Machine Learning Models for Predicting Protein Structure from Sequence

Lena Trnovec, Faculty of Computer and Information Science, lt89715@student.uni-lj.si

***Abstract*— Protein structure prediction is a critical issue in modern computational biology. Therefore, it is becoming increasingly important to predict the structure of proteins based on their amino acid sequence, using insights from already known structures. Although there are experimental methods to find out protein structures, the number of protein sequences and known structures is growing. To address these issues, various supervised and unsupervised machine learning methods have been used, significantly improving the state of the art in protein structure prediction. In this paper, we identify, characterize, and build three potential machine learning models that can predict secondary structure from an amino acid sequence. We implement the models on a Kaggle dataset and compare results.**

***Keywords*— protein structure prediction, Convolutional Neural network (CNN), Support Vector Machine (SVM), K-Nearest Neighbors (KNN)**

I. Introduction

A protein is a polymeric macromolecule composed of amino acid building blocks arranged in a linear chain and linked together by peptide bonds. The primary structure is usually represented by a sequence of letters of a 20-letter alphabet assigned to the 20 naturally occurring amino acids. Proteins account for nearly 20% of the weight of a eukaryotic cell, making them the largest component after water (70%). In its native environment, the chain of amino acids of a protein folds into local secondary structures, including alpha helices, beta strands, and nonregular coils. The secondary structure is specified by a sequence classifying each amino acid into the appropriate secondary structure element (e.g., alpha, beta, or gamma). The secondary structure elements are further packed to form a tertiary structure that depends on hydrophobic forces and side-chain interactions, such as hydrogen bonding, between amino acids. The tertiary structure is described by the x, y, and z coordinates of all the atoms in a protein or by the coordinates of the backbone atoms. Finally, several related protein chains may interact or assemble to form protein complexes (Figure 1). [1], [2]

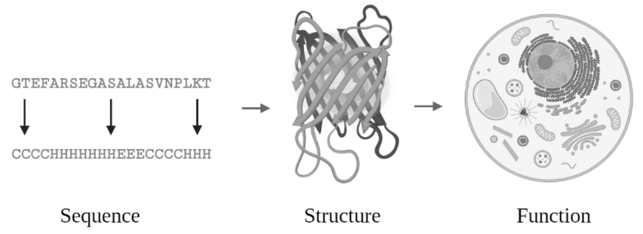


Fig. 1. The relationship between amino acid sequence, protein structure and protein function in cell. The amino acid sequence determines the secondary structure elements (alpha, beta, gamma) that are packed to form tertiary structure and finally assemble to form proteins.

Protein structure prediction is one of the most important problems in modern computational biology. Therefore, it is becoming increasingly important to predict the structure of proteins based on their amino acid sequence by using insights from already known structures. Although experimental methods (X-ray crystallography, nuclear magnetic resonance (NMR) spectroscopy, cryogenic electron microscopy (Cryo EM), and others) exist for determining protein structures, the gap between the number of protein sequences and known structures is widening. In this context, protein structure prediction is one of the most important problems in computational structural biology. [3]

The ability of machine learning-based predictors to predict secondary structure from sequence has largely been the focus of research over the past two decades. A number of competing predictors with similar methods, training datasets, protocols, and most importantly, similar prediction performance (e.g., about 80% for secondary structure prediction) have emerged from research over the past decade that has not significantly improved prediction accuracy. Most current approaches to secondary structure prediction use numerous neural networks, sometimes hundreds of them, most of which have been trained independently. Instead of using networks, many systems have been combined. Performance has also improved at the alignment level, as PSI-BLAST [4] is able to profile with increasingly distant homologs. [5], [6]

Over the years, various supervised and unsupervised machine learning methods have been applied to address these problems and have contributed significantly to improving the state of the art in protein structure prediction. The goal of this study is to identify, characterize, and build three different machine learning models that can predict secondary structure from an amino acid sequence. To this end, we first review recent efforts in protein structure prediction using machine learning, then preprocess a dataset from Kaggle [7] and train the models on it. We consider the importance of the parameters and finally compare the results.

II. Related Work

The most successful structure prediction methods to date are knowledge-based methods. Examples of knowledge-based methods include learning or extracting knowledge from previously solved protein structures and generalizing the gained knowledge to new proteins with unknown structures. Machine learning methods that can automatically extract knowledge from the PDB are an important class of tools that are widely used in all areas of protein structure prediction. A protein primary sequence is used as input to 1-D prediction problems, and the output is a sequence of predicted features for each amino acid in the sequence. The learning objective is to map the input amino acid sequence to the output feature sequence. The 1-D structure prediction problem is often viewed as a classification problem for each individual amino acid in the protein sequence. Early methods of secondary structure prediction relied on calculating statistical correlations between a window of consecutive amino acids in a protein sequence and the secondary structure classification of the amino acid in the center of the window. Simple correlation methods capture only a limited amount of information and have an accuracy of about 50%, which is far above chance. With the advancement of more powerful pattern recognition and nonlinear function matching methods, new approaches to predicting the secondary structure of proteins have emerged. [1]

Support vector machines (SVMs) perform well compared to other learning algorithms because they can effectively control the capacity of the classifier and the associated potential for overfitting. This is achieved by ensuring that the decision boundary separating two classes is made with a large margin. SVMs have other desirable properties such as efficient solutions, relatively few adjustable parameters, and the interchangeable use of kernel functions, which define a mapping of the data to a higher-dimensional feature space. [8]

Convolutional neural networks (CNNs) are another method; the first successful approaches date back to the late 1980s, and in 1993 a two-layer feed-forward network was the first algorithm to achieve more than 70% accuracy on the 3-state problem (discussed later in Data Preparation). PSIPRED, a technique using PSI-BLAST sequence profiles as improved input features and another simple two-layer network, achieved 80% classification accuracy by 1999. [9]

The conventional K-nearest neighbor (KNN) algorithm is another approach. This is a simple nonparametric classification method in which the class label of the input test sequence is assigned based on the class labels, which are usually represented by the number K of neighbors. Since KNN is an instance-based classification algorithm, it is easier to develop than SVM and neural networks and can classify multiple classes without increasing the complexity or time required for training. The KNN classification algorithm assigns an unknown sample to one of two classes by using a unique decision rule, even if there are two or more classes with equal proportions of samples. Moreover, the degree of confidence in the predicted class is unknown, which may increase the mishandling of uncertainties in classification problems. Related studies have proposed variations of the KNN classifier that led to better results, but we will compare it to other models using the conventional model. [10], [11]

III. Results

A. Data Preparation

The main Kaggle dataset lists peptide sequences and their corresponding secondary structures. It is a transformation of a RSCB PDB data set downloaded at 2018-06-06 (Table 1).

TABLE 1
DESCRIPTION OF COLUMNS

Column	Description
pdb_id	the id used to locate its entry on https://www.rcsb.org/
chain_code	when a protein consists of multiple peptides (chains), the chain code is needed to locate a particular one.
seq	the sequence of the peptide
sst8	the eight-state (Q8) secondary structure
sst3	the three-state (Q3) secondary structure
len	the length of the peptide
hasnonstdaa	whether the peptide contains nonstandard amino acids (B, O, U, X, or Z).

In predicting the secondary structure of proteins, we distinguish between 3-state prediction and 8-state prediction. In 3-state prediction, the goal is to classify each amino acid into either an alpha helix, i.e., a regular state marked with an 'H," a beta strand, i.e., a regular state marked with an 'E," or a coil region, i.e., an irregular state marked with a 'C' (Table 2).

TABLE 2
SIMPLIFICATION OF SECONDARY STRUCTURE INTO 8 AND 3 STATES

Secondary structure type	Q8 secondary structure	Q3 secondary structure
Loops and irregular elements (corresponding to the blank characters output by DSSP)	C	C
Turn	T	C
Bend	S	C
β-strand	E	E
β-bridge	B	E
α-helix	H	H
3-helix	G	H
π-helix	I	H

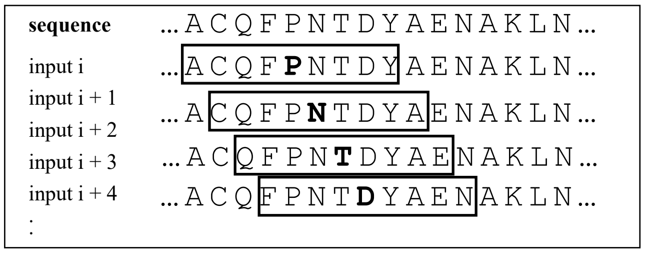
The key steps in the transformation were listing both Q3 and Q8 secondary structure sequences, masking all nonstandard amino acids, which includes B, O, U, X, and Z, with "*" character and adding an additional column (has_nonstd_aa) to indicate whether the protein sequence contains nonstandard amino acids. A subset of the sequences with low sequence identity and high resolution, ready for training, was also provided. This is the subset we used to train our models.

Orthogonal encoding (Table 3) was used to convert amino acid residues into numerical values and read the sliding window inputs. Based on the statistical correlation found between the secondary structure of a given residue position and the eight residues on either side of the prediction point, we chose a window of size 17. In the sliding window method, only the central amino acid is predicted, and binary encoding was used to assign numerical data to amino acid characteristics. As a result, there are 20 locations for amino acid characteristics. [12]

TABLE 3
10x17 ORTHOGONAL CODING INPUT

A	C	Q	F	P	N	T	D	Y	A	E	N	A	K	L	N	T	R
1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

For example, for each window of size 17, each amino acid entered is assigned a value of 1 depending on its position in the window, and all other positions are assigned 0. In this case, the input pattern consists of 20 x 17 inputs, of which 17 are assigned the value 1 and the rest are assigned the value 0. Figure 2 shows an example of the sliding window problem.



sequence ... A C Q F P N T D Y A E N A K L N ...
input i ... A C Q F P N T D Y A E N A K L N ...
input i + 1 ... A C Q F P N T D Y A E N A K L N ...
input i + 2
input i + 3 ... A C Q F P N T D Y A E N A K L N ...
input i + 4 ... A C Q F P N T D Y A E N A K L N ...
:

Fig. 2. The sliding window method (size 9). The window moves along the sequence and assigns the class to the center amino acid.

The data was split 80:20 into training set (TrS) and testing set (TeS). Random state was set to 0 to ensure the results are reproducible and that accuracy of models could be compared on the same testing set (TeS). Features were scaled and the distribution of classes is shown on figure 3.

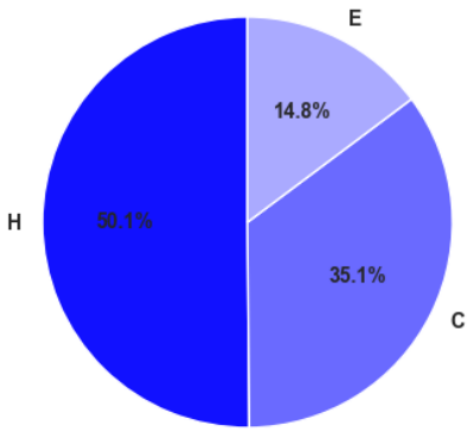
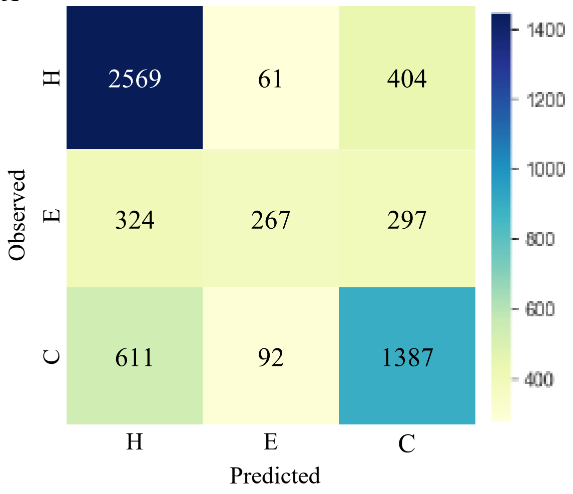


Fig. 3. The distribution of three-state classes in data subset (H, C, E).

B. SVM model

The SVM classifier constructs a hyperplane that divides the protein dataset into different classes according to orthogonal coding. In this work, the RBF kernel was used, and the cost parameter (C) was 1 throughout the experiment. We determined these parameters with Grid Search cross-validation provided by Scikit-learn on a smaller subset. The implementation of orthogonal encoding meant an increase in the amount of data. For this reason, and to avoid crashes, only a limited amount of data (30 000 windows) was analyzed. As a result, the accuracy of the model is somewhat lower than it would have been if the entire data set had been analyzed.

A



B

Accuracy score (TeS): 0.7024
Training set score (TrS): 0.8829
Null accuracy score: 0.5047
Average 6-fold CV score: 0.7077
Average 6-fold CV MAE: 0.4572
Average 6-fold CV MSE: 0.7870

	precision	recall	f1-score
H	0.73	0.85	0.79
E	0.64	0.30	0.41
C	0.66	0.66	0.66

C

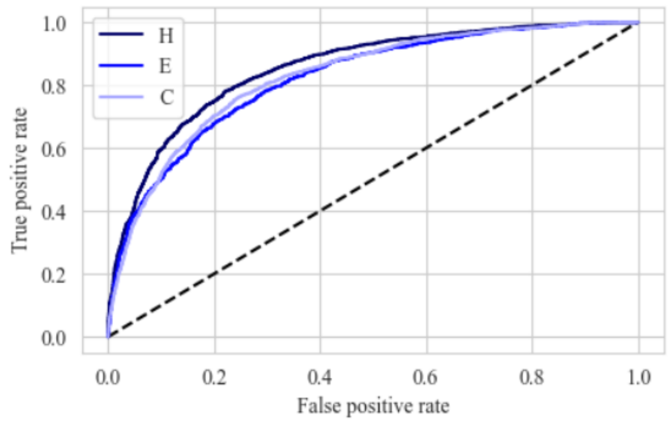


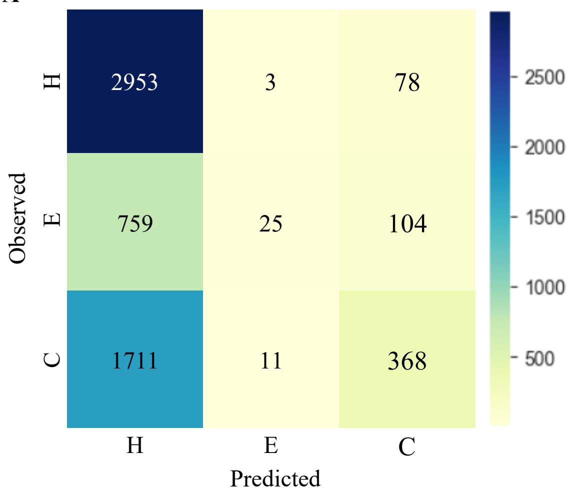
Fig. 4. A – Confusion matrix of SVM model evaluated on TeS. B – Summary of SVM model scores. C – SVM ROS curve.

Since classes are not equally represented, the recall and f1 scores for class E are lower. Still, the accuracy is higher than the null accuracy. To check for overfitting and underfitting, we used 6-fold cross validation (CV) and got a slightly higher average accuracy value. We also determined mean squared error (MSE) and mean average error (MAE).

C. *KNN model*

To determine the best value k for the KNN classifier we used Grid Search cross-validation provided by Scikit-learn. Moreover, k = 17 yielded the best result (Figure 5). We performed 6-fold cross validation to test for overfitting and obtained a slightly higher average accuracy value.

A



B

Accuracy score (TeS): 0.5566
Training set score (TrS): 0.5866
Null accuracy score: 0.5047
Average 6-fold CV score: 0.6042
Average 6-fold CV MAE: 0.6477
Average 6-fold CV MSE: 1.1516

	precision	recall	f1-score
H	0.54	0.97	0.70
E	0.64	0.03	0.05
C	0.67	0.18	0.46

C

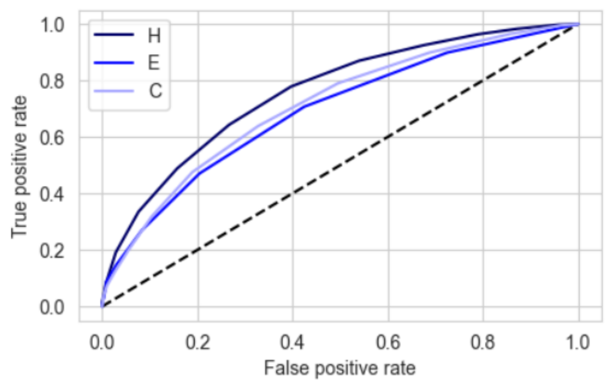


Fig. 5. A – Confusion matrix for KNN classifier (k=17). B – Summary of KNN model scores. C – KNN ROS curve.

D. *CNN model*

The final model we built was a convolutional neural network using eras library, as it has shown promising results in previous studies [13]. We determined the hidden layers and hyperparameters by observing models in related problems and by trial and error to minimize overfitting or underfitting while still getting high accuracy values (Table 4).

TABLE 4
CNN ARCHITECTURE AND HYPERPARAMETERS

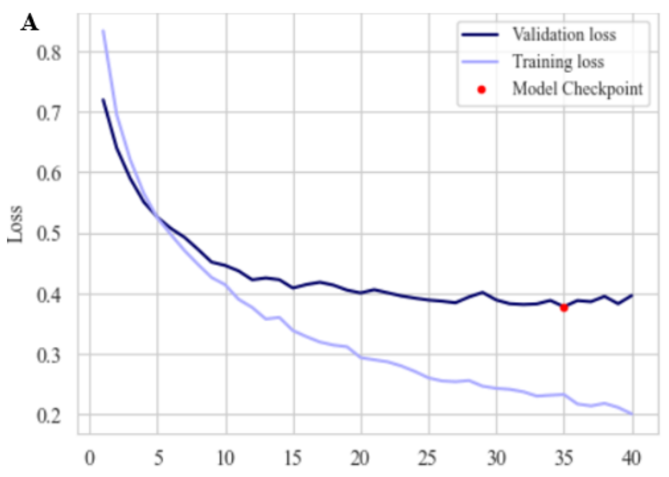
Learning Rate: 0.0009
Drop out: 0.38
Batch dim: 64
Number of epochs: 45
Loss: ‘categorical_crossentropy’

Layer (type)	Number of filters	Output shape	Number of parameters
Input		(17, 21)	
1D convolution layer (activation: relu, f=11, s=1)	128	(17, 128)	29696
Dropout (rate=0.38)		-	
1D convolution layer (activation: relu, f=11, s=1)	64	(17, 64)	90176
Dropout (rate=0.38)		-	
1D convolution layer (activation: relu, f=11, s=1)	3	(17, 3)	2115
Flatten		(51)	
Dense (softmax)		(3)	156

Trainable parameters: 122,143

s – stride
f – kernel size

To validate the model, we split 80:20 the training data into smaller training subset (TrS’) and validation set (VS). We fitted the model first to one validation data, then to the second, for 45 epochs in batches of size 64 (Figure 6). We defined the model checkpoint callback and stopped the fitting of the model when the validation loss did not improve after 5 epochs.



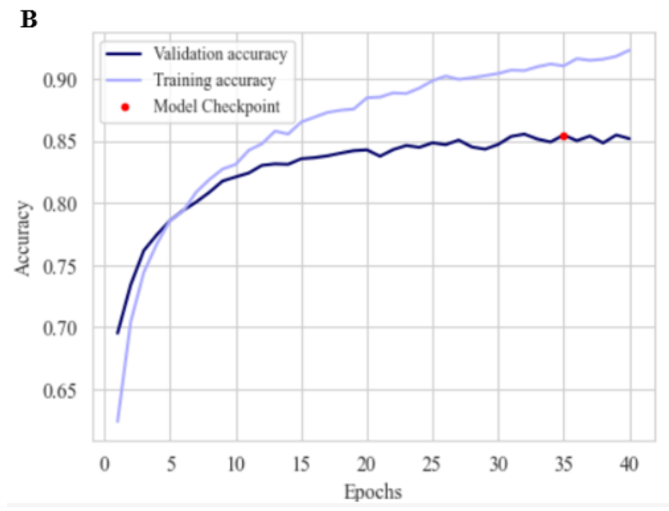
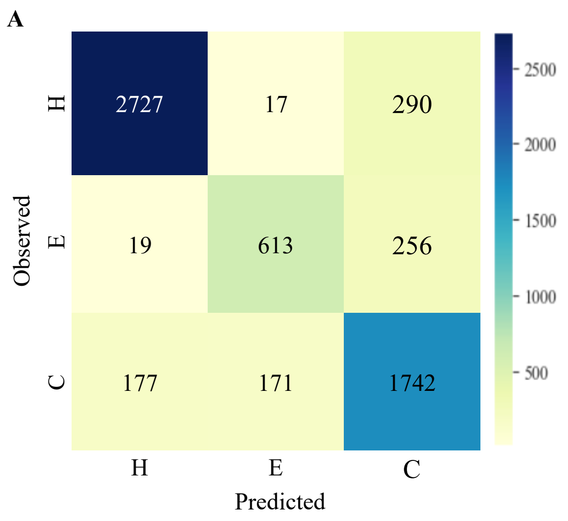


Fig. 6. A - Validation and training set loss in 35 epochs. B - Validation and training set accuracy in 45 epochs. The model used to test the data is labeled with Model Checkpoint.

After fitting the model, we evaluated the model on the testing set (Figure 7).



B

Accuracy score (TeS): 0.8453
Training set score (TrS'): 0.9889
Null accuracy score: 0.5047
Average 6-fold cross validation score: 0.8398
MAE: 0.1413
MSE: 0.0768

	precision	recall	f1-score
H	0.93	0.90	0.92
E	0.77	0.69	0.73
C	0.76	0.83	0.80

Fig. 7. A – Confusion matrix for CNN classifier. B – Summary of CNN model scores.

E. Model comparison

To determine the best model for three-state protein prediction from an amino acid sequence, we compared the model metrics (Table 5).

TABLE 5
MODEL COMPARISON

	SVM	KNN	CNN
Accuracy score on TeS	0.7024	0.5566	0.8453
Average 6-fold CV accuracy score	0.7077	0.6042	0.8398
Average 6-fold CV MAE	0.4572	0.6477	0.1413
Average 6-fold CV MSE	0.7870	1.1516	0.0768
H F1 score	0.79	0.70	0.92
E F1 score	0.41	0.05	0.73
C F1 score	0.66	0.46	0.80

IV. CONCLUSION

From the results, we can conclude that the best model for predicting secondary structure from amino acid sequence is the CNN model because its accuracy is the highest and it can reasonably classify even the classes that are not well represented in the data set (class E). It also has the lowest average MAE and MSE when performed with 6-fold cross validation. The second best model is the SVM model with an average 6-fold cross-validation accuracy of 0.71. The algorithm performs well in classifying classes H and relatively well in C, but fails to classify class E correctly. It also has a high MSE for 6-fold cross-validation. Although it is the fastest, the worst model is the KNN model, which performs reasonably well in classifying class H, but fails in classifying class C, and is terrible in classifying class E. The KNN model has a high MSE. Therefore, this model is not recommended for secondary structure prediction.

The models would be better if a larger amount of data had been used to train the models. Although the amount of information in the Kaggle dataset is sufficient, it is difficult to use such large quantities of data (using the sliding window approach) without encountering memory capacity issues. In the future, it would be interesting to train and test these models with larger data sets to see how well they perform in predicting eight-step secondary protein structure.

REFERENCES

[1] Jianlin Cheng, A. N. Tegge, and P. Baldi, “Machine Learning Methods for Protein Structure Prediction,” *IEEE Rev. Biomed. Eng.*, vol. 1, pp. 41–49, 2008, doi: 10.1109/RBME.2008.2008239.

[2] A. K. Mandle, “Protein Structure Prediction Using Support Vector Machine,” *IJSC*, vol. 3, no. 1, pp. 67–78, Feb. 2012, doi: 10.5121/ijsc.2012.3106.

[3] S. C. Pakhrin, B. Shrestha, B. Adhikari, and D. B. Kc, “Deep Learning-Based Advances in Protein Structure Prediction,” *IJMS*, vol. 22, no. 11, p. 5553, May 2021, doi: 10.3390/ijms22115553.

[4] “PSI-Blast at NCBI.” [Online]. Available: https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastSearch&PROGRAM=blastp&BLAST_PROGRAMS=psiBlast

[5] C. N. Magnan and P. Baldi, “SSpro/ACCpro 5: almost perfect prediction of protein secondary structure and relative solvent accessibility using profiles, machine learning and structural similarity,” *Bioinformatics*, vol. 30, no. 18, pp. 2592–2597, Sep. 2014, doi: 10.1093/bioinformatics/btu352.

[6] G. Pollastri, D. Przybylski, B. Rost, and P. Baldi, “Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles,” *Proteins*, vol. 47, no. 2, pp. 228–235, May 2002, doi: 10.1002/prot.10082.

[7] “Protein Secondary Structure Kaggle dataset.” [Online]. Available: https://www.kaggle.com/datasets/alfrandom/protein-secondary-structure

[8] J. J. Ward, L. J. McGuffin, B. F. Buxton, and D. T. Jones, “Secondary structure prediction with support vector machines,” *Bioinformatics*, vol. 19, no. 13, pp. 1650–1655, Sep. 2003, doi: 10.1093/bioinformatics/btg223.

[9] S. Wang, J. Peng, J. Ma, and J. Xu, “Protein Secondary Structure Prediction Using Deep Convolutional Neural Fields,” *Sci Rep*, vol. 6, no. 1, p. 18962, May 2016, doi: 10.1038/srep18962.

[10] W. Shang, H. Huang, H. Zhu, Y. Lin, Z. Wang, and Y. Qu, “An Improved kNN Algorithm – Fuzzy kNN,” in *Computational Intelligence and Security*, vol. 3801, Y. Hao, J. Liu, Y. Wang, Y. Cheung, H. Yin, L. Jiao, J. Ma, and Y.-C. Jiao, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 741–746. doi: 10.1007/11596448_109.

[11] Y. T. Tan and B. A. Rosdi, “FPGA-based hardware accelerator for the prediction of protein secondary class via fuzzy K-nearest neighbors with Lempel–Ziv complexity based distance measure,” *Neurocomputing*, vol. 148, pp. 409–419, Jan. 2015, doi: 10.1016/j.neucom.2014.06.001.

[12] L. H. Holley and M. Karplus, “Protein secondary structure prediction with a neural network.,” *Proc. Natl. Acad. Sci. U.S.A.*, vol. 86, no. 1, pp. 152–156, Jan. 1989, doi: 10.1073/pnas.86.1.152.

[13] J. Zhou, H. Wang, Z. Zhao, R. Xu, and Q. Lu, “CNNH_PSS: protein 8-class secondary structure prediction by convolutional neural network with highway,” *BMC Bioinformatics*, vol. 19, no. S4, p. 60, May 2018, doi: 10.1186/s12859-018-2067-8.