Protein-Protein Interaction Prediction from Amino Acid Sequence using Stacked Autoencoder

Sneha Suhitha Galiveeti
ssg163@psu.edu
906591347

April 30, 2020

Contents

1 Motivation

Proteins are very important structures present in all the living organisms which are responsible for carrying out all the biological functions. Understanding the structure of proteins and their interactions with other proteins is very helpful as this knowledge can help in understanding of many vital processes in our bodies such as antibody production, growth of tissues, development of diseases etc. Such knowledge can lead to discovery of new drugs and can help the mankind.

But the main limitation to the discovery of new proteins and their interactions in the lab manually is very tedious, time consuming and expensive. Due to these reasons, Machine Learning techniques can be used to predict the new protein protein interactions which can later be tested and verified in the lab, as these machine learning models can handle a large volume of data and can give results considering many parameters.

2 Goal

The main goal of this project is to address a research question on identifying the possibility of interaction between two proteins. Therefore, this project tries to sole the problem of the prediction of possible interactions between a given pair of proteins.

3 Problem statement

This project aims to perform a binary classification on the given pair of proteins using an autoencoder combined with a softmax layer for binary classification. The weights to this encoder are initialized with the weights generated from the autoencoder after it is trained with the data.

4 Code Repository

All the implemented code and the supporting code taken from [1], along with the dataset used and the results generated are pushed to my repository in github with name "PPI-prediction-from-sequence-using-Autoencoders" [2]. This is the link to this repository: https://github.com/SuhithaG/PPI-prediction-from-sequence-using-Autoencoders

5 Related Work

- The paper by Tanlin et al.[3], have used Stacked autoencoders to predict the protein-protein interactions from their sequence using 10 fold cross-validation and Conjoint method. They have shown that 10 fold cross-validation method produced better results on comparison with the Conjoint Triad method. The dataset used in this paper is the one which this project also uses and is taken from HPRD and Swiss-prot database. The best model in their paper have produced an accuracy of 97.19%.

- The paper by Gui et al.[4] have proposed a new coding technique combining both the Auto Covariance and Conjoint Triad method to use the advantages of both the techniques. They have used a simple DNN model on this new coding technique to train and get the results. This paper also uses the same dataset and I have collected the data from this paper's reference. Their best model has produced an accuracy of 98.137%.

- Apart from these two papers, I have also referred two literature review papers to understand the research question and know the existing literature. The paper[5], has described about all the possible domains in biology whose research questions can be answered through deep learning and machine learning techniques. It also discusses about the type of machine learning techniques which were used to solve the questions. The paper[6] is a literature review on data driven methods used to predict protein interfaces. This paper has a good review on what evaluation metrics to be considered to validate a model.

6 Dataset

Two types of dataset are used in this project, one to train and test the model and the other to only test the validity of the model. The datasets are described below:

- Benchmark Dataset - This dataset is provided by Pan et al.[7] and is downloaded from their web server. From this dataset, the positive protein pair samples are taken from Human Protein Reference Database, version 2007 and the negative samples are taken from Swiss-prot database, version 57.3. The data is preprocessed as mentioned by Gui et al.[4]. From this cleaned data, 30,000 positive pairs and 30,000 negative pairs of protein samples are randomly taken to form the training dataset. The remaining 12,915 are considered as test data. The summary is as follows:

  - Training data: 48,000 pairs
  - Validation data: 12,000 pairs
  - Test data: 12,915 pairs

- External datasets - Four other datasets are also used to test the implemented model. As mentioned in [4], the datasets are provided by [8]. The following are the four datasets:

  - *C. elegans*: 8060 (4030 positive + 4030 negative pairs)
  - *Drosophila*: 43,950 (21,975 positive + 21,975 negative pairs)
  - *E. coli*: 13,908 (6954 positive and 6954 negative pairs)
  - *H. sapiens*: 74,064 (37,027 positive and 37,027 negative pairs)

Apart from this, each protein pair is uniquely represented by the set of extracted features resulting from coding their amino acid sequence. The coding technique can be either one of Conjoint Triad method or Auto covariance method. If the method is conjoint Triad

then 686 features for each protein pair are generated which means our training data will be of size 60,000 x 686. If the Auto Covariance method is used, then the training data size would be 60,000 x 420.

7 Method

The model proposed involves two steps to achieve this binary classification. Firstly, the given amino acid sequence is to be converted into a representation of features which retains as much information about the protein as possible. Secondly, these features are to be trained on a Deep Learning model to extract the required features which can then predict new interactions of protein pairs. Following are the steps in detail:

1. **Coding of the Protein Sequences** This is an important step in this method since the representation of these sequences are given as input to the deep learning model to train. As we know the model is always as good as the data on which it is trained and the data here is the coded representation of the actual sequence. So, the coding has to be as close to the protein sequence as possible. I have used the following two techniques to code the given protein sequence:

   - **Conjoint Triad Method:** This coding technique is proposed by Shen et al.[9]. As mentioned in [9], the 20 amino acids are classified into 7 classes and the unit of consideration is any possible triad(three consecutive amino acids) in the sequence. A binary space(V,F) is used to represent the protein, where V is a vector storing the features with each v_i is a triad and F is a frequency vector with each f_i storing the frequency of v_i in the protein sequence. Since we have 7 classes of amino acids and the unit is a triad, there will be a total of 343 possible triads(7x7x7). Therefore, V is a vector of size 343. Since the value of F depends on length of the protein sequence and can be very large at times, it has to be normalized to the range 0 to 1. Shen et al. [9] has proposed a new vector D which is a normalized vector of F and is repesented as follows:

     $$d_i = \frac{f_i - min(f_1, f_2, ..., f_{343})}{max(f_1, f_2, ..., f_{343})}$$

     This D is the final representation of each protein. Since our input is pair of proteins, D vector of each protein is concatenated to get the final representation i.e, say two proteins, (D_A and D_B are concatenated to D_{AB} as follows [9]:

     $$D_{AB} = D_A \oplus D_B$$

     Since each D is of size 343, the final representation of each pair of proteins D_{AB} is of size 686.

   - **Auto Covariance:** This is an another technique to represent the features of proteins as mentioned in [10]. According to the paper, each of 20 amino acids are reflected by their seven physicochemical properties namely, hydrophobicity, hydrophicility, volumes of side chains of amino acids, polarity, polarizability, solvent-accessible surface area (SASA) and net charge index (NCI). Therefore,

there exits a descriptor to store the value of each amino acid for each property, say P_{ij}, where i is the amino acid and j is the property. The P is normalized as follows[10]:

$$P_{ij} = \frac{P_{ij} - P_j}{S_j}$$

where P_j is the mean of jth property over 20 amino acids and S_j is the corresponding Standard Deviation.
According to [10], AC variable to represent the sequence is calculated using the following formula:

$$AC_{lag,j} = \frac{1}{n - lag} \sum_{i=1}^{n-lag} \left(X_{ij} - \frac{1}{n} \sum_{i=1}^{n} X_{ij} \right) \left(X_{(i+lag),j} - \frac{1}{n} \sum_{i=1}^{n} X_{ij} \right)$$

where i is the position in amino acid sequence X, j is the property descriptor, n is the length of the sequence X and lag is the distance between residues. From the paper [4], optimal lag is considered to be 30, so the lag for this project is set to 30 which gives 210(30x7) features for each sequence. That is, the D vector representing the features of proteins will be of size 210 and since our input is a pair of proteins, D will be of size 420.

2. **Training of Stacked Autoencoder** Now that I have generated the representations of proteins using either of the methods discussed above, I have then trained my deep learning model with these representations. The deep leaning model which I have chosen is the stacked autoencoders. This step is again a two step process as discussed below:

   (a) **Stacked encoder with decoder:** I have implemented a simple autoencoder with 2 hidden layers. The architecture of the model is as follows:

      i. Encoder: This part of the model learns the important features of the data and can represent the data in lesser dimensions
         - Input layer: This layer takes in the input, where each input is a pair of proteins in the dimensions of 1xN, where N can be 686 or 420 depending on the coding technique.
         - Hidden layer 1: This layer takes the input of dimension 1xN and reduces it to 1x128 and gives it as input to next hidden layer.
         - Hidden layer 2: This layer takes the input of dimension 1x128 and reduces it to 1x64 and gives it as input to code layer.

      ii. Code: This layer store the data in minimum possible dimensions which when given to decoder can retrieve the original data. In our model the dimension of features in this layer is 1x32.

      iii. Decoder: This part of the model can give back original data given the feature representation in reduced dimension. The dimensions of layers in this part are the exact reverse of encoder part.
         - Hidden layer 1: This layer takes in the input of dimension 1x32 and gives out the features in 1x64 dimensions.

- Hidden layer 2: This layer takes in the input of dimension 1x64 and gives out the features in 1x128 dimensions.
- Output layer: This layer takes in 1x128 features and gives the data in original dimensions which is 1xN.

Following is a representation of the model architecture described above: This

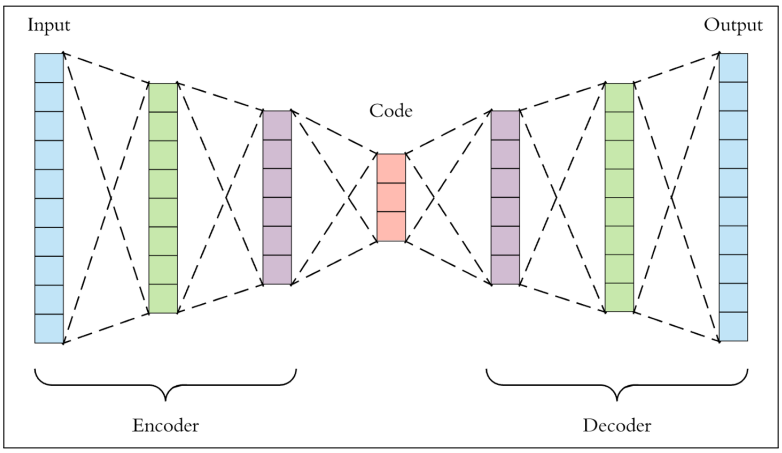

Figure 1: Model Architecture[11]

model is then trained with the data and the weights generated after the training are stored. I have trained this model for 20 epochs with a batch size of 500.

(b) **Encoder with softmax layer:** This is the second model which I have implemented and this model has only the encoder and it's architecture is same as that of the encoder used above. Now, to the output of this encoder, a softmax layer is attached to label the data into two classes 0 or 1, 0 being non interactive pairs and 1 being interactive pairs.

3. **Testing the model** After the model is trained, it is both tested with the benchmark dataset and the dataset from four other species. The results generated are evaluated and discussed below.

8 Evaluation Metrics Used

The accuracy, Precision, Recall and F1 score are the metrics which I have used to quantify the performance of the proposed model.

1. Accuracy = $\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$
2. Precision = $\frac{\text{TP}}{\text{TP+FP}}$
3. Recall = $\frac{\text{TP}}{\text{TP+FN}}$
4. F_1 Score = $2 * \frac{Precision * Recall}{Precision + Recall}$

where TP, TN, FP and FN denote the True Positives, True Negatives, False Positives and False Negatives respectively.

9 Results

9.1 Conjoint Triad Method:

Following are the plots generated using the Conjoint Triad Method as the coding for the sequences.

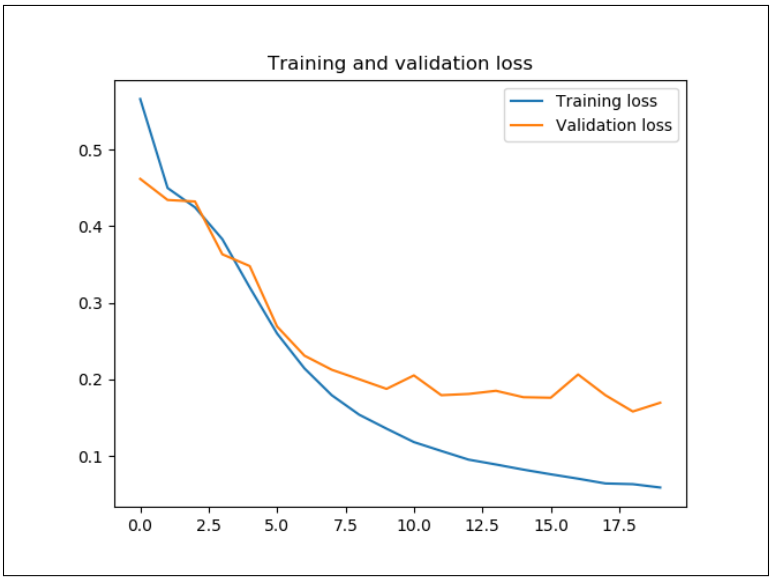

Figure 2: Plot of training and validation loss for 20 epochs

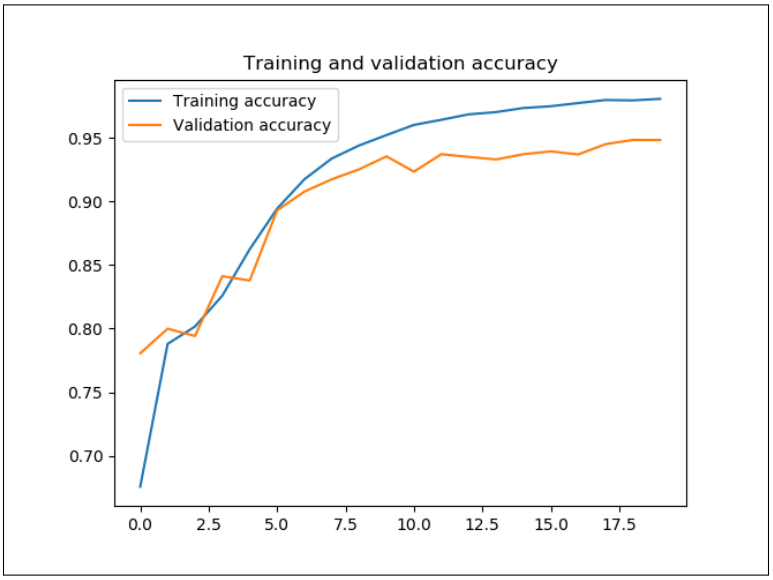

Figure 3: Plot of training and validation accuracy for 20 epochs

As we can see from figure 2 and figure 3, the training and validation are approximately similar with other, this means that there is no overfitting of data and that the model has learned the features well. These plots show that the test accuracies produced by this model are true.

Now, lets look at the heatmaps generated when the model is tested with the benchmark dataset and four other datasets.

1. Benchmark dataset:

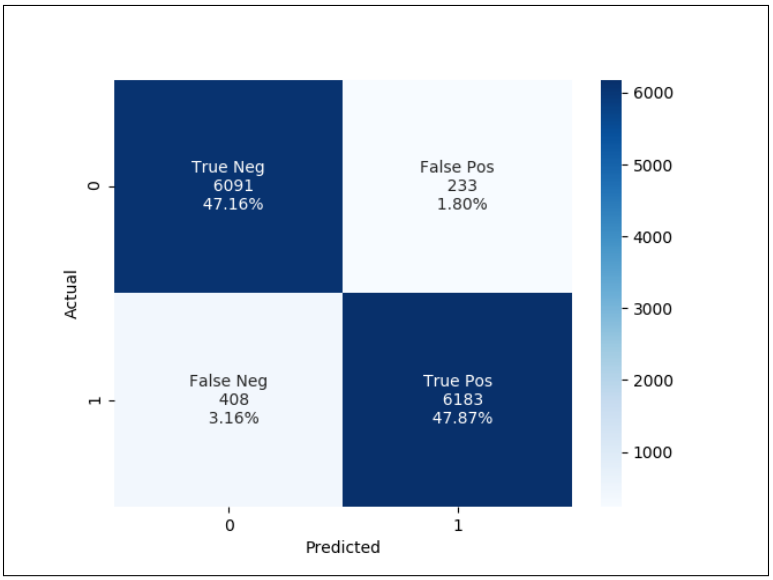

Figure 4: Heatmap of Benchmark dataset

From figure 4, we can say that the test prediction is highly accurate as the true positives and true negatives sum upto almost 96% which is great.

2. *C.elegans*:

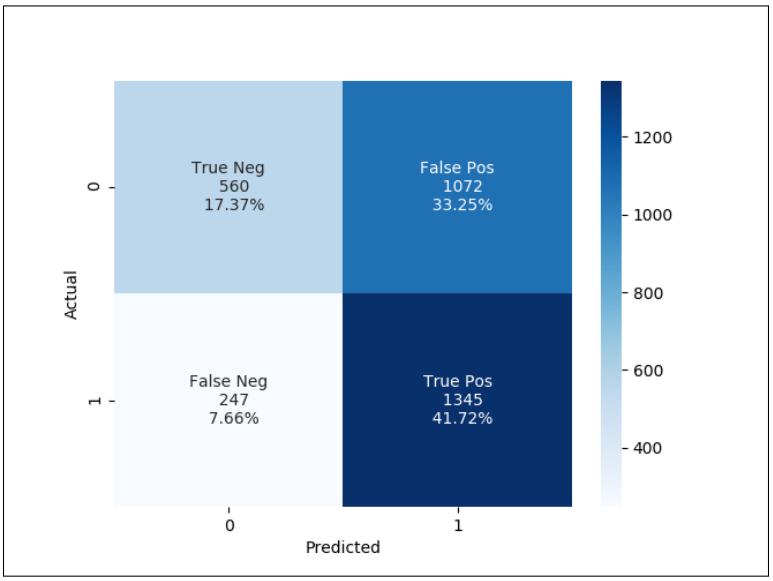

Figure 5: Heatmap of *C.elegans* dataset

3. *Drosophila*:

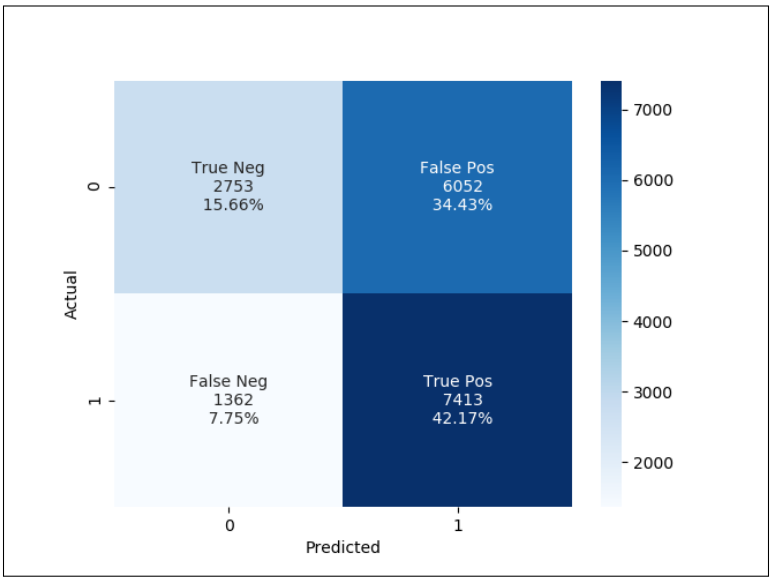

Figure 6: Heatmap of *Drosophila* dataset

4. *E.coli*:

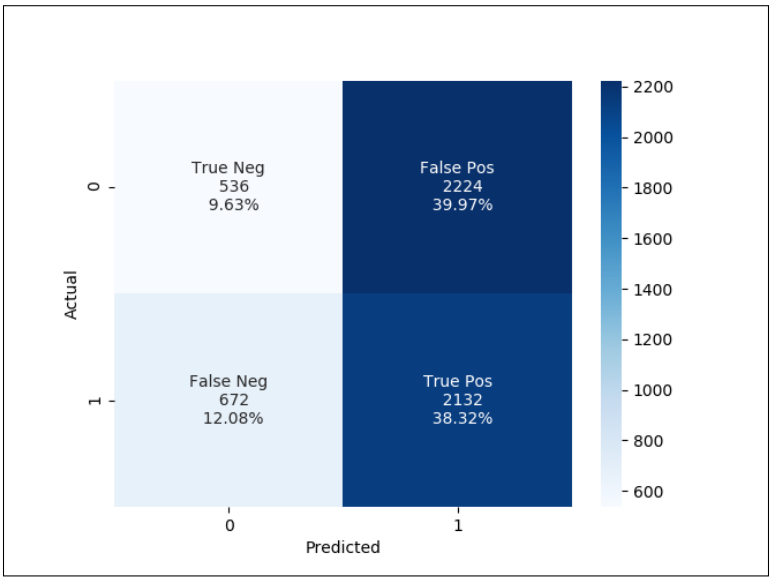

Figure 7: Heatmap of *E.coli* dataset

5. *H.sapiens*:

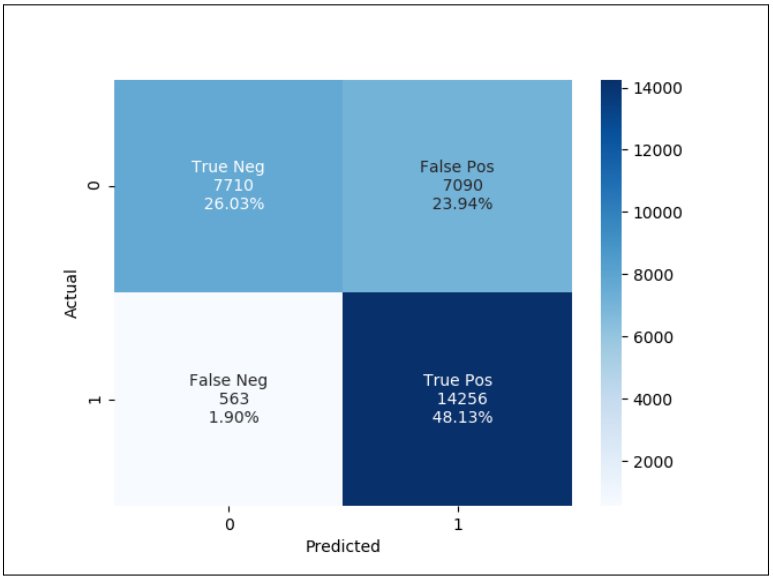

Figure 8: Heatmap of *H.sapiens* dataset

The following is the plot of summary of test accuracies for all the five datasets with which the model is tested.

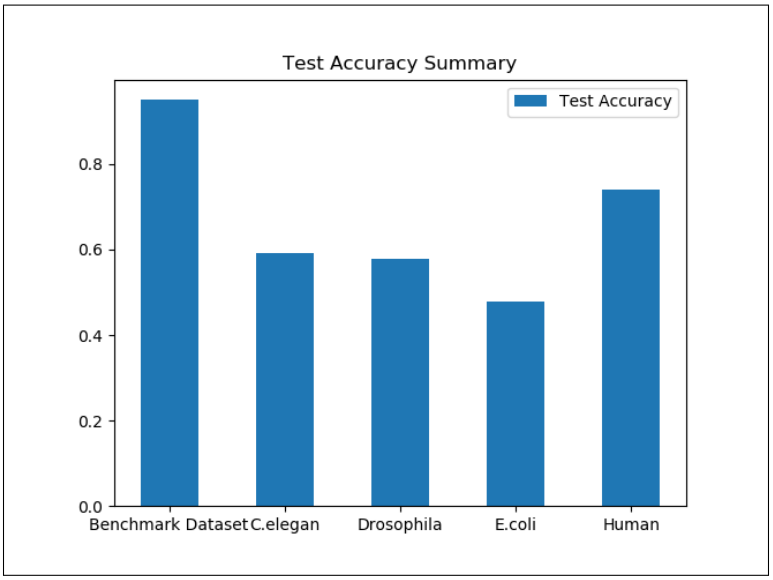

Figure 9: Test Accuracy Summary

From this figure we can say that the test accuracy of benchmark data is very accurate compared to others. This is because the model is trained on this dataset and it is comparatively performing well for it's data. Nonetheless, the test accuracies of other four datasets are also good enough to predict accurately for about 50%.

Let's now evaluate the model in terms of Precision, recall and F_1 Score. These 3 metrics are the macro-averages which means, each metric is computed independently for

each class(in our case 2 classes) and then their average is calculated. The following table gives the summary of these metrics calculated for all the 5 types of datasets.

Metric / Model	Precision	Recall	F1 score
Benchmark Dataset	0.95	0.95	0.95
C.elegans	0.63	0.59	0.57
Drosophila	0.61	0.58	0.55
E.coli	0.47	0.48	0.43
H.sapiens	0.80	0.74	0.73

Table 1: Summary of evaluation metrics

9.2 Auto Covariance Method:

Following are the plots generated using the Auto Covariance Method as the coding for the sequences. The test accuracies and the other evaluation metrics such as precision, recall and F1 score are similar to the ones generated for the Conjoint Triad Method.

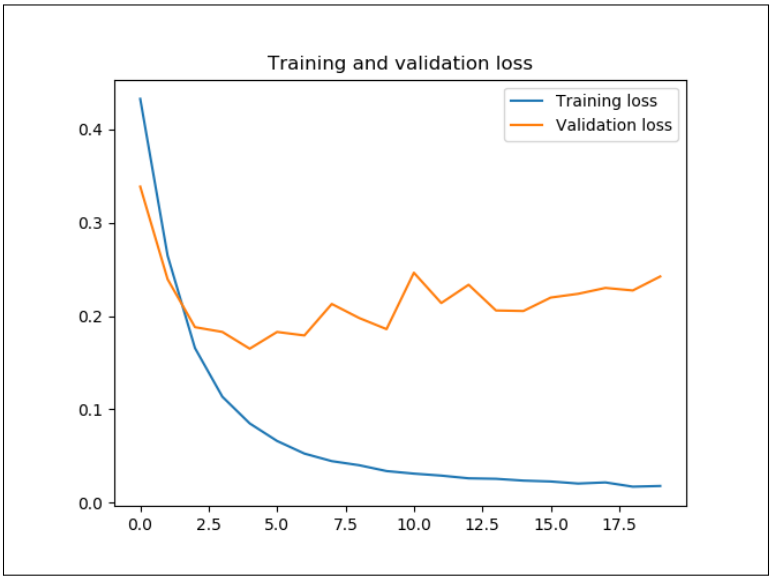

Figure 10: Plot of training and validation loss for 20 epochs

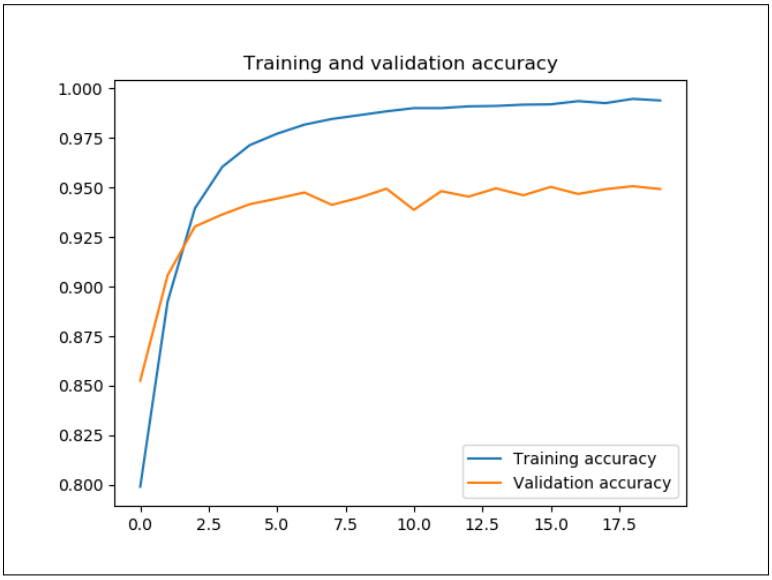

Figure 11: Plot of training and validation accuracy for 20 epochs

As we can see from figure 10 and figure 11, even this method doesn't lead to overfitting of data therby giving good predictions.

Following are the heatmaps generated when the model is tested with the benchmark dataset and four other datasets.

1. Benchmark dataset:

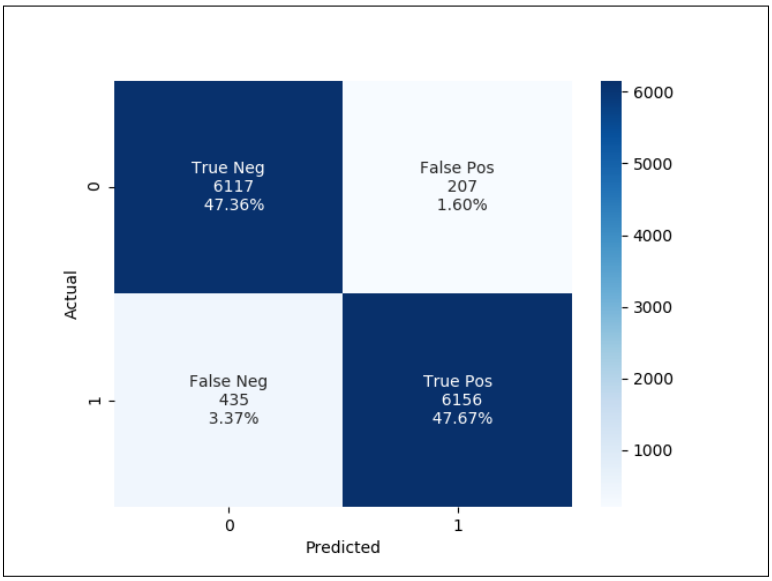

Figure 12: Heatmap of Benchmark dataset

From figure 12, we can say that the test prediction for AC method is also high and is around 97% which is great.

2. *C.elegans*:

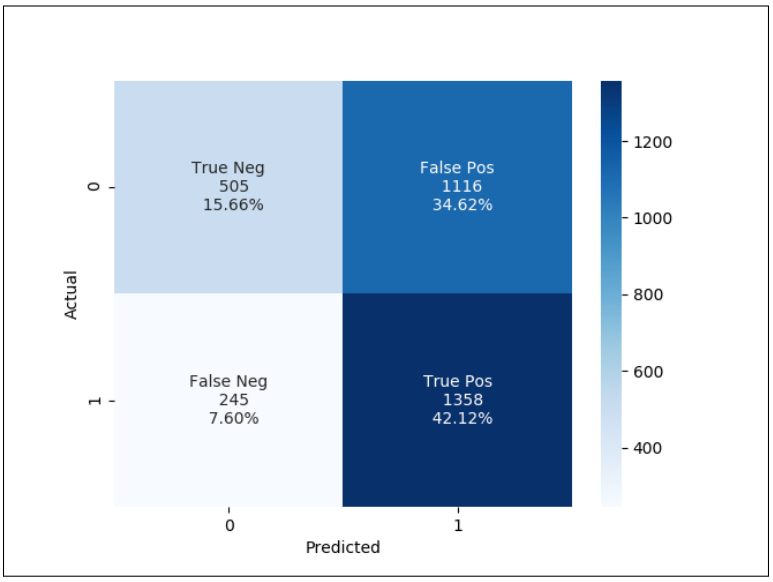

Figure 13: Heatmap of *C.elegans* dataset

3. *Drosophila*:

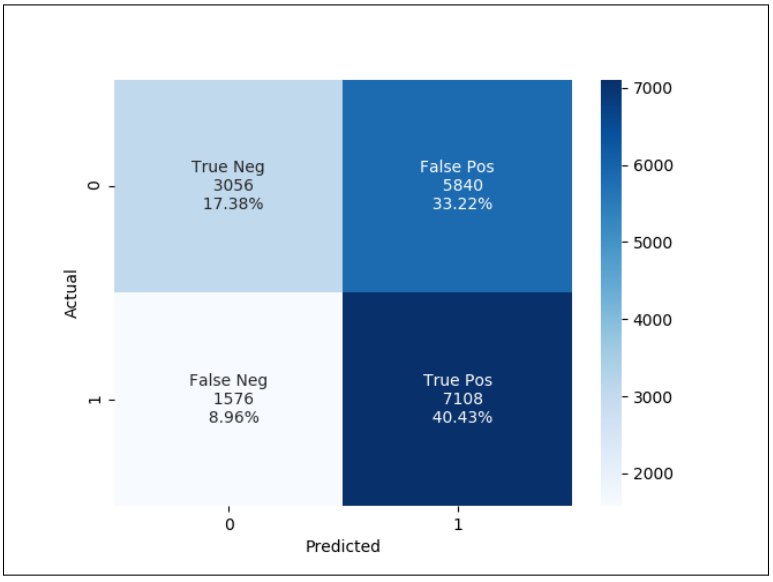

Figure 14: Heatmap of *Drosophila* dataset

4. *E.coli*:

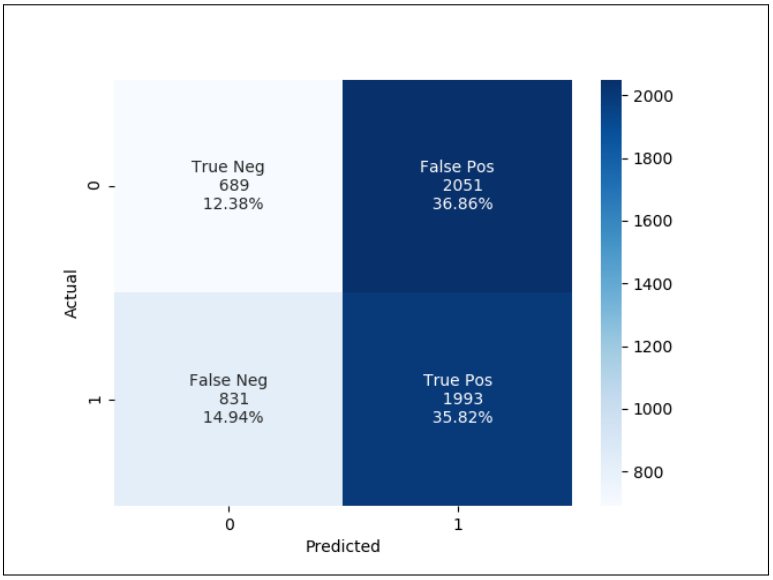

Figure 15: Heatmap of *E.coli* dataset

5. *H.sapiens*:

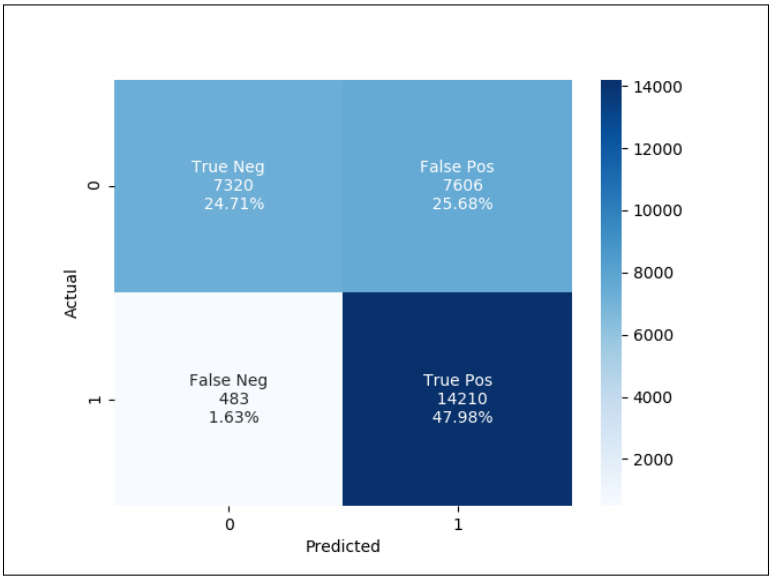

Figure 16: Heatmap of *H.sapiens* dataset

The following figure is the display of test accuracy summary for all the five datasets with which the model is tested.

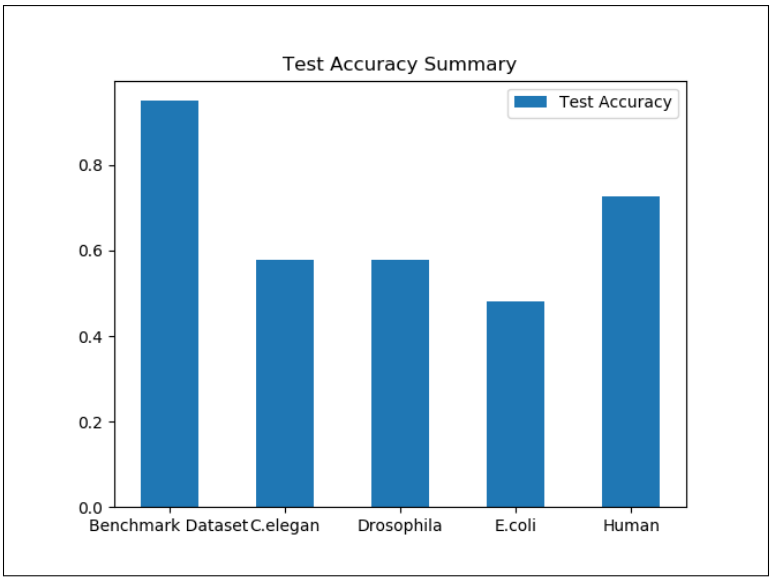

Figure 17: Test Accuracy Summary

Even the Auto Covariance Method gives almost same results as CT method and the prediction on Benchmark dataset is better than four other species.

Following table is the summary of the evaluation metrics calculated by testing the model with the five types of datasets. The metrics are in their macro-average form.

Metric / **Model**	**Precision**	**Recall**	**F1 score**
Benchmark Dataset	0.95	0.95	0.95
C.elegans	0.61	0.58	0.55
Drosophila	0.60	0.58	0.55
E.eoli	0.47	0.48	0.45
H.sapiens	0.79	0.73	0.71

Table 2: Summary of evaluation metrics

On comparing the results of both Conjoint Triad Method and Auto Covariance method, the test results were good in both the methods on benchmark dataset compared to other four datasets. Even though other datasets did not match with the accuracies of Benchmark, but 50% prediction accuracy is quite good.

10 Limitations

This method is limited on how well the protein sequence is represented. This is because, I have used a deep learning model to get trained and predict new interactions, and a deep learning model is only as good as the data on which it is trained. So, the training data

has to be good which is a coded representation of protein. So, the coding has to capture as many properties of protein as possible.

11 Future Scope

This problem can be further improved by using different or better coding techniques than the ones used in the project. Also, other classification models can be used on the same data used in this project and con do a comparative analysis to find out, if there is any better model which can give greater accuracy than the one used in this project.

References

[1] [Online]. Available: https://github.com/smalltalkman/hppi-tensorflow 2

[2] [Online]. Available: https://github.com/SuhithaG/PPI-prediction-from-sequence-using-Autoencoders 2

[3] T. Sun, B. Zhou, L. Lai, and J. Pei, "Sequence-based prediction of protein protein interaction using a deep-learning algorithm," *BMC Bioinformatics*, vol. 18, no. 1, p. 277, May 2017. [Online]. Available: https://doi.org/10.1186/s12859-017-1700-2 2

[4] X. Wang, R. Wang, Y. Wei, and Y. Gui, "A novel conjoint triad auto covariance (ctac) coding method for predicting protein-protein interaction based on amino acid sequence," *Mathematical Biosciences*, vol. 313, pp. 41 – 47, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0025556418307168 3, 5

[5] C. Angermueller, T. Pärnamaa, L. Parts, and O. Stegle, "Deep learning for computational biology," *Molecular Systems Biology*, vol. 12, no. 7, p. 878, 2016. [Online]. Available: https://www.embopress.org/doi/abs/10.15252/msb.20156651 3

[6] L. C. Xue, D. Dobbs, A. M. Bonvin, and V. Honavar, "Computational prediction of protein interfaces: A review of data driven methods," *FEBS Letters*, vol. 589, no. 23, pp. 3516–3526, 2015. [Online]. Available: https://febs.onlinelibrary.wiley.com/doi/abs/10.1016/j.febslet.2015.10.003 3

[7] X.-Y. Pan, Y.-N. Zhang, and H.-B. Shen, "Large-scale prediction of human proteinprotein interactions from amino acid sequence based on latent topic features," *Journal of Proteome Research*, vol. 9, no. 10, pp. 4992–5001, 2010, pMID: 20698572. [Online]. Available: https://doi.org/10.1021/pr100618t 3

[8] [Online]. Available: http://cic.scu.edu.cn/bioinformatics/predict_ppi/default.html 3

[9] J. Shen, J. Zhang, X. Luo, W. Zhu, K. Yu, K. Chen, Y. Li, and H. Jiang, "Predicting protein–protein interactions based only on sequences information," *Proceedings of the National Academy of Sciences*, vol. 104, no. 11, pp. 4337–4341, 2007. [Online]. Available: https://www.pnas.org/content/104/11/4337 4

[10] Y. Guo, L. Yu, Z. Wen, and M. Li, "Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences," *Nucleic Acids Research*, vol. 36, no. 9, pp. 3025–3030, 04 2008. [Online]. Available: https://doi.org/10.1093/nar/gkn159 4, 5

[11] [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798 6