the numbers of templates, finding the correct template takes a lot of time. QuickRet tries to reduce time to find possible templates by discarding the most irrelevant templates in a fast manner.

3 Data

Dataset for training models are gathered from different sources: DOCKGROUND[9], PPI4DOCK[14], PIFACE[5] and Protein Data Bank[4]. Most of the positive interfaces are gathered from the combination of PIFACE and Protein Data Bank. Data for negative class are obtained from DOCKGROUND and PPI4DOCK.

There are several challenges to create a balanced and meaningful data set for protein protein interactions. Creating a class that contains already existing or predicted interfaces is an easy task. Challenging part is to create a meaningful class of negative data for the data set.

Positive class is determined by documented entries in the Protein Data Bank. On the other hand, to determine the negative class, Critical Assessment of Predicted Interactions[6] is used. Using data obtained from PPI4DOCK and DOCKGROUND and measuring their CAPRI score, an extensive negative interface dataset is created in this thesis [2], which more detailed information can be found in it.

Statistically, dataset contains train, validation and test parts. Train part contains 255580 interfaces. Positive and negative classes have same amount of interfaces. Validation set contains 18111 data points, 11611 positive and 6500 negative data points. Lastly, test set contains 27165 data points with 17416 positive and 9749 negative data points.

Data set for heuristic algorithm contained 2 chains for each interface cluster, which means it has $22604 \times 2 = 45208$ elements. To test, we took 1 protein (2 chains) from 420 clusters individually, all containing more than 5 members.

4 Models and Experiments

4.1 Deep Learning

To create a valid and usable model, we are testing the data with a LeNet[10] like neural network with different optimization algorithms. Data is prepared as voxel data, which has three spatial dimensions. Hence, convolutional neural network contains three spatial dimensions, and input/output channels. Data comes with dimensions $(5, 5, 5, 40, 100)$, which means every minibatch contains 100 data points with dimensions $(25, 25, 25, 40)$. Neural network architecture contains two convolution layers and two dense layers. Convolution layer parameters are $(5, 5, 5, 40, 50)$ and $(5, 5, 5, 50, 60)$. Dense layer parameters are $(1620, 920)$ and $(920, 2)$. This means in the first convolution layer kernel dimensions are $(5, 5, 5)$ and number of input channels are 40 and number of output channels are 60. In the second convolution layer, kernel dimensions are same, and number of input channels are 50 and number of output channels are 60. Lastly, we are transferring the information to two fully connected layers with input size of 1620 and 920. Output has 2 nodes, referring to the predictions for each class. For each layer we used ReLU as activation function.

To compare with different results, we ran the tests with three different op-

timization algorithms. First algorithm is Stochastic Gradient Descent, second algorithm is Adam[8], an adaptive optimization algorithm based on previous gradients. Lastly, we trained the neural network with Square-root of Minima of Sums of Maxima of Squared-gradients Method (SM3)[1], a memory efficient adaptive optimization algorithm for large models, using the advantage of activation patterns. Which could be a good choice since our data can be considered as sparse, and SM3 could use its activation pattern advantage in order to obtain satisfying results.

4.2 QuickRet

QuickRet contains two different phases. In the first step, algorithm deposits a representative interface from each non redundant protein clusters to a hash table. Clusters are obtained from PIFACE [5], a non-redundant template data base for protein protein interactions.

After constructing a hash table from the representatives of all clusters that taken from PIFACE [5], we can determine what a query protein is structurally similar to hashed clusters by looking the hash table. QuickRet is a fast algorithm, and the reason behind is it uses hash tables and compare proteins by the most little possible parts.

In the first step, algorithm takes representatives from each cluster. Then, it extracts a small part of it called base. A base consists of 4 atoms, 2 Carbon Alpha atoms (CA) and 2 Carbon Beta atoms (CB). CA atoms obtained from different residues between distance of $4\mathring{A}$ to $13\mathring{A}$. CB atoms are obtained from those residues.

For a better understanding, let's call CA_1, CB_1 and CA_2, CB_2 as CA and CB atoms for first and second residue respectively. Algorithm creates a base from these four atoms, then calculates four features. These are:

- Distance between CA_1 and CA_2.
- Bond angle between CB_1, CA_1, CA_2.
- Bond angle between CA_1, CA_2, CB_2.
- Dihedral angle by all atoms.

By using the feature thresholds, we will make integer division amongst all features. Hence, all features will be discretized. Then an item of value of the interface hashtable will be a tuple, as $(base, interfaceid)$ where base is a matrix with dimensions $3x4$, each column is the coordinate of an atom going as CB_1, CA_1, CA_2, CB_2; and interface id is the id of the interface which base is extracted to. There will be many bases with same features (or keys) so all bases will create many lists under same keys. These can be from different interfaces or same interfaces, it does not matter.

This concludes first phase, hence the main cause and functionality of QuickRet structure. In **Algorithm 1**, first phase of QuickRet is presented:

Algorithm 1 QuickRet - Phase 1

```
1: Data: List of interface structures
2: Result: Hash table of features extracted from the interface structures
3: initialize hashtable
4: for interface in the interface list do
5:     read interface from PDB file
6:     assign a model number i to the interface
7:     list of feature-base pairs ← extractFeatures(interface)
8:     for f, b in list of feature-base pairs do
9:         key ← discretize f
10:        insert (b, i) into hashtable(key)
```

In second phase, algorithm takes an input as query protein. Main object is to find the best similar structures. Algorithm starts by extracting the same features as templates in the first phase. Then, by using the discretized version of features, finds the matches in hashtable. It is clear that same thresholds must be used in both phases.

If a match occurs between query protein's features and haslist(key), algorithm stores those matches in a new table called *matchlist*. After this operation is done for all extracted features, *matchlist* contains all similar base matches between query protein and interfaces, categorized by template id's.

By looking the lengths of *matchlist* bins, we can say the longest matches and shortest matches. Algorithm uses this value as a score. One important key to note is interface lengths might differ, so when creating the score we are dividing each *matchlist* bin length to number of features coming from related template for normalization. This is the first scoring.

After first scoring, algorithm iterates through all interfaces and all elements inside of it. When iterating, calculates structural alignment parameters using Kabsch algorithm [7].

Kabsch algorithm returns 3 parameters, two vectors for transforming each component's center of mass to origin, and lastly a matrix for rotation operations. QuickRet takes first two vectors, and calculates resultant vector. Secondly, by rotation matrix, algorithm uses Euler angle extraction method and derives 3 rotation angles.

By using resultant vector and rotation angles as key parameters, algorithm creates a third hash table for storing transformations. Keys are also discretized. This time hash tables are individually calculated for a single interface. Searcing through transformation hash table results, algorithm finds the longest one. Then algorithm creates a second score by two ways: first is dividing the length of longest transformation to length of related matchlist bin. Second is very similar, only difference is taking square of length of longest transformation, then dividing it to matchlist length.

For more rigorous understanding, second phase, scoring of a query protein phase can be observed in **Algorithm 2**.

Algorithm 2 QuickRet - Phase 2

Data: Query Protein
Result: Similarity score for each interface
read protein structure from PDB file
list of feature-base pairs ← **extractFeatures**(protein)
initialize *matchlist*
for f, b_q in list of $feature - base$ pairs **do**
    $key \leftarrow$ discretize f
    retrieve $(b_{inteface}, i)$ pairlist from $hashtable(key)$
    for $(b_{interface}, i)$ in pairlist **do**
        insert $(b_{interface}, b_q)$ into $matchlist[i]$
initialize scoretable
$score_1 \leftarrow$ **calcScore**(matchlist)
for i in *matchlist* **do**
    initialize posetable
    for $(b_{interface}, b_q) inmatchlist[i]$ **do**
        $transformation \leftarrow$ **calcTransformation**$(b_{interface}, b_q)$
        key ← discretize $transformation$
        insert transformation into posetable(key)
    bestpose ← **length**(**max**(posetable))
    $score_2 \leftarrow$ **calcScore**(bestpose, **length**(matchlist[i]))
    scoretable[i] ← $score_1 + score_2$

We conducted different tests by changing parameters of QuickRet. From PIFACE [5], we randomly choose one protein from different clusters where number of cluster members are 5 or above. We have chosen 820 chains from 410 proteins. For classification scores we held different metrics.

Normally, in docking-based methods such as PRISM, algorithm uses structural alignment [12] to find best matching template. Considering there are 22604 possible templates [5], this operation costs time. Hence, we measured QuickRet's calculation time for 820 protein chains and kept top one, top ten, top fifty and top hundred results.

5 Results

5.1 Deep Learning

For a difficult task, different optimization algorithms performed quite different from each other. Yet, we can say that they tried to achieve their goals. Amongst all three optimization algorithms, Adam[8] performed best. For every 100^{th} iteration, we tested the neural network using validation sets. Train and validation set results can be seen from the images below.

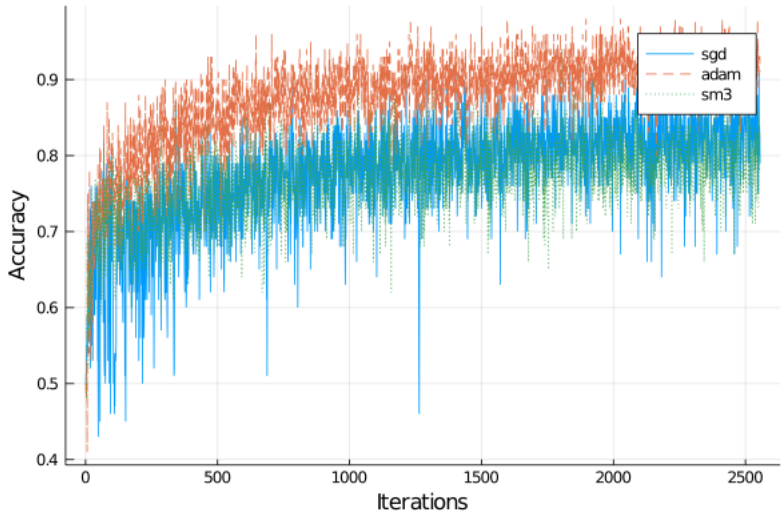

Figure 1: Train data accuracy for all three optimization algorithms. Minibatch size: 100

From this figure, we can say that Adam[8] performed significantly better than Stochastic Gradient Descent and SM3[1]. Pairwise comparisons can be seen from the figures below.

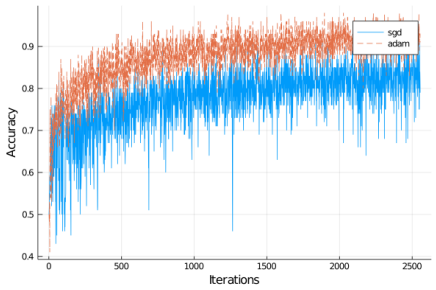

Figure 2: SGD vs Adam comparison.

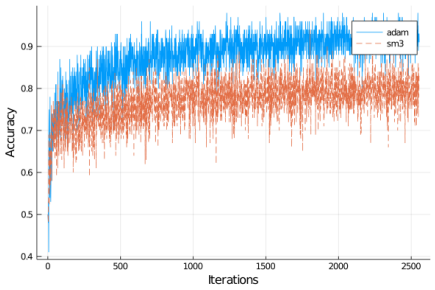

Figure 3: Adam vs SM3 comparison.

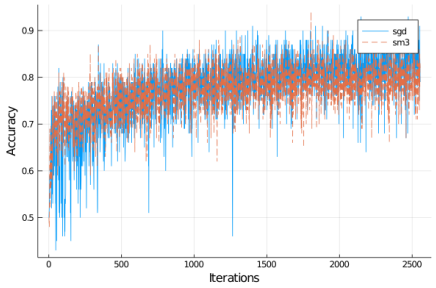

Figure 4: SGD vs SM3 comparison.

Neural network's performance over validation set can be seen below.

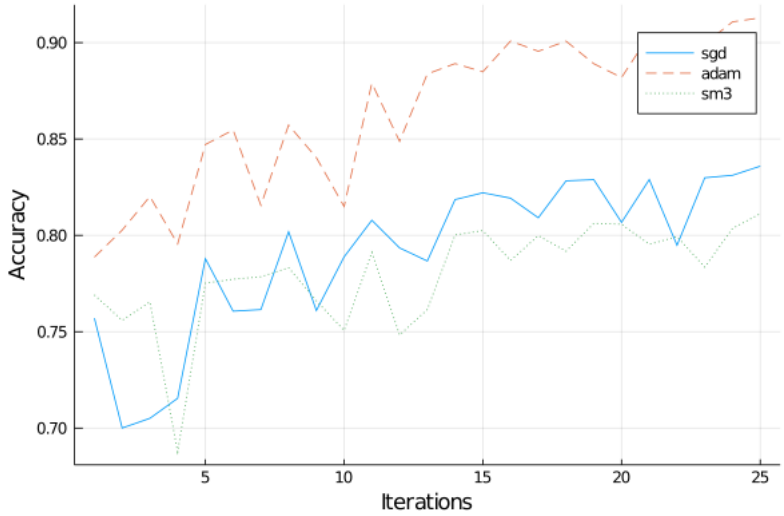

Figure 5: Validation data accuracy for all three optimization algorithms. Mini-batch size: 100

Adam start significantly better and keeps its accuracy higher. Other algorithms performed nearly close to each other.

5.2 Structural Comparison

We calculated similarity scores for 820 chains. There were 8 different tests. We tested with two different similarity scores.

- First score: $\frac{length(matchlist[i])}{sum(features_i)} + \frac{length(max(bestpose_i))}{length(matchlist[i])}$
- Second score: $\frac{length(matchlist[i])}{sum(features_i)} + \frac{(length(max(bestpose_i)))^2}{length(matchlist[i])}$

By using these two score functions, we conducted top one, top ten, top fifty and top hundred classification tests. If the correct interface were in one of the first specified range, it counted as correct.

Calculation Time (s)	First Score	Second Score
Top One	4.027219	4.150832
Top Ten	42.626253	43.118639
Top Fifty	214.092662	213.319734
Top Hundred	423.350253	424.268901

Accuracy performances are given in the table and graph below.

Accuracy (%)	First Score	Second Score
Top One	0.11	0.3402
Top Ten	0.21	0.356
Top Fifty	0.25	0.362
Top Hundred	0.27	0.3683

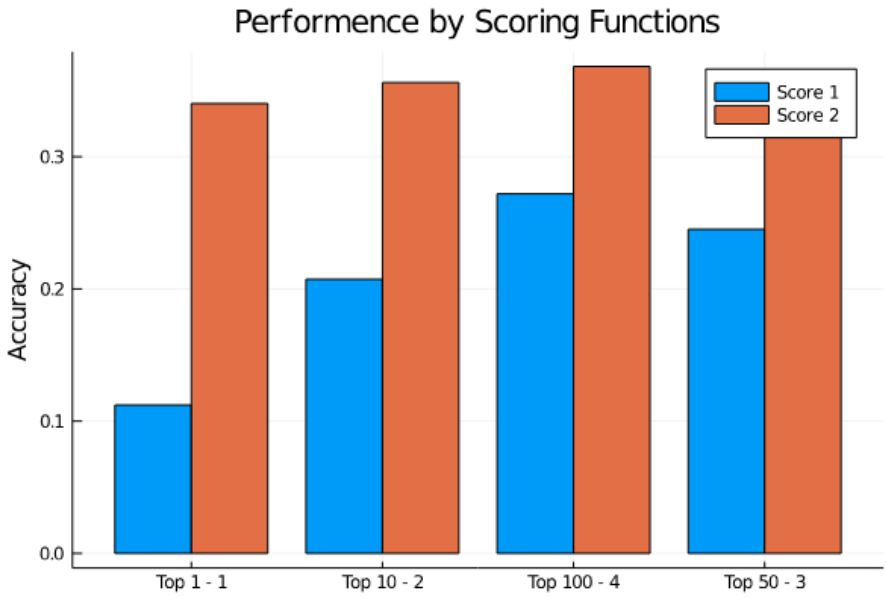

Figure 6: QuickRet accuracy performance compared with two different scoring function.

6 Conclusion

Both algorithms performed sufficiently. Comparing deep learning models with heuristic algorithms, it is clear that deep learning algorithms have great future in bioinformatics and computational structural biology, yet heuristic algorithms are still preferable for speed and filtering irrelevant structures for unrelated interfaces. I think computer science have a lot of competence and future in biological systems but still there needs to be done a lot research about it.

References

[1] Rohan Anil, Vineet Gupta, Tomer Koren, and Yoram Singer. Memory-efficient adaptive optimization for large-scale learning. *arXiv preprint arXiv:1901.11150*, 4, 2019.

[2] Ali Tuğrul Balcı. *Fast Screening Algorithms for Protein Interface Structural Alignments*. PhD thesis, Koç University, 2018.

[3] Alper Baspinar, Engin Cukuroglu, Ruth Nussinov, Ozlem Keskin, and Attila Gursoy. Prism: a web server and repository for prediction of protein–protein interactions and modeling their 3d complexes. *Nucleic acids research*, 42(W1):W285–W289, 2014.

[4] Stephen K Burley, Helen M Berman, Charmi Bhikadiya, Chunxiao Bi, Li Chen, Luigi Di Costanzo, Cole Christie, Ken Dalenberg, Jose M Duarte, Shuchismita Dutta, Zukang Feng, Sutapa Ghosh, David S Goodsell, Rachel K Green, Vladimir Guranović, Dmytro Guzenko, Brian P Hudson, Tara Kalro, Yuhe Liang, Robert Lowe, Harry Namkoong, Ezra Peisach, Irina Periskova, Andreas Prlić, Chris Randle, Alexander Rose, Peter Rose, Raul Sala, Monica Sekharan, Chenghua Shao, Lihua Tan, Yi-Ping Tao, Yana Valasatava, Maria Voigt, John Westbrook, Jesse Woo, Huanwang Yang, Jasmine Young, Marina Zhuravleva, and Christine Zardecki. RCSB Protein Data Bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. *Nucleic Acids Research*, 47(D1):D464–D474, 10 2018.

[5] Engin Cukuroglu, Attila Gursoy, Ruth Nussinov, and Ozlem Keskin. Non-redundant unique interface structures as templates for modeling protein interactions. *PloS one*, 9(1):e86738–e86738, 2014.

[6] Joël Janin, Kim Henrick, John Moult, Lynn Ten Eyck, Michael JE Sternberg, Sandor Vajda, Ilya Vakser, and Shoshana J Wodak. Capri: a critical assessment of predicted interactions. *Proteins: Structure, Function, and Bioinformatics*, 52(1):2–9, 2003.

[7] W. Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 34(5):827–828, Sep 1978.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Petras J. Kundrotas, Ivan Anishchenko, Taras Dauzhenka, Ian Kotthoff, Daniil Mnevets, Matthew M. Copeland, and Ilya A. Vakser. Dockground: A comprehensive data resource for modeling of protein complexes. *Protein Science*, 27(1):172–181, 2018.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[11] Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[12] Maxim Shatsky, Ruth Nussinov, and Haim J Wolfson. A method for simultaneous alignment of multiple protein structures. *Proteins: Structure, Function, and Bioinformatics*, 56(1):143–156, 2004.

[13] Nurcan Tuncbag, Attila Gursoy, Ruth Nussinov, and Ozlem Keskin. Predicting protein-protein interactions on a proteome scale by matching evolutionary and structural similarities at interfaces using prism. *Nature protocols*, 6(9):1341, 2011.

[14] Jinchao Yu and Raphaël Guerois. PPI4DOCK: large scale assessment of the use of homology models in free docking over more than 1000 realistic targets. *Bioinformatics*, 32(24):3760–3767, 08 2016.