U.Porto Journal of Engineering, 8:2 (2022) 108-120
ISSN 2183-6493
DOI: 10.24840/2183-6493_008.002_0009

Received: 17 November, 2021
Accepted: 8 January, 2022
Published: 28 April, 2022

Optimization based Long Short Term Memory Network for Protein Structure Prediction

Pravinkumar M. Sonsare[1], Gunavathi C.[2*]
[1]School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, India; Department of Computer Science and Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur, India (sonsarep@rknec.edu) ORCID 0000-0002-8355-3334; [2*]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India (gunavathi.cm@vit.ac.in) ORCID 0000-0002-4996-069X (*corresponding author)

Abstract
One of the challenging tasks in computational biology is the anticipation of protein secondary structure (PSS) from amino acid sequences. Numerous computational and statistical methods are used for this purpose. With the growing attention of deep learning, models such as convolutional neural network and recurrent neural network are also used for this prediction. But, these strategies require a lot of hyperparameters tuning to accomplish the best outcome. In this paper, we proposed a bidirectional embedded recurrent deep neural system using long short term memory (LSTM) cells with continuous coin betting optimizer (COCOB) to tune the hyperparameters for the prediction of PSS. We have performed this experiment on Nvidia DGX station. We assessed our model on a FASTA-formatted file which consists of Protein Data Bank (PDB) sequences and their relative secondary structure. We report better performance (Q3=79.01% and Q8=82.38%) than best in class (Q3=64.9% and Q8=68.2%) methods.

Author Keywords. Proteomic, Bioinformatics, Deep Learning, Protein Secondary Structure, COCOB.

Type: Research Article
Open Access Peer Reviewed CC BY

1. Introduction

Many biotechnologists are teaming up with computer science researchers to do faster analysis of protein structure. Protein consists of 20 type of amino acids which are originated from DNA arrangement. There are four types of protein structures listed as primary, secondary, tertiary and quaternary. The tertiary structure prediction of a protein is a very important function in proteomics. It is based on the prediction of protein secondary structure. It gives significant bits of knowledge to understand the protein function. Existing techniques to make predictions on the protein structure are time-consuming. It couldn't address the issue of the real world like measuring the actual percentage of amino acids in three and eight state structure. An amino acid sequence is an important factor to anticipate the protein structure. It is also an important element to determine the function of the protein. Many significant researches are being carried out in the field of bioinformatics for protein secondary structure. Deep learning for amino acid sequence processing is similar to pattern recognition technique that is applied to words, sentences and paragraphs.

In this work, we have handled one of the crucial applications of bioinformatics. It interprets the structure of protein from amino acid sequences. This task is the base to foster protein function identification. Identification of protein function is useful to design the drugs. Normally, raw text strings are converted into tensors for deep neural system. In our implementation, we extracted n-grams of character from sequences. Then, we transformed each n-groups of sequence into a tensor. The n-grams are intersecting sets of numerous successive characters. We have provided this numeric tensor as input to bidirectional embedded recurrent deep neural system which has Continuous Coin Betting optimizer (COCOB). COCOB is a learning rate free optimizer inspired from a coin betting game.

Two unique classifiers namely Support Vector Machine (SVM) and fuzzy Nearest Neighbor has been used for this prediction task in the existing works. The outcome of the classification results are aggregated to summarize PSS (Khalatbari et al. 2019). Machine learning algorithms like k-Nearest Neighbor (kNN) classifier is used for the recognition purpose (Khedgaonkar, Raghuwanshi, and Singh 2018). A model named Particle Swarm Optimization (PSO)– Tabu search (TS) depends on the 2D Hydrophobic–Polar (HP) have used PSO and TS for the prediction purpose. TS can help PSO to avoid getting caught in local optima. TS expels the limitation of PSO in anticipating protein structure by the 2D-HP model (Yang et al. 2019). A model is constructed for predicting the secondary structure uses random forest, fuzzy SVM, Hidden Markov Models (HMMs). This model is able to determine the structure of an unknown protein (Lasfar and Bouden 2018; Kathuria, Mehrotra, and Misra 2018; Morshedian, Razmara, and Lotfi 2019). Numerous deep neural systems are modeled for finding the internal relations between amino acid sequences and PSS. It also solves complex relationships between amino acids (Zhang, Li, and Lü 2018; Wang, Mao, and Yi 2017; Hu et al. 2018; Babaei, Geranmayeh, and Seyyedsalehi 2010). There are many challenges that arise while predicting the protein secondary structure from protein sequence. They are complex relationship between structure and sequence, influence of feature to learner's effectiveness, partial noisiness in protein sequence and their related known structures, unequal distribution of amino acid samples into its classes (Krissinel 2007; Alirezaee, Dehzangi, and Mansoori 2012; Liu, Zheng, and Wang 2010). S-glutathionylation based proteins and their dependent are recognized by deep learning model (Li et al. 2020).

In the proposed model, our contributions are:

- We have processed the raw protein sequence strings using vectorization to convert them into numeric tensor.

- We devised a bidirectional embedded recurrent deep neural system. This deep neural system consists of long short term memory cell.

-We have used COCOB for hyperparameters tuning.

-We have compared the accuracy of the proposed model using various optimizers.

The rest of the paper is organized as follows. In section 2, we briefly described the preliminaries required for the proposed work. In section 3, we described the process of data preparation. In section 4, we explained the working of the proposed model. In sections 5 and 6, we examined the model with different optimizers. In section 7, we concluded the paper and discussed the future work.

2. Preliminaries

The secondary structure prediction of protein has been well studied in the literature. Researchers have used many deep learning approaches for PSS. In this section, we discussed the prerequisites required for the proposed work.

2.1. Deep Neural System

Deep learning is a type of machine learning approaches which is based on the interpretation of data. This system consists of a series of layers, which forms a neural network. It computes the likelihood of each output (Sonsare and Gunavathi 2019). Activation functions such as sigmoid, hyperbolic, softmax and Rectified Linear Unit (ReLU) are used for nonlinear transformation. Loss function and regularization are minimized by tuning the hyperparameters. Many optimization methods are also used for transformation like stochastic gradient descent (SGD), AdaGrad and RmsProp. They are selected according to the type of data considered for the analysis (Cao et al. 2018). Different architectures have been proposed in deep neural systems. These architectures consist of many networks. Recurrent neural network is an efficient deep neural system for processing the sequence of characters.

2.2. Recurrent Neural System

In most of the densely connected networks in neural network architecture, there is no memory. For this type of architecture, we have to give the entire sequence as input to the network at a time to show the independency. But, the sequence of characters or word has to be kept in memory to get the meaning conveyed by the sequence. A recurrent neural network (RNN) routes sequences by summarizing over the sequence features. Data which has been learned by recurrent neural network are maintained by a loop as shown in Figure 1. The state of the RNN is retuned amongst two different inputs. The sequences are two dissimilar independent protein sequences. We consider the data point as a single sequence. It is also the only input to the network.

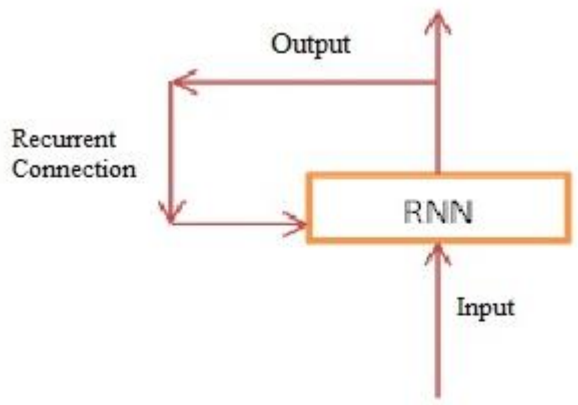


Figure 1: A Recurrent Network

Two dimensional tensor of size (timesteps, input_features) is input to the RNN. It iterates over timesteps. The RNN produces the output based on the given input at current time and current state. The current state is always the previous output. The initial state of RNN has been initialized with zero vectors. Transformation of parameters in a network is handled by tensors. Two matrices are used for this purpose and they are termed as weight matrices. W is the weight matrix connecting the input and hidden units. The weight matrix U connects the hidden units to output. One additional matrix is used for bias vector. Detailed pseudocode for the RNN (Figure 2) is as follows (Chollet 2018):

```
Initial_state = 0
for sequence_t in protein_sequence:
secstr_t = activation (dot (W, sequence_t) + dot(U, Initial_t) + bias)
initial_t = secstr_t
```

Figure 2: Pseudocode of RNN

The above recurrent neural network corresponds to the following Formula (1) and Formula (2).

$$h_{(t)} = g\left(b + Uh_{(t-1)} + Wx_{(i)}\right) \tag{1}$$

$$o_{(t)} = c + Vh_{(i)} \tag{2}$$

Here time is t, weights are W and V, b is bias for hidden layer, c is bias for output layer and an activation function is g.

2.3. Long Short Term Memory Cell (LSTM)

RNN is trained by adding more number of hidden layers using gradient based optimization to get better accuracy. During this backpropagation process of RNN, gradient tends to get smaller value which leads the network to learn slowly. This phenomenon of getting gradient smaller and smaller as training progresses is called as vanishing gradient. Due to the vanishing gradient problem it is difficult to learn long-term dependencies for simple RNN. The LSTM cells are used to carry information along every timesteps. LSTM prevents older signals from gradually vanishing during the learning process. This saves the information for future processing. In LSTM, data is carried across timesteps (Ct) which converts the current output to next state. LSTM experiences three distinct transformations in the form of RNN is shown in Figure 3. All the three transformations have different weight matrices.

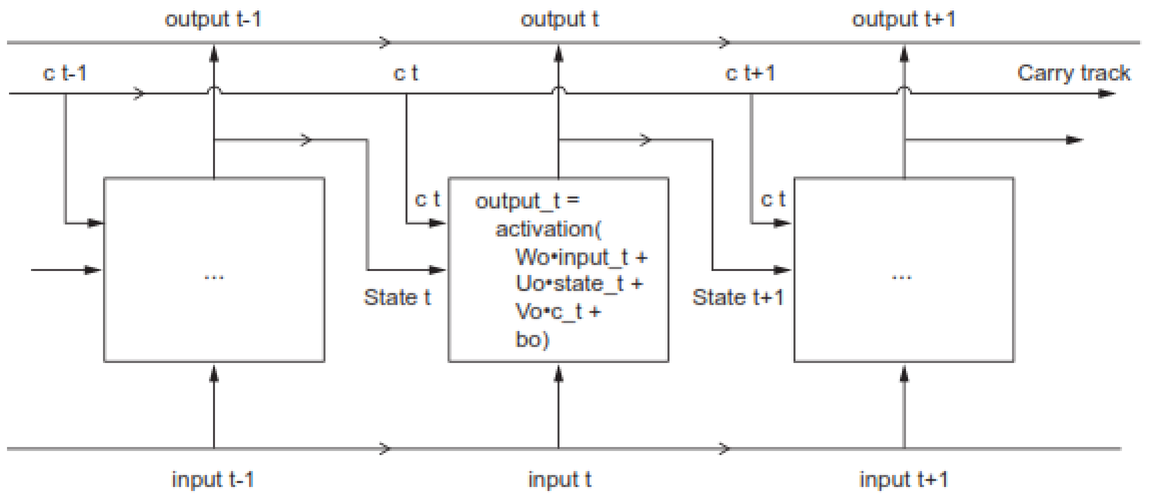
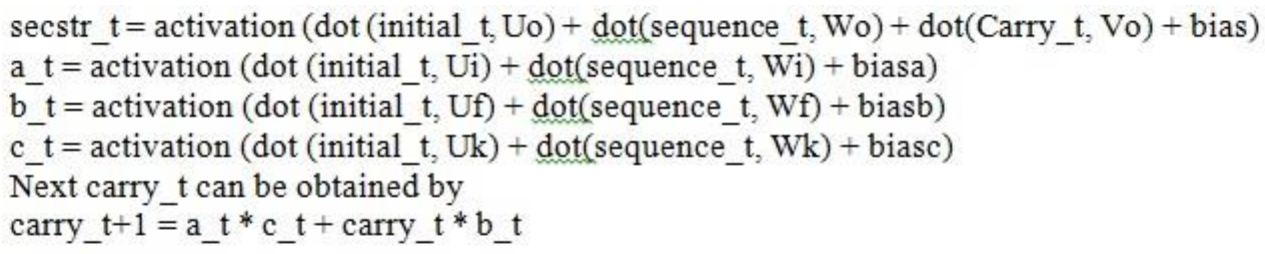


Figure 3: RNN with LSTM

Pseudocode for RNN with LSTM is shown in Figure 4 (Chollet 2018):



```
secstr_t = activation (dot (initial_t, Uo) + dot(sequence_t, Wo) + dot(Carry_t, Vo) + bias)
a_t = activation (dot (initial_t, Ui) + dot(sequence_t, Wi) + biasa)
b_t = activation (dot (initial_t, Uf) + dot(sequence_t, Wf) + biasb)
c_t = activation (dot (initial_t, Uk) + dot(sequence_t, Wk) + biasc)
Next carry_t can be obtained by
carry_t+1 = a_t * c_t + carry_t * b_t
```

Figure 4: Pseudocode for RNN with LSTM

2.4. Bidirectional RNN

In simple RNN, the entire sequence is known beforehand. It is not desirable in secondary structure prediction. The solution for this problem is bidirectional RNN (Schuster and Paliwal 1997). Bidirectional RNN has two separate RNN, the forward RNN starts from x1 and goes forward whereas backward RNN start from xn and goes backward. The output of forward network and backward network are combined and normalized to get the final output. Following (Formula 3-12) illustrate bidirectional LSTM model. This model uses feed-forward network that is responsible for getting final output using softmax prediction. This model inserts a feed-forward network between recurrent hidden states with shortcut connections between the recurrent hidden layers (Sønderby and Winther 2014).

$$i_t = \sigma(x_t W_{xi} + h_{t-1} W_{hi} + b_i) \tag{3}$$

$$f_t = \sigma\left(x_t W_{xf} + h_{t-1} W_{hf} + b_f\right) \tag{4}$$

$$o_t = \sigma(x_t W_{xo} + h_{t-1} W_{ho} + b_o) \tag{5}$$

$$g_t = tanh\left(x_t W_{xg} + h_{t-1} W_{hg} + b_g\right) \tag{6}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{7}$$

$$h_t = o_t \odot tanh(c_t) \tag{8}$$

$$h_{t-rec} = h_t + feedforwardnet(h_t) \tag{9}$$

$$\sigma(z) = \frac{1}{1 + exp(-z)} \tag{10}$$

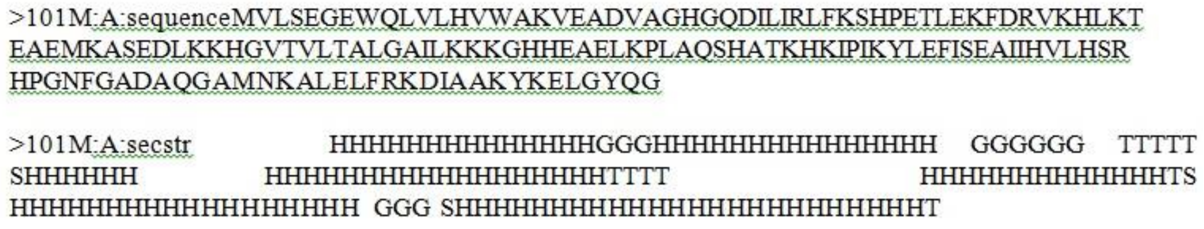
$$\odot = Elementwise\ multiplication \tag{11}$$

$$x_t = input\ from\ the\ previous\ layer: h_t^{t-1} \tag{12}$$

3. Methods

3.1. Data preparation

The dataset used for this study comprises of amino acid sequences. It is also labeled with secondary structure. The dataset is downloaded from PDB (https://cdn.rcsb.org/etl/kabschSander/ss.txt.gz) and annotated with Dictionary of Secondary Structure of Proteins program (DSSP) (Kabsch and Sander 1983). The example of protein sample in the dataset is shown in Figure 5.

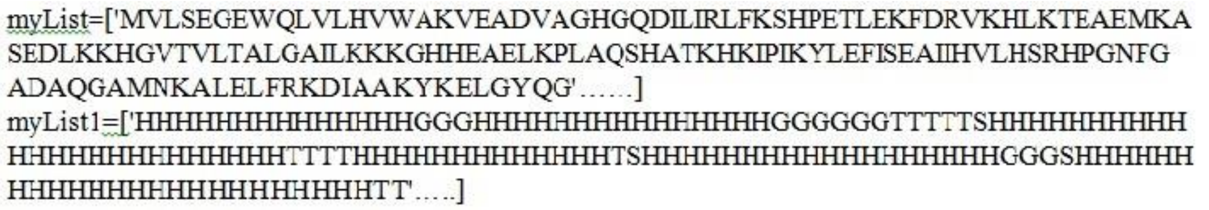


```
>101M:A:sequenceMVLSEGEWQLVLHVWAKVEADVAGHGQDILIRLFKSHPETLEKFDRVKHLKT
EAEMKASEDLKKHGVTVLTALGAILKKKGHHEAELKPLAQSHATKHKIPIKYLEFISEAIIHVLHSR
HPGNFGADAQGAMNKALELFRKDIAAKYKELGYQG

>101M:A:secstr          HHHHHHHHHHHHHHHGGGHHHHHHHHHHHHHHHH   GGGGGG   TTTTT
SHHHHHH          HHHHHHHHHHHHHHHHHHHTTTT                    HHHHHHHHHHHHHHTS
HHHHHHHHHHHHHHHHHHH  GGG SHHHHHHHHHHHHHHHHHHHHHHHHHHT
```

Figure 5: Protein Sample in the dataset

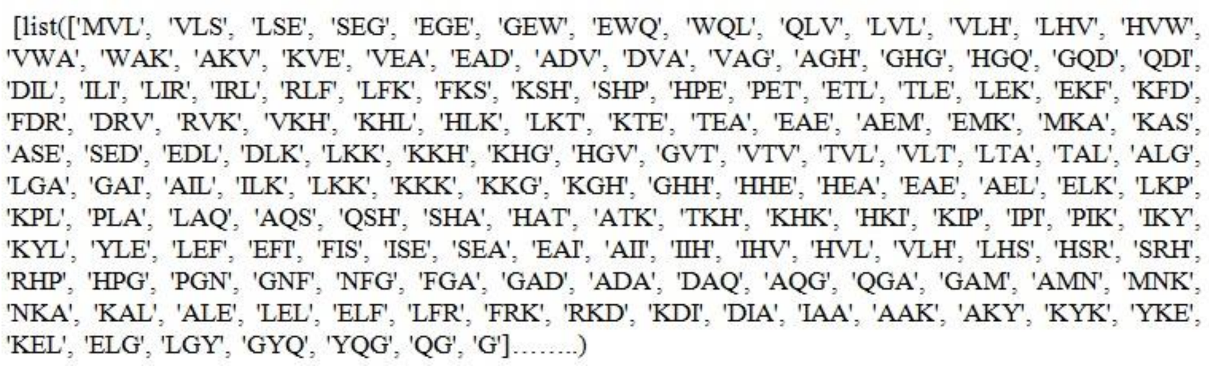
Sequences consist of amino acid along with few undefined characters. The PSS is classified into eight states. These eight states are further combined into three states namely H, E and C. In the three states, H (α-helix) consists of H, G; E(β-strand) consists of E,B; C(loop) consists of rest of the classes from eight states. We have collected this sequence of characters in two different lists using biopython package (Figure 6).



```
myList=['MVLSEGEWQLVLHVWAKVEADVAGHGQDILIRLFKSHPETLEKFDRVKHLKTEAEMKA
SEDLKKHGVTVLTALGAILKKKGHHEAELKPLAQSHATKHKIPIKYLEFISEAIIHVLHSRHPGNFG
ADAQGAMNKALELFRKDIAAKYKELGYQG'......]
myList1=['HHHHHHHHHHHHHHHGGGHHHHHHHHHHHHHHHHGGGGGGTTTTTSHHHHHHHHHH
HHHHHHHHHHHHHHHTTTTHHHHHHHHHHHHHHTSHHHHHHHHHHHHHHHHHHHGGGSHHHHHH
HHHHHHHHHHHHHHHHHHTT'.....]
```

Figure 6: List created using Biopython Package

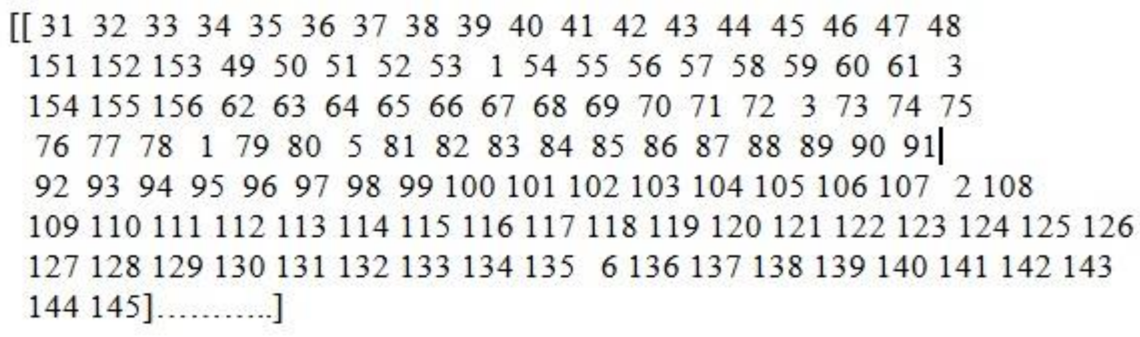
Deep learning model takes numeric tensor as input instead of raw text. This can be done by the process of vectorization. We used n-grams of character for vectorization. Overlying clusters of several successive characters is named as n-grams. We transformed each 3-gram of character into a vector (Figure 7).



```
[list(['MVL', 'VLS', 'LSE', 'SEG', 'EGE', 'GEW', 'EWQ', 'WQL', 'QLV', 'LVL', 'VLH', 'LHV', 'HVW',
'VWA', 'WAK', 'AKV', 'KVE', 'VEA', 'EAD', 'ADV', 'DVA', 'VAG', 'AGH', 'GHG', 'HGQ', 'GQD', 'QDI',
'DIL', 'ILI', 'LIR', 'IRL', 'RLF', 'LFK', 'FKS', 'KSH', 'SHP', 'HPE', 'PET', 'ETL', 'TLE', 'LEK', 'EKF', 'KFD',
'FDR', 'DRV', 'RVK', 'VKH', 'KHL', 'HLK', 'LKT', 'KTE', 'TEA', 'EAE', 'AEM', 'EMK', 'MKA', 'KAS',
'ASE', 'SED', 'EDL', 'DLK', 'LKK', 'KKH', 'KHG', 'HGV', 'GVT', 'VTV', 'TVL', 'VLT', 'LTA', 'TAL', 'ALG',
'LGA', 'GAI', 'AIL', 'ILK', 'LKK', 'KKK', 'KKG', 'KGH', 'GHH', 'HHE', 'HEA', 'EAE', 'AEL', 'ELK', 'LKP',
'KPL', 'PLA', 'LAQ', 'AQS', 'QSH', 'SHA', 'HAT', 'ATK', 'TKH', 'KHK', 'HKI', 'KIP', 'IPI', 'PIK', 'IKY',
'KYL', 'YLE', 'LEF', 'EFI', 'FIS', 'ISE', 'SEA', 'EAI', 'AII', 'IIH', 'IHV', 'HVL', 'VLH', 'LHS', 'HSR', 'SRH',
'RHP', 'HPG', 'PGN', 'GNF', 'NFG', 'FGA', 'GAD', 'ADA', 'DAQ', 'AQG', 'QGA', 'GAM', 'AMN', 'MNK',
'NKA', 'KAL', 'ALE', 'LEL', 'ELF', 'LFR', 'FRK', 'RKD', 'KDI', 'DIA', 'IAA', 'AAK', 'AKY', 'KYK', 'YKE',
'KEL', 'ELG', 'LGY', 'GYQ', 'YQG', 'QG', 'G']........)
```

Figure 7: Vector Representation of 3-gram character

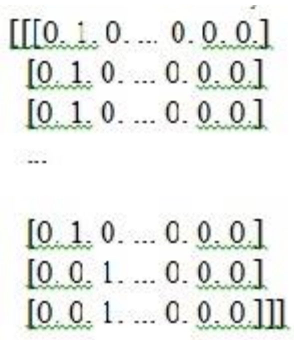
We have used tokenization scheme for text-vectorization process. This associates numeric vectors with the generated tokens. After tokenization, the input data tensors are shown as follows (Figure 8):



```
[[ 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48
  151 152 153  49  50  51  52  53   1  54  55  56  57  58  59  60  61   3
  154 155 156  62  63  64  65  66  67  68  69  70  71  72   3  73  74  75
   76  77  78   1  79  80   5  81  82  83  84  85  86  87  88  89  90  91
   92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107   2 108
  109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
  127 128 129 130 131 132 133 134 135   6 136 137 138 139 140 141 142 143
  144 145]...........]
```

Figure 8: Input Data Tensor

After tokenization, the output data tensors are shown as follows (Figure 9):



```
[[[0. 1. 0. ... 0. 0. 0.]
  [0. 1. 0. ... 0. 0. 0.]
  [0. 1. 0. ... 0. 0. 0.]
  ...

  [0. 1. 0. ... 0. 0. 0.]
  [0. 0. 1. ... 0. 0. 0.]
  [0. 0. 1. ... 0. 0. 0.]]]
```

Figure 9: Output Data Tensor

After preprocessing the raw data, we got input data of tensor size (41789, 128) and target data of size (41789, 128, 8). This means that there are 41789 labeled protein samples from PDB. This input tensor, the processed raw text is set as an input to deep learning prototype.

4. The Proposed Model and Algorithm

The proposed model of secondary structure prediction of protein is presented in this section. It uses bidirectional embedded recurrent deep neural system for the transformation of sequences. Learned sequences are remembered using long short term memory cells. This architecture uses a continuous coin betting optimizer (COCOB) for tuning the hyperparameters. We have executed our code on Nvidia DGX station, which has four GPU core.The procedure for learning the parameters is discussed here in detail.

4.1. Bidirectional Embedded Recurrent Deep Neural System with Long Short Term Memory Cell and Continuous Coin Betting Optimizer

A bidirectional RNN offers more prominent execution than an ordinary RNN on specific applications. It utilizes two customary RNNs to achieve their order sensitivity. In this work, we have used LSTM layers which used forward and backward architecture for output and input respectively. The word embedding connects a vector with a word that learned from the information.

4.2. Embedding layer

The embedding layer provides a low dimensional input to the actual network. It is subsequently sensible to learn new embedding space with each new task. This is simplified by backpropagation. The embedding layer is tied with learning the weights of a layer. It requires a lexicon with the dimension of embedding to get reduced tensor.

Word index $\longrightarrow$ Embedding layer $\longrightarrow$ Corresponding word vector

Two dimensional matrices of whole numbers are given as an input to the embedding layer. Every entry is a series of numbers in the shape of (samples, sequence_length). It can insert successions of variable lengths. Here, some shapes are (32, 10), which translates into a bunch of 32 arrangements of length 10. The shapes (64, 15) are a cluster of 64 groupings of length 15. All entries in a group must have a similar length to be embedded into a solitary tensor. Sequences that are shorter in length than others ought to be cushioned with zeros; sequences those are longer ought to be shortened by dense vector.

An embedding layer returns multi-dimensional floating-point m. The return tensor is in the shape of (sequences, length of sequences, dimensionality of embedding). An RNN layer then processes this 3D tensor. In our study, 3D floating-point tensor has the shape of (9037, 128, 128).

At the point when we start up an Embedding layer, its weights are arbitrary, similar to some other layers. These word vectors are continuously balanced through backpropagation during training. The embedding vector will rearrange the learned information for a specific scenario.

4.3. Bidirectional LSTM with dropout

A bidirectional RNN enhances the efficiency of sequential linear RNNs. It sees its information arrangement in two different ways as shown in Figure 10, obtaining possibly better result and repeating pattern of features that may have been missed by the sequential request.

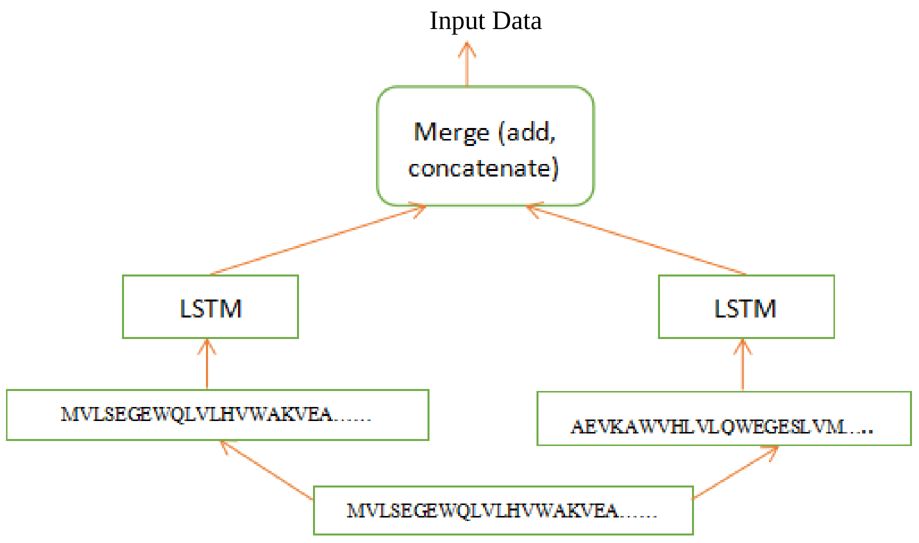


Figure 10: Sequence arrangement for Bidirectional LSTM

Bidirectional layer takes two parameters. First parameter is a repetitive layer instance of sequences. Second is separate example of this repetitive layer. It handles single occurrence of information groupings in sequential request. The other occurrence is for preparing the input arrangement in reverse request which appears to be overfit. Bidirectional layer has twice the same number of parameters as an ordered LSTM. The bidirectional methodology would be a solid performer on this task.

Overfitting of the model is shown by a training and validation curve. The losses of model begin to deviate impressively after a couple of epochs. Dropout is a method which haphazardly reduces the input units of a layer by breaking the chance of connections in the training information that the layer is presented. Similar dropout throughout the network is preferable for learning of the network. Recurrent dropouts determine the dropout pace of the recurrent units.

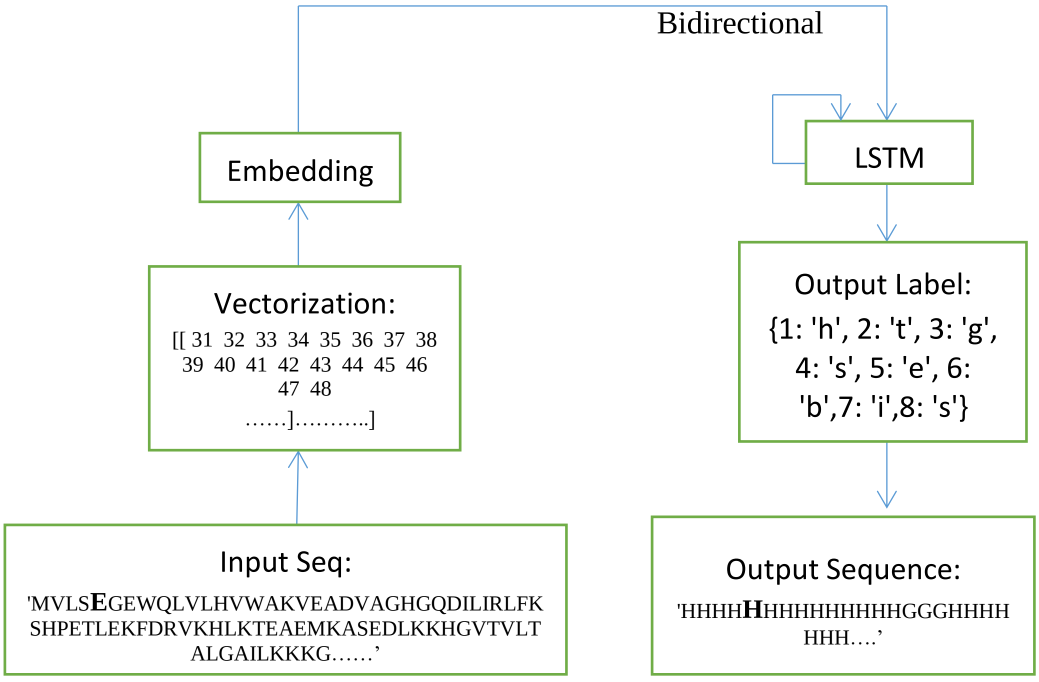

Bidirectional
Embedding
LSTM
Vectorization:
[[31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 ……]………..]
Output Label:
{1: 'h', 2: 't', 3: 'g', 4: 's', 5: 'e', 6: 'b',7: 'i',8: 's'}
Input Seq:
'MVLSEGEWQLVLHVWAKVEADVAGHGQDILIRLFK SHPETLEKFDRVKHLKTEAEMKASEDLKKHGVTVLT ALGAILKKKG……'
Output Sequence:
'HHHHHHHHHHHHHHHGGGHHHH HHH….'

Figure 11: Flow chart of the Proposed Model

Our bidirectional LSTM layer has 64 hidden units with recurrent drop rate 0.1 to avoid overfitting. This bidirectional LSTM is given with embedded input. We have also used timedistributed wrapper. TimeDistributed wrapper is applied on each layer. These layers consist of temporal slice of an input. TimeDistributed dense uses dense operation on each timesteps of a 3D tensor. The proposed protein secondary structure recurrent LSTM is shown in Figure 11. We have tuned the weights using continuous coin betting optimizer.

4.4. COntinuous COin Betting Algorithm

COntinuous COin Betting (COCOB) is a novel algorithm for stochastic subgradient descent, which optimizes a function with bounded subgradient, and so reducing the optimization time for learning a network. Stochastic gradients are linked with the outcomes of the coins. COCOB uses the sign and the amount of bets to maintain the gradient of network. COCOB is a learning rate free optimization process (Orabona and Tommasi 2017).

The optimization procedure is reduced to a game of betting on a coin, in which, the player starts betting with some initial amount ε, which is higher than zero. Consider +1 signifies heads and −1 means tails in a set gt={-1,+1}. Any amount on head or tail can be bet by a player. If he loses then he loses his betted sum. If he wins then he recovers the betted sum and gets a similar sum as a reward (Orabona and Tommasi 2017).

The player's bet is encoded in round t. It is betted by an only number wt. The sign of wt encodes +1 and -1 for bets on heads and tails respectively. The betted sum is encoded by an absolute value. Here, Wealth_t and Reward_t are outstanding and winning amount (Orabona and Tommasi 2017).

$$Wealth_t = \epsilon + \sum_{i=1}^{t} w_i g_i \tag{13}$$

$$Reward = Wealth_t - \epsilon = \sum_{i=1}^{t} w_i g_i \tag{14}$$

We will also refer to a bet with βt, where βt is such that

$$w_t = \beta_t Wealth_{t-1} \tag{15}$$

Player's decision on heads and tails is shown by sign of βt. The restriction is that players can't obtain money implies that βt ∈ [−1, 1]. We also somewhat sum up the issue by permitting the result of the coin flip gt to be any genuine number in [−1, 1], that is a ceaseless coin; wealth and reward in Formula (13) and Formula (14) continue as before.

The results of the coins are connected to the stochastic gradient. Every result of a coin is equivalent to the arrangement of negative stochastic slope. COCOB depends on the technique to bet a signed part of the present wealth. Wealth includes a small amount of enormous total wealth. This leads to a succession of equivalent result like gradient. This permits to build bet for any value of wealth. This methodology guarantees that the wealth of a player is constantly positive. The calculation is without scale in light of the fact that multiplying all the subgradient and coin flip by any positive factor would bring about a similar succession of wt,i (Orabona and Tommasi 2017).

Despite of everything, it has to know the most extreme range of the gradient on each epoch. Each layer will have an alternate scope of the gradient for the impact of the vanishing gradients (Hochreiter 1991). Additionally, the weights of the system can develop after some time, expanding the estimation of the gradients as well. Consequently, it is difficult to know the range of every angle in advance and utilize any procedure dependent on betting.

COCOB-Backprop is used for better optimization of network which deals with range of gradients. It enforces the positivity of reward but does not give guarantees. Removal of sigmoid function simplifies the optimization and gives better result. The assessments of the hyperparameters in the initial iterations of the algorithm are restricted. This is evaluated by changing the value of the bet fraction (Orabona and Tommasi 2017).

5. Results

The proposed methodology is applied to PDB dataset. The percentage of residues for which the predicted secondary structures are accurate is defined as the Q3 and Q8 accuracy, and it is evaluated for Q3 (Singh 2005) and Q8 (Kabsch and Sander 1983) prediction. We executed the proposed model on PDB secondary structure dataset, using the COCOB optimizer for hypertuning the parameters. We have compared the performance of COCOB optimizer with the popular stochastic gradient learning algorithms RMSProp, Adadelta, AdaGrad, Adam, Adamax and SGD. We have implemented bidirectional LSTM with COCOB in tensorflow. We also implemented other optimizers in tensorflow. We created an embedding layer on top of the bidirectional LSTM with 64 units. The bidirectional network output returns a single vector that is then processed to eight state softmax networks that give the probability of state of protein secondary structure. The network is regularized using 10% dropout. We have run our experimental set up for 41789 proteins PDB ID with their associated secondary structure. We combined all protein samples in a list and trained further.

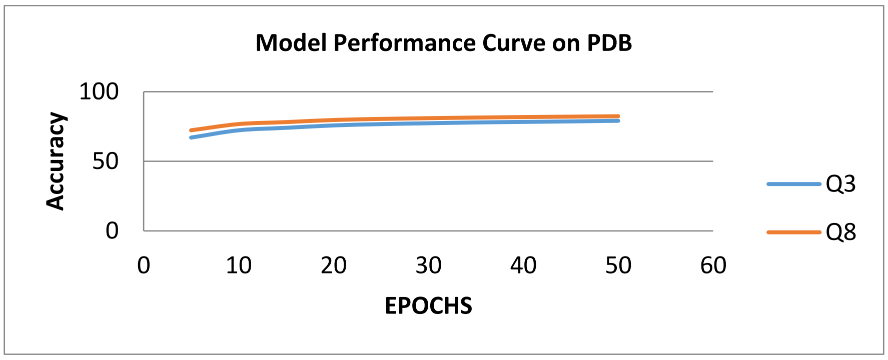
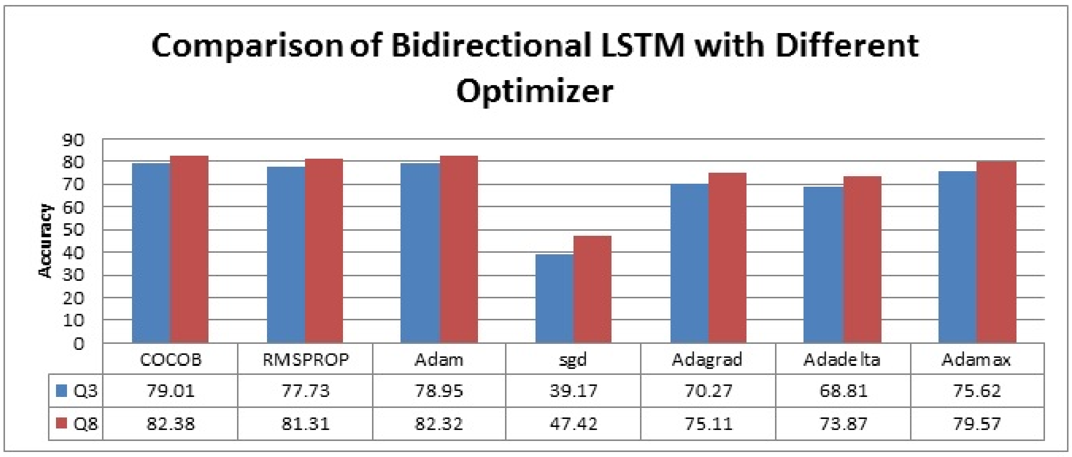


Figure 12: Model Performance Curve on PDB

Figure 12 shows the prediction accuracy for PDB secondary structure prediction. Our model shows better accuracy about 80% after 50 epochs which is better than the result in Zhou and Troyanskaya (2014) which requires 300 epochs.



	COCOB	RMSPROP	Adam	sgd	Adagrad	Adadelta	Adamax
🟦 Q3	79.01	77.73	78.95	39.17	70.27	68.81	75.62
🟥 Q8	82.38	81.31	82.32	47.42	75.11	73.87	79.57

Figure 13: Comparison of Bidirectional LSTM with Different Optimizers

We have executed our model using different optimizers. We have observed that our model with COCOB optimizer reports better result than other optimizers like RMSProp and Adam, and outperformed the other optimizers SGD, Adagrad, Adadelta and Adamax. Figure 13 shows the comparison between all optimizers we have used in our model.

6. Discussion

We have compared the performance of our proposed model for protein secondary structure prediction with some other methods in the literature. Table 1 shows the comparison of Q8 accuracy of our model with few other models in literature. As per our knowledge, our model gives better result than the best model in the literature.

Model	Q8 Accuracy(%)
BRNN (Pollastri et al. 2002)	51.1
CNF (Wang et al. 2011)	64.9
GSN (Zhou and Troyanskaya 2014)	66.4
LSTM (Sønderby and Winther 2014)	67.4
SSREDN (Wang, Mao, and Yi 2017)	68.2
Our Model	82.38

Table 1: Q8 accuracy Comparison

We also compared our results with some publically available programs with the same setup. We have also compared with programs like SSpro, RaptorX for Q3 prediction. We achieved a Q3 accuracy which outperforms SSpro and RaptorX accuracies. The Result comparison is listed in Table 2.

Model	Q3 Accuracy(%)
SSpro	63.5
RaptorX	64.9
Our Model	79.01

Table 2: Q3 accuracy Comparison

7. Conclusion and Future Work

In this research work, we have proposed a bidirectional embedded recurrent deep neural system with long short term memory cell. The proposed model uses continuous coin betting optimizer to tune hyperparameters for the prediction of sequence-structure relation for protein secondary structure. We compared our method with best in class methods available in the literature. We also compared our method with available programs like SSpro and RaptorX. Our method shows significant improvement in the performance. We have achieved 79.01% and 82.38% accuracy for Q3 and Q8 prediction respectively. For the further improvement in the proposed model, it can be tested with different deep learning architectures with refinement in the parameters. In future, this model can be modified with heuristic algorithm for optimization of weights in the architecture.

References

Alirezaee, M., A. Dehzangi, and E. Mansoori. 2012. "Ensemble of neural networks to solve class imbalance problem of protein secondary structure prediction". *International Journal of Artificial Intelligence & Applications* 3, no. 6: 9-20. https://doi.org/10.5121/ijaia.2012.3602.

Babaei, S., A. Geranmayeh, and S. A. Seyyedsalehi. 2010. "Protein secondary structure prediction using modular reciprocal bidirectional recurrent neural networks". *Computer Methods and Programs in Biomedicine* 100, no. 3: 237-47. https://doi.org/10.1016/j.cmpb.2010.04.005.

Cao, C., F. Liu, H. Tan, D. Song, W. Shu, W. Li, Y. Zhou, X. Bo, and Z. Xie. 2018. "Deep learning and its applications in biomedicine". *Genomics, Proteomics and Bioinformatics* 16, no. 1: 17-32. https://doi.org/10.1016/j.gpb.2017.07.003.

Chollet, F. 2018. *Deep Learning with Python*. Manning.

Hochreiter, S. 1991. "Untersuchungen Zu Dynamischen Neuronalen Netzen". Master’s thesis, Institut Für Informatik, Technische Universität München.

Hu, Y., T. Nie, D. Shen, and G. Yu. 2018. "Sequence translating model using deep neural block cascade network: Taking protein secondary structure prediction as an example". In *Proceedings - 2018 IEEE International Conference on Big Data and Smart Computing, BigComp 2018*, 58-65. IEEE. https://doi.org/10.1109/BigComp.2018.00018.

Kabsch, W., and C. Sander. 1983. "Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features". *Biopolymers* 22, no. 12: 2577-637. https://doi.org/10.1002/bip.360221211.

Kathuria, C., D. Mehrotra, and N. K. Misra. 2018. "Predicting the protein structure using random forest approach". *Procedia Computer Science* 132: 1654-62. https://doi.org/10.1016/j.procs.2018.05.134.

Khalatbari, L., M. R. Kangavari, S. Hosseini, H. Yin, and N. M. Cheung. 2019. "MCP: A multi-component learning machine to predict protein secondary structure". *Computers in Biology and Medicine* 110: 144-55. https://doi.org/10.1016/j.compbiomed.2019.04.040.

Khedgaonkar, R., M. M. Raghuwanshi, and K. R. Singh. 2018. "Patch-based face recognition under plastic surgery". In *ICSCCC 2018 - 1st International Conference on Secure Cyber Computing and Communications*, 364-68. IEEE. https://doi.org/10.1109/ICSCCC.2018.8703270.

Krissinel, E. 2007. "On the relationship between sequence and structure similarities in proteomics". *Bioinformatics* 23, no. 6: 717-23. https://doi.org/10.1093/bioinformatics/btm006.

Lasfar, M., and H. Bouden. 2018. "A method of data mining using Hidden Markov Models (HMMs) for protein secondary structure prediction". *Procedia Computer Science* 127: 42-51. https://doi.org/10.1016/j.procs.2018.01.096.

Li, S., K. Yu, D. Wang, Q. Zhang, Z. X. Liu, L. Zhao, and H. Cheng. 2020. "Deep learning based prediction of species-specific protein S-glutathionylation sites". *Biochimica et Biophysica Acta - Proteins and Proteomics* 1868, no. 7: Article number 140422. https://doi.org/10.1016/j.bbapap.2020.140422.

Liu, T., X. Zheng, and J. Wang. 2010. "Prediction of protein structural class using a complexity-based distance measure". *Amino Acids* 38, no. 3: 721-28. https://doi.org/10.1007/s00726-009-0276-1.

Morshedian, A., J. Razmara, and S. Lotfi. 2019. "A novel approach for protein structure prediction based on an estimation of distribution algorithm". *Soft Computing* 23, no. 13: 4777-88. https://doi.org/10.1007/s00500-018-3130-0.

Orabona, F., and T. Tommasi. 2017. "Training deep networks without learning rates through coin betting". In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 1-11. https://proceedings.neurips.cc/paper/2017/hash/7c82fab8c8f89124e2ce92984e04fb40-Abstract.html.

Pollastri, G., D. Przybylski, B. Rost, and P. Baldi. 2002. "Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles". *Proteins: Structure, Function and Genetics* 47, no. 2: 228-35. https://doi.org/10.1002/prot.10082.

Schuster, M., and K. K. Paliwal. 1997. "Bidirectional recurrent neural networks". *IEEE Transactions on Signal Processing* 45, no. 11: 2673-81. https://doi.org/10.1109/78.650093.

Singh, M. 2005. "Predicting protein secondary and supersecondary structure". In *Handbook of Computational Molecular Biology*, 29.1–29.29. Chapman and Hall/CRC Press.

Sønderby, S. K., and O. Winther. 2014. "Protein secondary structure prediction with long short term memory networks". Preprint, submitted December 25, 2014. https://arxiv.org/abs/1412.7828.

Sonsare, P. M., and C. Gunavathi. 2019. "Investigation of machine learning techniques on proteomics: A comprehensive survey". *Progress in Biophysics and Molecular Biology* 149: 54-69. https://doi.org/10.1016/j.pbiomolbio.2019.09.004.

Wang, Y., H. Mao, and Z. Yi. 2017. "Protein secondary structure prediction by using deep learning method". *Knowledge-Based Systems* 118: 115-23. https://doi.org/10.1016/j.knosys.2016.11.015.

Wang, Z., F. Zhao, J. Peng, and J. Xu. 2011. "Protein 8-class secondary structure prediction using conditional neural fields". *Proteomics* 11, no. 19: 3786-92. https://doi.org/10.1002/pmic.201100196.

Yang, C. H., Y. S. Lin, L. Y. Chuang, and Y. D. Lin. 2019. "Effective hybrid approach for protein structure prediction in a two-dimensional Hydrophobic–Polar model". *Computers in Biology and Medicine* 113: Article number 103397. https://doi.org/10.1016/j.compbiomed.2019.103397.

Zhang, B., J. Li, and Q. Lü. 2018. "Prediction of 8-state protein secondary structures by a novel deep learning architecture". *BMC Bioinformatics* 19, no. 1: Article number 293. https://doi.org/10.1186/s12859-018-2280-5.

Zhou, J., and O. G. Troyanskaya. 2014. "Deep supervised and convolutional generative stochastic network for protein secondary structure prediction". In *31st International Conference on Machine Learning, ICML 2014*, 1121-29.

Acknowledgements

We would like to thank the Department of Computer Science and Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur for allowing us to experiment on Nvidia DGX station and the School of Information Technology and Engineering, Vellore Institute of Technology, Vellore for their continuous support.