A Hybrid Method for Prediction of Protein Secondary Structure Based on Multiple Artificial Neural Networks

Haris Hasic, Emir Buza, Amila Akagic
Faculty of Electrical Engineering
Department for Computer Science and Informatics
University of Sarajevo, Bosnia and Herzegovina
Email: {haris.hasic, ebuza, aakagic}@etf.unsa.ba

***Abstract*—The prediction of protein secondary structure is the method of finding the way in which an amino acid sequence causes the protein structure to fold and bend into *alpha helices*, *beta strands* and other shapes. Until today, the problem of finding protein secondary structure is not fully resolved. Classification or clusterization based methods have an accuracy rate of circa 80 percent and they mainly work on a reduced set of shapes and folds. It is very difficult to predict how a local sequence of amino acids is going to behave and in which way it is going to affect the future of protein structure. Based upon the predicted secondary structure of the protein, the tertiary and quaternary predictions show the real nature and function of the protein as a whole. In this paper, we address the problem of the secondary structure prediction of protein and propose a new hybrid method based on the usage of multiple neural networks with the use of a consensus function and compare our approach with other efficient methods.**

***Keywords*—Bioinformatics, Protein Secondary Structure Prediction, Hybrid Method, Neural Networks, Machine Learning**

I. Introduction

Bioinformatics is an important interdisciplinary field in which information technologies are used to successfully solve existing biological problems in the world today. Usually, the most accurate ways to solve these kinds of problems are through experimental methods. Some of those approaches are described in [1]–[3]. The main obstacle, which undermines the significant progress of experimental methods, is the high cost of the entire process. One of the reasons for the high cost is the high amount of pure protein required to perform experiments on, not to mention the required amount of computational power. On the other hand, information technologies are rapidly improving and spreading over various fields, which opens up the space for new approaches in solving biology problems. These kinds of approaches are typically referred to as *ab initio* approaches since they usually focus on solving problems ”from scratch” rather than using existing structures obtained through experimental methods. According to [4], some of those methods can achieve up to 75% to 80% accurate results, while the theoretically highest possible accuracy lies at 90%.

The problem that has proven difficult to solve efficiently is the prediction of protein structures. Predicting the protein structure and function does not solely include predicting classes and structures, but also predicting the environmental and other potential influences, protein-protein interactions etc. The problem is rather complicated to be precisely determined only with machine learning and other mathematical-based methods. However, a combination of different algorithms merged together and combined with some biology field knowledge might just give good results. This fusion of different experimental and mathematical approaches represents the domain of hybrid algorithms and it is proven to be an efficient way to improve existing algorithms.

In this paper, we focus on the improvement of the machine learning algorithms which are typically referred to as *in silico* methods, especially neural network approaches. The accuracy of neural network based methods is around 60% [5] and the results largely depend on the protein that is being analyzed. We try to increase this accuracy and also overcome the large oscillations that can occur if the input datasets which contain a large number of differently structured proteins. Our focus is the identification and exact classification of the two most common secondary structural classes, *alpha helices* and *beta sheets* (Fig.1). Other structures that can form during the process are aggregated in the *coil* class. We present a new hybrid method based on multiple neural networks combined together through a census function. The networks that compose the ensemble are trained with different parameters which are determined empirically, through analysis of benchmarking results. The local results obtained from the neural networks are analyzed and, through a majority voting process, combined into a global ensemble result. This ensures the higher consistency and accuracy of the method in comparison to the single network approach, regardless of the diversity of the input dataset. The accuracy of our method is 65% for all the datasets used.

This paper is organized as follows: Related work regarding different machine learning approaches is briefly reviewed in section II. In section III a new, neural network based, hybrid method for protein secondary structure prediction is proposed. The implementation of the proposed method and discussion of the achieved results are stated in section IV. We conclude the paper in section V with appropriate remarks.

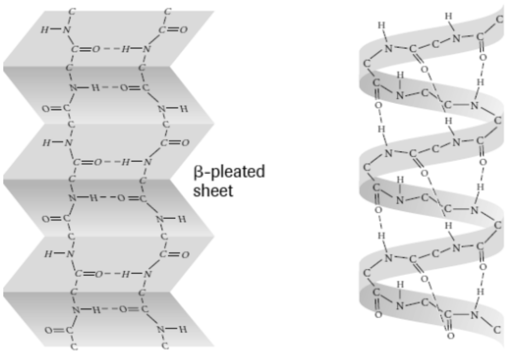

Fig. 1: Two of the most common protein secondary structure elements: *alpha helices* and *beta sheets*.

II. Related Work

In the early development stages of secondary structure prediction methods, amino acids had been mostly observed statistically, one residue at a time. Those methods were also constrained by the small amount of predetermined protein structures available at the time. As more secondary structures were acquired through experimental methods, the *in silico* methods also advanced in consistency and accuracy as they had much more example data to work with. One of the first fairly consistent methods for secondary structure prediction was the *Chou-Fasman* method [6] which combined different statistical and heuristic rules. The main problem with this method was the mentioned inspection of isolated amino acids in the chain which couldn't exactly reflect the real state of the protein as a whole. This issue was resolved in the *GOR* [7] [8] method where the surrounding of the amino acids was also included in the secondary structure prediction.

After the initial simple approaches, the algorithms began to improve drastically as higher degree interactions between elements were observed. More advanced statistical approaches were implemented and elements of machine learning were integrated in the methods. Advanced methods make use of nearest neighbor approaches and fuzzy logic [9]–[11], hidden Markov models [12] [13], support vector machines [14] [15], neural networks etc. Today, approaches that only predict protein structure from a single organism are getting more popular since they avoid the need for generalization and therefore offer higher accuracy in prediction. One example is the specified structure protein interaction for the yeast species *Saccharomyces Cerevisiae* is described in [16].

One of the more interesting approaches is the usage of neural networks. The network is trained with a primary structure and the corresponding secondary structure later predicts secondary structural classes for protein with unknown secondary structure. There are many different ways to tackle the defined problem using different types of neural networks, sometimes in a combination with other algorithms as described in [17]–[20]. The deep learning algorithms are also gaining popularity as described in [21]–[24]. They emphasize the learning process of the networks and achieve more accurate results.

III. New Hybrid Method

The problem of the secondary structure prediction is one of the most challenging problems in bioinformatics. The neural networks ”style” of problem solving is a one way of solving this problem. We formalize the main steps of the modeling process for our method as:

(A) Window length selection
(B) Binarization of inputs and outputs
(C) Construction of the neural network as a classifier
(D) Ensemble construction

The first two steps offer a detailed description of the data preparation process with focus on the inclusion of the immediate surroundings of each residue. After the input and output data format is established, available datasets are preprocessed and divided into appropriate training, validation and test sets. The main contribution of our method is in the next two steps, where we introduce network selection and ensemble construction. Based on achieved results for different parameters, the best performing networks are chosen and a diversified ensemble is constructed. The voting process which unifies single neural network results is implemented. In the end, a series of tests containing data from different datasets than the ones used in the training process are carried out to measure performance of the algorithm. The next subsections offer detailed description of all individual steps of the process.

A. Window Length Selection

Looking at a single amino acids individually does not get good results. The interaction between residues in the chain needs to be preserved in some way. For example, if the window size is 11 and an amino acid at the n^{th} position in the chain is in focus, elements at positions $\{n\text{-}5, ..., n, ..., n\text{+}5\}$ also need to be taken into consideration as depicted in Fig. 2. In [25], it is shown that there are many factors in successfully determining the optimal window size, but it largely depends on the protein in focus. Many protein secondary structures depend on factors such as hydrophobicity, motifs, b-factors etc. and that makes it difficult to find the general optimal sliding window size for all the protein in existence. For example, the transmembrane proteins have the average of one transmembrane *alpha helix* spanning through the membrane so the optimal size should be around 20 residues. The only way to determine the optimal window size is empirical and thus multiple neural networks with different windows must be constructed, trained and evaluated.

... Ala Arg Asn Asp Cys Gln Glu Gly His Ile Met Phe Phe ...

Fig. 2: Sliding window size.

B. Binarization of Inputs and Outputs

The string representation of the amino acid chain is simple for people to understand, but difficult for machines to process. Because of that, we need a fitting transformation of the input in order to model a neural network and gain efficiency in terms of processing speed. Generally speaking, one of the easier types of data for machines to process is the binary format. If the amino acid chain is composed from a total of 20 different amino acid types, the matching binary form will contain 20 positions, where only one position is set to 1 and the other to 0 to represent the type of amino acid as annotated in Table I.

TABLE I: Binary codes for each of the 20 amino acids.

Ala	[10000 .. 0 .. 0 .. 0]	***Met***	[0 .. 0 .. 10000 .. 0]
Arg	[01000 .. 0 .. 0 .. 0]	***Phe***	[0 .. 0 .. 01000 .. 0]
Asn	[00100 .. 0 .. 0 .. 0]	***Pro***	[0 .. 0 .. 00100 .. 0]
Asp	[00010 .. 0 .. 0 .. 0]	***Ser***	[0 .. 0 .. 00010 .. 0]
Cys	[00001 .. 0 .. 0 .. 0]	***Thr***	[0 .. 0 .. 00001 .. 0]
Gln	[0 .. 10000 .. 0 .. 0]	***Trp***	[0 .. 0 .. 0 .. 10000]
Glu	[0 .. 01000 .. 0 .. 0]	***Tyr***	[0 .. 0 .. 0 .. 01000]
Gly	[0 .. 00100 .. 0 .. 0]	***Val***	[0 .. 0 .. 0 .. 00100]
His	[0 .. 00010 .. 0 .. 0]	***Asx***	[0 .. 0 .. 0 .. 00010]
Ile	[0 .. 00001 .. 0 .. 0]	***Glx***	[0 .. 0 .. 0 .. 00001]

This indicates that the input needs to be at least a 20x*m* matrix where *m* is the number of amino acids in the chain. Because of the sliding window size that needs to be incorporated into the input, the matrix also needs to have *n* residues on each side of the current amino acids. Therefore, the final input matrix needs to be a (20**ws*)x*m* matrix where *ws* represents the sliding window size. For example, if we use a window with a size of 17, the final input matrix will be 340x(*n* – 16). The 16 elements that are subtracted are the first and last 8 elements of the amino acid sequence, that are ignored because the window size can be applied only at the 9^{th} position. That leads to the conclusion that some of the elements of the primary structure will not be included in the prediction of the secondary structure which is one of the disadvantages of this surrounding-inclusive approach. The input data has grown in dimensionality since it went from a simple string to a fairly big matrix but in this format it is much easier for the computers to process and also an excellent fit for the input of a neural network.

The same principle of binarization can also be applied for the output. The main difference is that there are only three classes to represent. The *alpha helix* labeled A, the *beta sheet* labeled B and the *coil* labeled C.

$$bout(c) = \begin{cases} [1 \quad 0 \quad 0]^T & \text{if } c \text{ is } \textit{alpha helix} \\ [0 \quad 1 \quad 0]^T & \text{if } c \text{ is } \textit{beta strand} \\ [0 \quad 0 \quad 1]^T & \text{if } c \text{ is } \textit{coil} \end{cases}$$

The *c* represents the resulting structural class, and *bout(c)* represents the output binarization function which translates the three structural classes from the string into the binary format.

C. Construction of the Neural Network

Multilayer feed-forward neural networks with backpropagation learning algorithm are suitable for advanced classification problems [26] as the one that is being treated in this paper.

1) Neural Network Architecture: In a feed-forward network the output of a node *y* is described as a function of the input *x*. The input to a given node is a sum of previous nodes and their associated weights:

$$X = \sum_{i=1}^{n} x_i w_i \tag{1}$$

where *n* is the number of neurons and w_i is the associated weight. This value is then passed through a sigmoid activation function:

$$Y^{sigmoid} = \frac{1}{1 + e^{-X}} \tag{2}$$

which guarantees that the neuron output is bounded between 0 and 1. If we consider equation (1), the activation of the nodes y_i can be defined as:

$$y_i = f_i(X) = f_i(\sum_{i=1}^{n} x_i w_i) \tag{3}$$

For any dataset given as input and the corresponding weights, there is a certain error measured by an error function. Since the backpropagation learning rule is applied, there are two contrary directions of information flowing across the network. Input signals $(x_1, x_2, ..., x_n)$ are propagated from the left to right and the error signals $(e_1, e_2, ..., e_n)$ from right to left. Error signals are calculated for the output of each neuron and the general error function for one epoch is defined as the sum of the squares of the differences between all target node outputs and actual node outputs:

$$E_p = \frac{1}{2} \sum_{n} (t_{jn} - w_{jn})^2 \tag{4}$$

where t_{jn} is the target activation value for the node *n* and *p* marks the current epoch. Given the equation (4), the networks overall error is simply calculated by summing all of the E_p values for a given set of training patterns. The respective formula and the standardized version, the MSE (abbr. *Mean Squared Error*) equation are shown below.

$$E = \sum_{p} E_p = \sum_{p} \sum_{n} (t_{jn} - w_{jn})^2 \tag{5}$$

$$MSE = \frac{1}{2PN} \sum_{p} \sum_{n} (t_{jn} - w_{jn})^2 \tag{6}$$

The MSE shows the difference between the correct output and what's estimated. Since the algorithm uses the backpropagation learning rule, which is based on the *Widrow-Hoff delta learning* rule, the main goal is to adjust the neural network parameters in a way that the MSE is minimized below a certain threshold. As that is not always bound to happen, an additional maximum epoch number is given after which the algorithm terminates regardless of the current MSE value.

2) Important Parameters of the Neural Network: One of the important factors in this algorithm is the configuration of the parameters that suits the secondary structure prediction problem. Three neurons make up the initial hidden layer. By increasing this number, more accurate results can be achieved, but there is also the risk of *overfitting*. The neural network can get too adapted to the training data and try to memorize the previous examples instead of learning how to generalize and adapt its structure to successfully solve unknown structures that show up as the input after training ends. All the different layers and the previously described backpropagation learning flow of data inside a neural network is depicted in Fig 3.

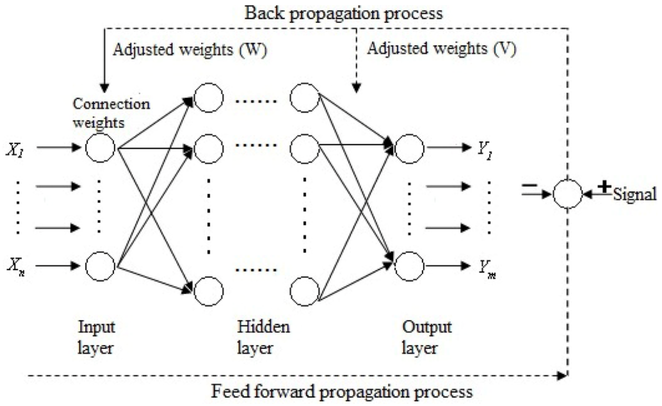

Fig. 3: Neural network architecture and data flow.

For the training of the networks in the ensemble the Rost-Sander RS121 [27] and the FC699 datasets [28] are used, since they contain a wide range of different protein primary structures and their corresponding secondary structures. This makes the selection of datasets justifiable for the initial training. Through the *k-fold cross-validation* method of determining training, validation and test sets, the individual datasets are split and translated into the correct input matrix format with a split ratio of 70-15-15 for all the sets, respectively. Other parameters that have a significant impact on the networks performance are also: the training algorithm, number of neurons in the hidden layer, number of input vectors etc.

D. Construction of the Ensemble

The idea of creating a learning ensemble is relatively simple. Since the neural network largely depends on the quality and diversity of the input data, it cannot provide the correct prediction all the time. This is especially true within the structure prediction problem since a wide range of protein families exists. That is why multiple networks are created and their results are combined through one of the consensus methods. Individual networks can have different architectures, different number of neurons in their layers, different window sizes etc. The important aspect is that the output is one of the structural classes. In this way, if H_n is the hypothesis space of one of the ensemble members, multiple hypothesis spaces narrow down the possible solution space with their intersections as depicted in Fig. 4. This limits the search space for the optimal solution marked as h_{best}. Thus, the best classification rule is constructed through approximation of multiple classification rules. One important factor is the aggregation of multiple outputs. There are many different approaches, however in this paper we used majority voting method to determine the final output. If one or two networks fail to correctly classify a structural class, other networks with correct predictions can override the bad result. It all depends on the way other networks are constructed and trained. Therefore, the entire ensemble will give incorrect results only if the majority of the networks fails to identify the correct structural class. The ensemble approach in our method can add some additional security to the classification that is sometimes needed to provide satisfactory results.

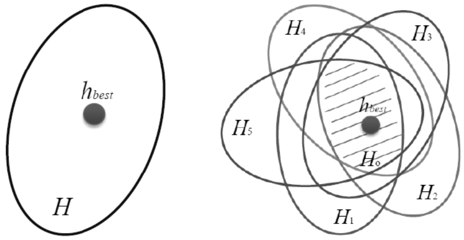

Fig. 4: Multiple hypothesis space intersection.

The complete pseudo code for our hybrid method, based on multiple neural networks working together within an ensemble, can be formulated as in Algorithm 1. The method requires certain parameters which are usually determined empirically as described in the previous sections. The first two lines of the for loop represent on of the advantages of the method as pre-processed datasets with diversified data are given as inputs to neural networks to ensure the stability of the ensemble.

Algorithm 1 Hybrid Method based on an Ensemble of Multiple Artificial Neural Networks

```
function PREDICTPROTEINSECONDARYSTRUCTURE
    Require:
    size - Size of the ensemble;
    ws - Sliding window size;
    datasets[] - Datasets used for training;
    annParameters[] - Network parameters;
    inputSequence - Amino acid chain;
    k - k-fold cross-validation parameter;

    for $i < size; i \leftarrow i + 1$ do
        binIn = binarizeInputs(datasets[n], ws);
        binOut = binarizeOutputs(datasets[n]);
        dataset = combine(binIn, binOut);
        [tr,te,val] = crossValidation(dataset, k);
        ann = constructANN([tr,te,val], annParameters[i]);
        results[i] = ann.predict(inputSequence);
    end for

    secondaryStructure = consenusMethod(results);
    return secondaryStructure;
```

IV. Results

The algorithm described in previous chapters was implemented in MATLAB version 8.5.0 (R2015a) using the Neural Network Toolbox. The datasets used for training were the RS121 and FC699. The dataset used for testing was a combined dataset containing parts from the 25PDB [28] and CB513 [29] datasets and other proteins and their respective secondary structures which did not occur in the training sets. The standard *Q3* or *Average Percentage Accuracy* method was used as the quality measurement of the results. The results that were achieved for a single neural network with different parameter combinations are listed in Table II. Throughout all runs, the highest percentage achieved is at about 61% with 10 neurons in the hidden layer and a window size of 17. It is also visible that, with these configuration parameters, the network has big oscillations, since the accuracy for window size of 3 lies little below 35%. That makes these results not trustworthy.

TABLE II: Q3 accuracy results for a single neural network.

Train	10	15	20	[3 5 3]	[5 10 5]	[10 20 10]
3	35,0%	57,1%	54,3%	55,3%	53,4%	53,3%
5	58,4%	60,1%	59,7%	56,3%	54,9%	59,0%
9	60,2%	61,6%	62,3%	58,3%	52,8%	62,0%
17	61,6%	65,1%	64,1%	47,1%	61,4%	58,9%
21	64,8%	58,5%	65,6%	46,6%	46,7%	59,0%
Test	**10**	**15**	**20**	**[3 5 3]**	**[5 10 5]**	**[10 20 10]**
3	34,8%	55,8%	52,8%	52,9%	52,3%	54,5%
5	56,9%	58,8%	59,2%	53,9%	52,6%	55,6%
9	57,1%	59,5%	60,1%	55,4%	55,0%	60,2%
17	61,0%	59,8%	60,3%	47,9%	56,3%	59,1%
21	59,8%	57,1%	59,1%	46,8%	48,9%	56,2%

According to the measurements, the best and most consistent results are achieved with a window size of 17 and 20 neurons in the hidden layer. If Table II is translated into a percentage bar chart for the most successful parameter configurations, the correlation between the window size and the prediction accuracy becomes visible. That leads to the conclusion that the optimal empirical values for the window size are between 17 and 20, depending on the protein structure.

These results show that the isolated neural network performance possibilities lie at around 60% as shown in Fig. 5. Of course, these numbers can be increased by implementing some advanced network improvements as mentioned in Section II, but in this paper we focus is on the ensemble and integration of methods. The described ensemble method is tested by executing 20 runs with 20 different test sets than the ones used to train the neural networks. The results achieved through the whole process of testing the proposed method are as shown in Table III. The highest and lowest accuracy is also marked.

The average accuracy lies at approximately 65,3% which shows the improvement made by simply combining differentiated neural networks together. If the datasets that caused the two best, two worst and a near-average performance in the ensemble are given as input in a single neural network and *Naive Bayes* classificator, a good accuracy comparison can be made. As depicted in Fig. 6, the proposed hybrid method solves the structural classification much more efficiently than the single neural network approaches and the common classification methods such as the *Naive Bayes* classificator [30]. That is to be expected because of the lack of diversity in network training and the previously described problems with purely statistical approaches cannot cover all the processes within the secondary structure formation process.

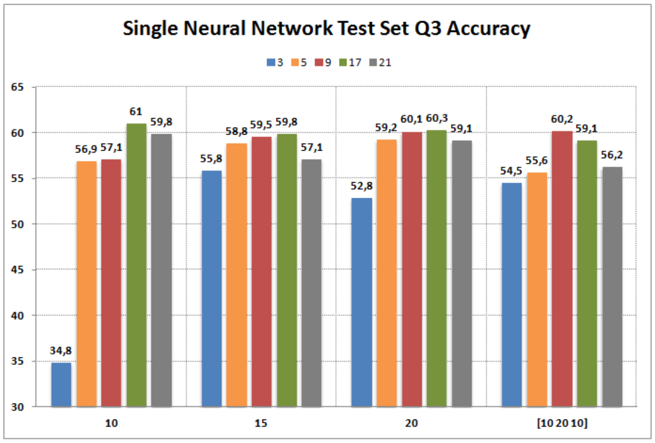

Fig. 5: Neural network Q3 accuracy bar chart.

TABLE III: Q3 accuracy results for a network ensemble.

	1	2	3	4	5
Q3 Accuracy	64,5%	68,9%	64,9%	64,8%	65,4%
	6	**7**	**8**	**9**	**10**
Q3 Accuracy	68,7%	65,2%	65,0%	65,5%	64,2%
	11	**12**	**13**	**14**	**15**
Q3 Accuracy	64,2%	64,0%	66,1%	64,4%	64,3%
	16	**17**	**18**	**19**	**20**
Q3 Accuracy	65,1%	65,0%	64,6%	66,5%	64,9%

It is also worth noting that the isolated neural networks work well under the additional pressure of differentiating datasets. That means that, for all the individual neural networks, a good parameter configuration is chosen and that the networks are capable of good generalization. The common problems that can arise, such as *overfitting* and *underfitting*, are thereby successfully avoided.

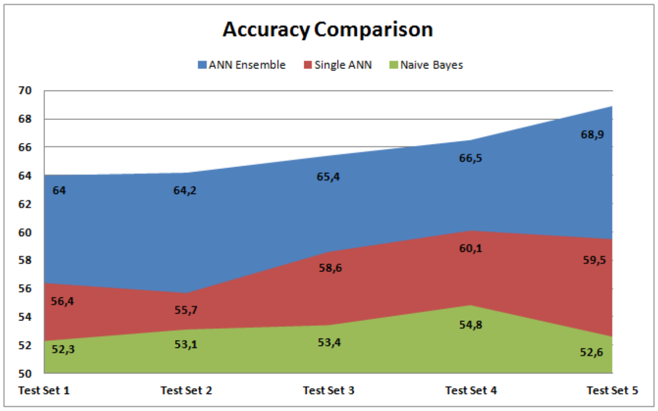

Fig. 6: Accuracy comparison for different approaches.

V. Conclusion

The search for a universal algorithm for the protein secondary structure prediction is not an easy problem to solve. However, in this paper, a hybrid, multiple neural network ensemble approach is proposed which shows promising results of improving the accuracy of existing algorithms. Through a simple aggregation of different prediction methods, this approach narrows down the possible hypothesis space in which the optimal solution is located and therefore increases the time that is needed to find the optimal solution. That also makes it more likely that the optimal solution will be found, i.e. that the MSE parameter will drop down below the given accuracy threshold within the set number of epochs.

The proposed method, based on multiple differently trained neural networks achieves around 65% accuracy of successfully predicted secondary structures. The final evaluation was based upon creations of different smaller datasets partially derived from the 25PDB and CB513 datasets and other protein structures gathered for the purpose of testing. The input data was formed by combining data from different sources, which proves that our method, along with the accuracy increase, is stable in prediction of diverse protein structures. Since the accuracy of methods based solely on neural networks lies around 60% [5], and those methods can have oscillations for differently structured protein than those used for training, our proposed method is suitable for classification of protein secondary structures. Also, it offers a good example on how to combine different methods and, more importantly, how to properly train and incorporate these elements into a bigger, more advanced algorithm for secondary structure prediction.

References

[1] M. Pukáncsik, Á. Orbán, K. Nagy, K. Matsuo, K. Gekko, D. Maurin, et al. (2016). *Secondary Structure Prediction of Protein Constructs Using Random Incremental Truncation and Vacuum-Ultraviolet CD Spectroscopy*. PLoS ONE 11(6): e0156238. doi:10.1371/journal.pone.0156238

[2] L. Whitmore, B.A. Wallace (2008). *Protein Secondary Structure Analyses from Circular Dichroism Spectroscopy: Methods and Reference Databases*. Biopolymers 89: 392–400. PMID: 17896349

[3] K. Matsuo, K. Sakai, Y. Matsushima, T. Fukuyama, K. Gekko (2003). *Optical Cell With a Temperature-Control Unit for a Vacuum-Ultraviolet Circular Dichroism Spectrophotometer*. Analytical Sciences 19: 129–132. PMID: 12558036.

[4] O. Dor, Y. Zhou; Zhou (2006). *Achieving 80% Tenfold Cross-validated Accuracy for Secondary Structure Prediction by Large-scale Training*. Proteins. 66 (4): 838–45. doi:10.1002/prot.21298. PMID 17177203.

[5] J. Chandonia, M. Karplus (1994). *Neural Networks for Secondary Structure and Structural Class Predictions*. Protein Science (1995), 4: 275-285.

[6] P. Privilege Jr., G.D. Fasman (1989). *Chou-Fasman Prediction of the Secondary Structure of Proteins*. MIT.

[7] J. Garnier, B. Robson (1989). *Prediciton of Protein Structure and the Principles od Protein Conformation*. Plenum Press, New York.

[8] J. Garnier, J.F. Gibrat, B. Robson (1996). *GOR Method for Predicting Protein Secondary Structure from Amino Acid Sequence*.

[9] J. Sim, S.Y. Kim2, J. Lee (2005). *Prediction of Protein Solvent Accessibility using Fuzzy k-nearest Neighbor Method*. Vol. 21 no. 12 2005, pages 2844–2849. doi:10.1093/bioinformatics/bti423

[10] E.G. Mansoori, M.J. Zolghadri, S.D. Katebi (2009). *Protein Superfamily Classification Using Fuzzy Rule-Based Classifier*. IEEE Transactions on Nanobioscience, Vol. 8, No. 1.

[11] H. Shen, J. Yang, X. Liu, K. Chou (2005). *Using Supervised Fuzzy Clustering to Predict Protein Structural Classes*. Biochemical and Biophysical Research Communications 334 (2005) 577–581.

[12] J. Martin, J.F. Gibrat, F. Rodolphe (2005). *Choosing the Optimal Hidden Markov Model for Secondary-Structure Prediction*. IEEE Intelligent Systems, vol. 20, no. , pp. 19-25, November/December 2005, doi:10.1109/MIS.2005.102.

[13] N. Nguyen, M. Nute, S. Mirarab, T. Warnow (2016). *HIPPI - Highly Accurate Protein Family Classification with Wnsembles of HMMs*. 14th Annual Research in Computational Molecular Biology (RECOMB) Comparative Genomics Satellite Workshop, Montreal, Canada. BMC Genomics 2016, 17(Suppl 10):765. doic 10.1186/s12864-016-3097-0.

[14] Y. Dong, X. Liu, G. Zhou (2001). *Support Vector Machines for Predicting Protein Structural Class*.doi:10.1186/1471-2105-2-3.

[15] B. Bhushan, M.K. Singh (). *Protein Structure Prediction using Neural Networks and Support Vector Machines*. International Journal of Engineering Science and Advanced Technology, Volume-3, Issue-3, 145-156.

[16] J. Zubek et al. (2015). *Multi-level Machine Learning Prediction of Protein–Protein Interactions in Saccharomyces Cerevisiae*. PeerJ 3:e1041; DOI 10.7717/peerj.1041.

[17] Z. Sun, X. Rao, L. Peng, D. Xu (1997). *Prediction of Protein Supersecondary Structures Based on the Artificial Neural Network Method*. vol.10 no.7 pp.763–769, 1997.

[18] K. Lin, V.A. Simossis, W.R. Taylor, J. Heringa (2005). *A Simple and Fast Secondary Structure Prediction Method Using Hidden Neural Networks*. Vol. 21 no. 2 2005, pages 152–159. doi:10.1093/bioinformatics/bth487.

[19] Z. Li, Y. Yu (2016). *Protein Secondary Structure Prediction Using Cascaded Convolutional and Recurrent Neural Networks*. International Joint Conferences on Artificial Intelligence. arXiv:1604.07176v1[q-bio.BM].

[20] C. Mirabello, A. Adelfio, G. Pollastri (2014). *Reconstructing Protein Structures by Neural Network Pairwise Interaction Fields and Iterative Decoy Set Construction*. Biomolecules 2014, 4, 160-180; doi:10.3390/biom4010160.

[21] K. Paliwal, J. Lyons, R. Heffernan (2015). *A Short Review of Deep Learning Neural Networks in Protein Structure Prediction Problems*. Adv Tech Biol Med 3:139. doi: 10.4172/2379-1764.1000139.

[22] S. Wang et al. (2016). *Protein Secondary Structure Prediction Using Deep Convolutional Neural Fields*. Sci. Rep. 6, 18962; doi: 10.1038/srep18962 (2016).

[23] Z. Lin, J. Lanchantin, Y. Qi (2016). *MUST-CNN: A Multilayer Shift-and-Stitch Deep Convolutional Architecture for Sequence-based Protein Structure Prediction*. AAAI 2016. arXiv:1605.03004v1 [cs.LG].

[24] A. Busiay, J. Collins, N. Jaitly (2016). *Protein Secondary Structure Prediction Using Deep Multi-scale Convolutional Neural Networks and Next-Step Conditioning*. RECOMB 2017. arXiv:1611.01503v1 [cs.LG].

[25] K. Chen, L. Kurgan, J.Ruan (2006). *Optimization of the Sliding Window Size for Protein Structure Prediction*. CIBCB '06: 2006 IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology, pp.1-7, 28-29, doi:10.1109/CIBCB.2006.330959.

[26] G.P. Zhang (2000). *Neural Networks for Classification: A Survey*. IEEE Transactions on Systems, Man and Cybernetics-Part C: Applicationss and Reviews, vol. 30, No. 4, November 2000. Publisher Item Identifier S 1094-6977(00)11206-4.

[27] B. Rost, C. Sander, R. Schneider (1994). *Redefining the Goals of Protein Secondary Structure Prediction*. J. Mol. Biology. 235:13–26. [PubMed].

[28] L. Kurgan, K. Cios, K. Chen (2008). *SCPRED: Accurate Prediction of Protein Structural Class for Sequences of Twilight-zone Similarity with Predicting Sequences*. BMC Bioinformatics, 9:226.

[29] J. Cuff, G. Barton (1999). *Evaluation and Improvement of Multiple Sequence Methods for Protein Secondary Structure Prediction*. PROTEINS: Structure, Function, and Genetics 1999; 34508–519.519. [PubMed].

[30] Q .Li, D. Dahl, M. Vannucci, J. Hyun, J. Tsai (2014). *Bayesian Model of Protein Primary Sequence for Secondary Structure Prediction*. PLoS ONE 9(10): e109832. doi:10.1371/journal.pone.0109832.