Protein Secondary Structure Prediction using Deep Neural Network and Particle Swarm Optimization Algorithm

Angela U. Makolo
Department of Computer Science
University of Ibadan, Ibadan, Nigeria

Adetayo Sylvester Ibijola
Department of Computer Science
University of Ibadan, Ibadan, Nigeria

ABSTRACT
The use of laboratory techniques such as X-ray Crystallography and Nuclear Magnetic Resonance (NMR) Spectroscopy for protein secondary structure prediction, although effective, are expensive and time-consuming especially when it is to be done in a large scale. Computational techniques are being employed to predict the structures of protein in order to overcome these limitations. Various methods such as HMM, SVM, ANN, etc. have been used for the prediction of the protein secondary structure with different accuracies, weaknesses, and strengths. The current prediction accuracy obtained from these existing tools has been between 60% and 80% over years and research is still ongoing to get better prediction accuracy in the prediction of the secondary structure of proteins. In this work, deep neural network with three (3) hidden layers and particle swarm optimization algorithms are combined to predict the secondary structure of proteins from their primary structures (Amino Acids Sequence). The basic particle swarm optimization algorithm was used in training a deep neural network as implemented using Java programming language with spring boot framework for generating the various APIs. The dataset used was retrieved from the JPred Server 1.2 which contained 1349 training set and 149 test set. The model had a maximum accuracy of 53.18% on epoch 180 due to the early convergence of the model at local minimal.

Keywords
Deep Neural Networks, DNN, Protein Strictures, Particle Swarm Optimization Algorithm, BPSO, Protein Secondary Structure Prediction, Swarm Intelligence.

1. INTRODUCTION
Proteins perform most biological and chemical functions in a cell which is why they are referred to as the building blocks of a cell [2]. The function of a protein is solely determined by its structure. This is more reason why protein structural bioinformatics is one of the vital research areas in bioinformatics. Predicting the structure of a protein from his primary structures (amino acids) is a challenging area in structural bioinformatics research [8].

There are four hierarchies of protein structures. These are the primary structure, secondary structure, tertiary structure, and the quaternary structure. The primary structure is made up of the sequence of amino acids. This is considered the simplest structure as amino acid residues are linked linearly with the peptide bond (PPMVTPSVGASAGVVA) The sequences of amino acid in a particular protein structure is not arranged randomly but are rather fixed as was discovered first by Frederick Sanger when he successfully sequenced the protein Insulin using the x-ray crystallography [3]. The secondary structures are local conformation of protein structures which are stabilized by hydrogen bond [5]. They are basically made up of Alpha helices (α-helices), Beta sheets (β-Sheets) and the coils. The Tertiary structure is the 3D structure of the protein. They come in various forms which are generally categorized into Globular or Membrane proteins.

Particle Swarm Optimization (PSO) algorithm is a computational method that optimizes a function by iteratively trying to improve a candidate solution based on its properties and its environment. This was originally designed by Kennedy and Eberhart in 1995 for studying social behavior [4]. But as study proceeds, it became obvious that the behavior of the algorithm is good for optimization and this algorithm has been applied successfully to wide varieties of search and optimization problems. The PSO algorithm originated from the simulation of a simplified social system of birds flocking, fish schooling and ants swarm. It was observed that for example there is a piece of food in an area and a group of birds in this particular search area are in search of the food. Now, they don't know where the food is, but they know how far the food is on every movement. The most optimized way to get to this food is to follow the closest bird to the food [7].

ANN is composed of layers that are made up of neurons. Typically, an ANN is composed of the input layer that accepts the features to be trained, the hidden layer and then the output layer that classifies the input. The neurons in each layer are connected to the neurons in the next layer by a set of weights and the output of each layer is calculated with the neurons and the weights that connect them. Thereafter, an activation function is used in moderating the output from every neuron based on the error term from the output layer.

In the case of a deep neural network, there are multiple hidden layers that enhance training. These features are being learned over and over for a better accuracy in the classification. ANN is popularly trained with gradient descent and backward propagation while the error term is calculated with the Mean Squared Error (MSE).

In this work, a model is developed to train a Deep Neural Network (DNN) with Particle Swarm Optimization (PSO) Algorithm for predicting Protein Secondary Structure and compare our result against the conventional way of training a Neural Network with Gradient Descent using the backward propagation.

Section 2 reviewed some literature from previous works done in this research space while Section 3 explained the methodology of the model adopted and then proceeded to the discussion of the result. Section 4 gave the conclusion of the work while Section 5 gave some recommendation for future works.

2. RELATED WORKS
This section presents some works that have been done in protein secondary structure prediction.

[1] considered four proteins which are Hemoglobin, Sickle Cell Anemia, Myoglobin, and Insulin. He used a coding scheme based in BCD (Binary Coded Decimal) to represent each amino acid for a Neural Network input layer. A Multilayer Perceptron (MLR) feedforward neural network with backward propagation for weight update was employed also. This work employed three artificial neural networks. The first is used to classify the amino acid into their various BCD code, the second then classifies them into their various primary structure while the third is used is to classify the output of the second ANN into the secondary structure.

(Busia A, Collins J &Jaitly N, 2016) used the CullPDB and CB513 dataset. They represented the primary sequence in a 42-dimensional vector- the first 21 dimension is the one-hot encoding of the Amino acid while the last 21 is the Position Specific Scoring Matrix (PSSM). TensorFlows's ADAM optimizer was used in training the ANN. (Busia A, Collins J &Jaitly N, 2016) employed two models which are the Fully-Connected model and the Convolutional model. The Fully Connected model used a deep neural network of five feedforward layers. They accepted a window size of seventeen (17) for the input layer and used an error rate of 0.00004 which is reduced by 50% on every 35,000 training iteration by the ADAM's optimizer. The convolutional model is then built upon the Deep Neural Network to improve the fixed size window input.

[6] argued that the use of Neural Networks and Support Vector Machines for the prediction of protein secondary structure is not ideal due to the fact the primary structure of proteins cannot naturally be represented as a vector of fixed dimensionality. [6] recognized that sliding window is used to get around this problem but still argued that sliding window is not efficient in solving this problem. He thereby used the long short term memory networks without peepholes because it has been revealed in recent papers that the model works better without the peepholes.

[2] investigated the use of cuckoo search algorithm which is a global optimization algorithm inspired by the behavior of the cuckoo species such as *Crotophaga Ani* and *Guira Cuckoo* and the foraging patterns of animals. At the end of the experiment, they concluded that the Cuckoo search algorithm was able to solve the local convergence trapping problem experienced by other Evolutionary Algorithms in the prediction of protein structure.

[5] used the DeepCNF in the prediction of both 3-state and 8-state protein secondary structure prediction. The DeepCNF is said to have combined the advantages of both the Conditional Neural Fields (CNF) and the Deep Convolutional Neural Network (DCNN). The work used the five publicly available datasets from CullPDB, CB513, CASP10, and CASP11. This method is said to have outperformed other methods including PSIPRED. This method is said to have obtained an accuracy of 83.8% on 8-state and 71.9% on 3-state. Nonetheless, this method is said to have a drawback on proteins with very sparse sequence profile. This makes it very challenging to predict from the primary sequence instead of sequence profile using this method.

3. METHODOLOGY

Various computational methods such as HMM (Hidden Markov Model), ANN (Artificial Neural Network), SVM (Support Vector Machine), etc. have been used in the prediction of protein secondary structure problem. In this work, the Particle Swarm Optimization Algorithm is used to train a Deep Neural Network for the prediction of the Protein Secondary Structure.

3.1 Deep Neural Network

Neural network has been in existence for a long time now but its applications of recent have produced a high performance and state of art results in many fields. They are used in image recognition, natural language processing and in many more problems.

The neural network simulates the operations of the human brain. The human brain is primarily composed of nerve cells called neurons (Fig 1). These neurons are connected via a strand of fiber called dendrites. These dendrites are used to accept information from other neurons and then pass out electrical spikes (information) through the axons.

The axon then splits into various branches which have a structure called synapse at the tip. These synapses convert the information from the axons into electrical effect that triggers or excites activity in the connected neuron.

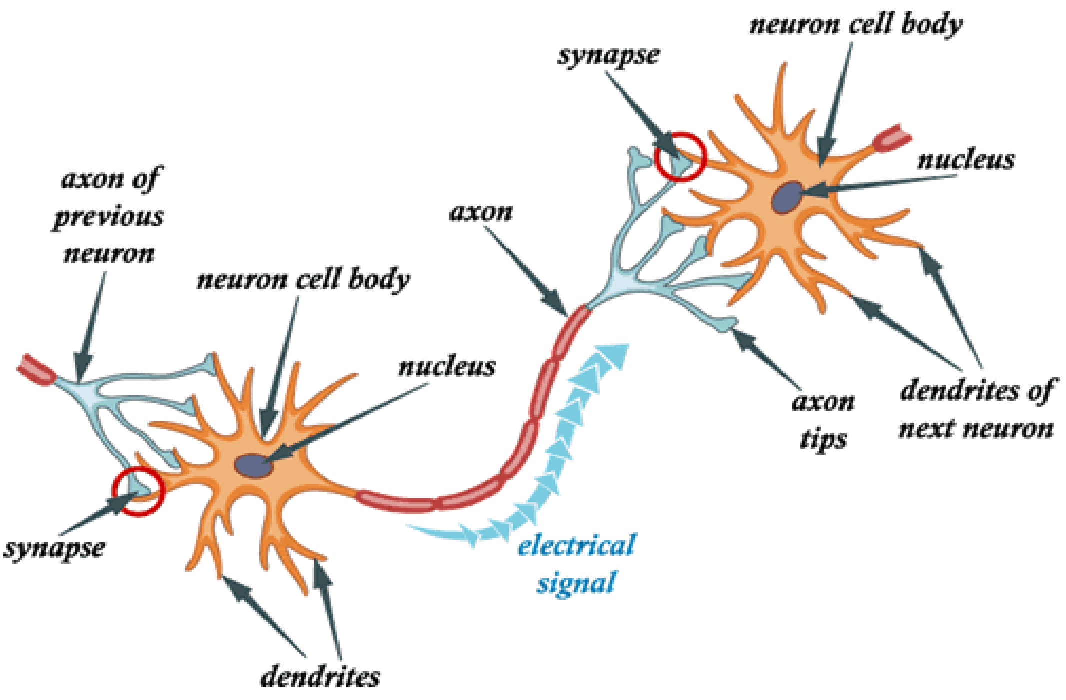

Fig 1: Human Brain Network

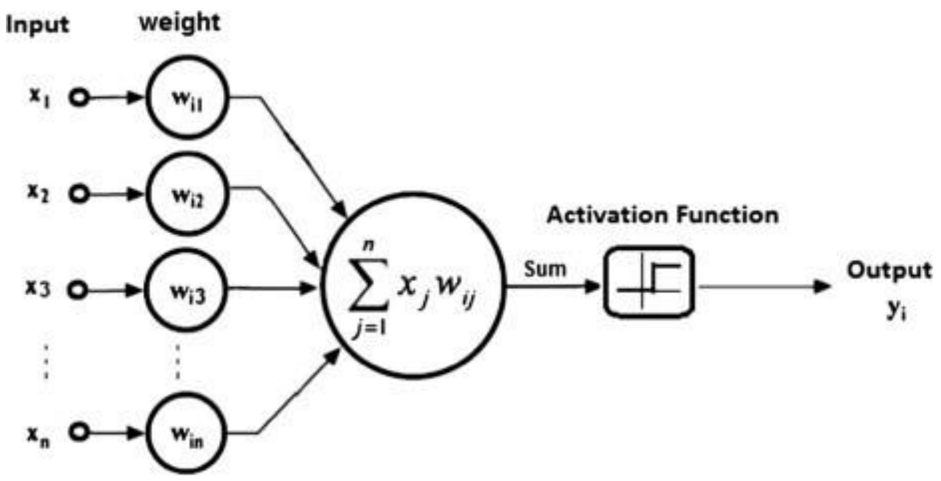

Fig 2: Artificial Neural Networks

The concept of these neuron activities is emulated in the Artificial Neural Network (ANN). ANN is composed of various layers (depends on how it is designed) and these layers consist of neurons. The neurons in this layer are connected to the next layer with set weights. These weights can be seen as the signal strength that is used to calculate the value of the next neuron with the help of some mathematical functions called the activation function.

In a multilayered neural network, the output generated from the activation function is then used as the input to the next layer.

There are various variants of ANN but the most popularly used is the feed-forward neural network. This neural network is composed of an input layer, zero or more hidden layer(s), and an output layer. A neural network with multiple hidden layers is referred to as a deep neural network.
Popularly, neural networks are trained with the Stochastic Gradient Descent and Backward propagation.

Backward propagation is a supervised learning algorithm of artificial neural networks using gradient descent. This algorithm is fully called “backward propagation of errors” because the error of the output is calculated using the weights with a given error function and then propagated back through the network layers. Based on the error derivative gotten, the weights of the network are then updated and the output regenerated.

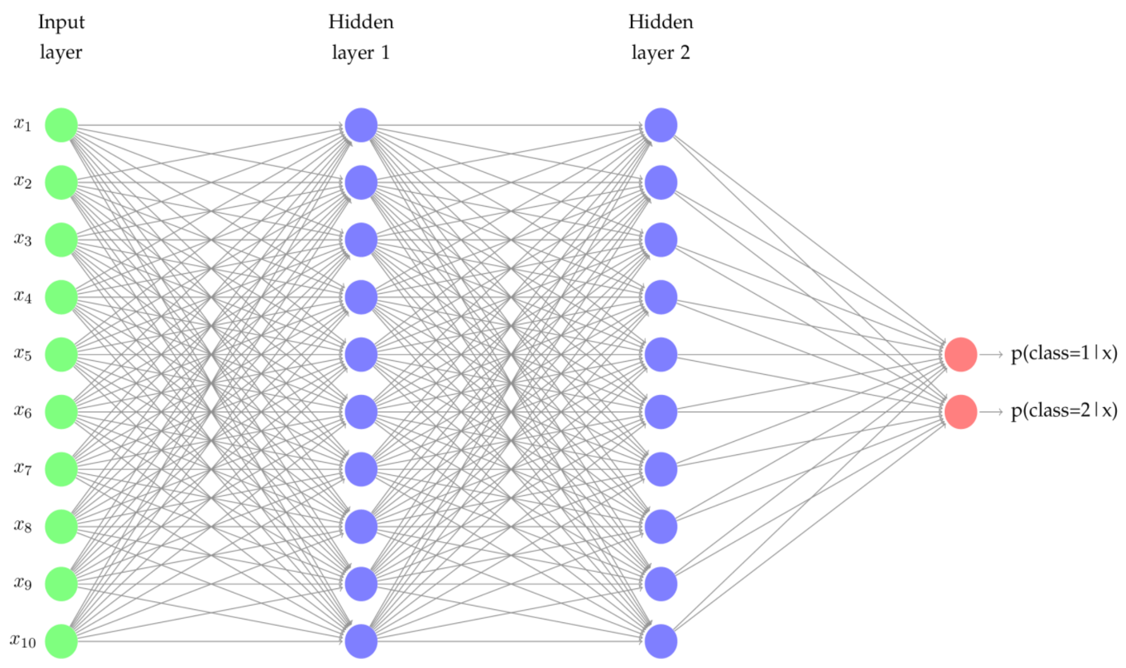

Fig 1: Deep Neural Network

The most popular error (loss) function used in this training is the Mean Squared Error (MSE). In the MSE, there is a vector of the target value (t(x)) that the model is trained to learn and then a vector of outputs (y(x)) from the output layer. The error is then calculated as shown in equation (3.1) below.

$$MSE = \frac{1}{n}\sum(y(x) - t(x))^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.1)$$

y(x): output value
t(x): target value

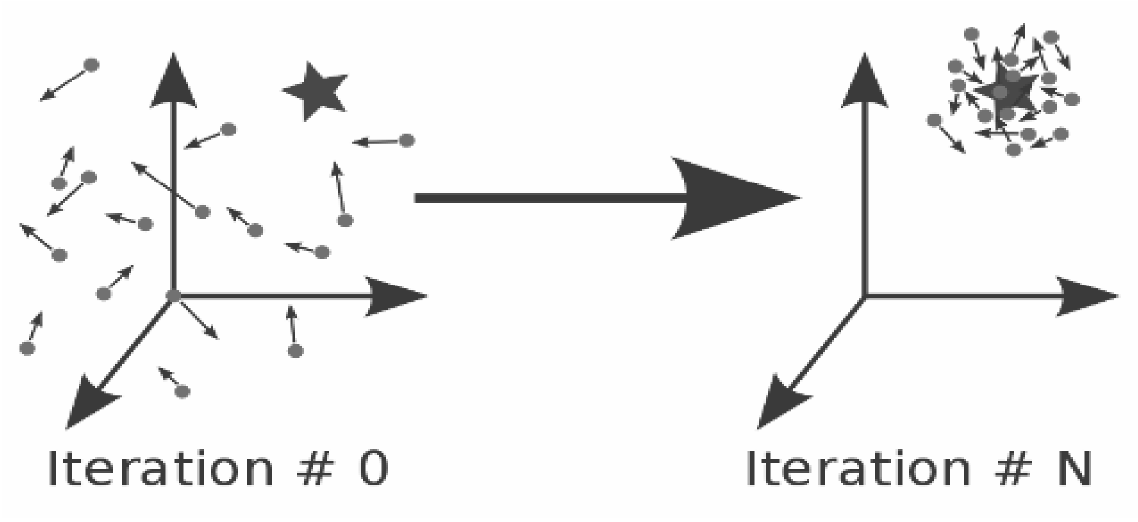

Fig 2: Particle Swarm Optimization

The derivative of this error function is then used in updating the weights of the network.

3.2 Particle Swarm Optimization(PSO)

PSO is a "metaheuristic" as it makes no assumptions about the problem been optimized and it can search very large space of possible solutions. Unlike the backward propagation, PSO does not require the problem to be differentiable as it does not use the gradient of the problem to be optimized. It looks for the best solution in a search area called the swarm.

Potential solutions are represented in particles and then introduced into the swarm. Each particle has a position, velocity, error, best position and best error. Also, the swarm has a general best position which is been referred to as the "global best position" and a global best error.

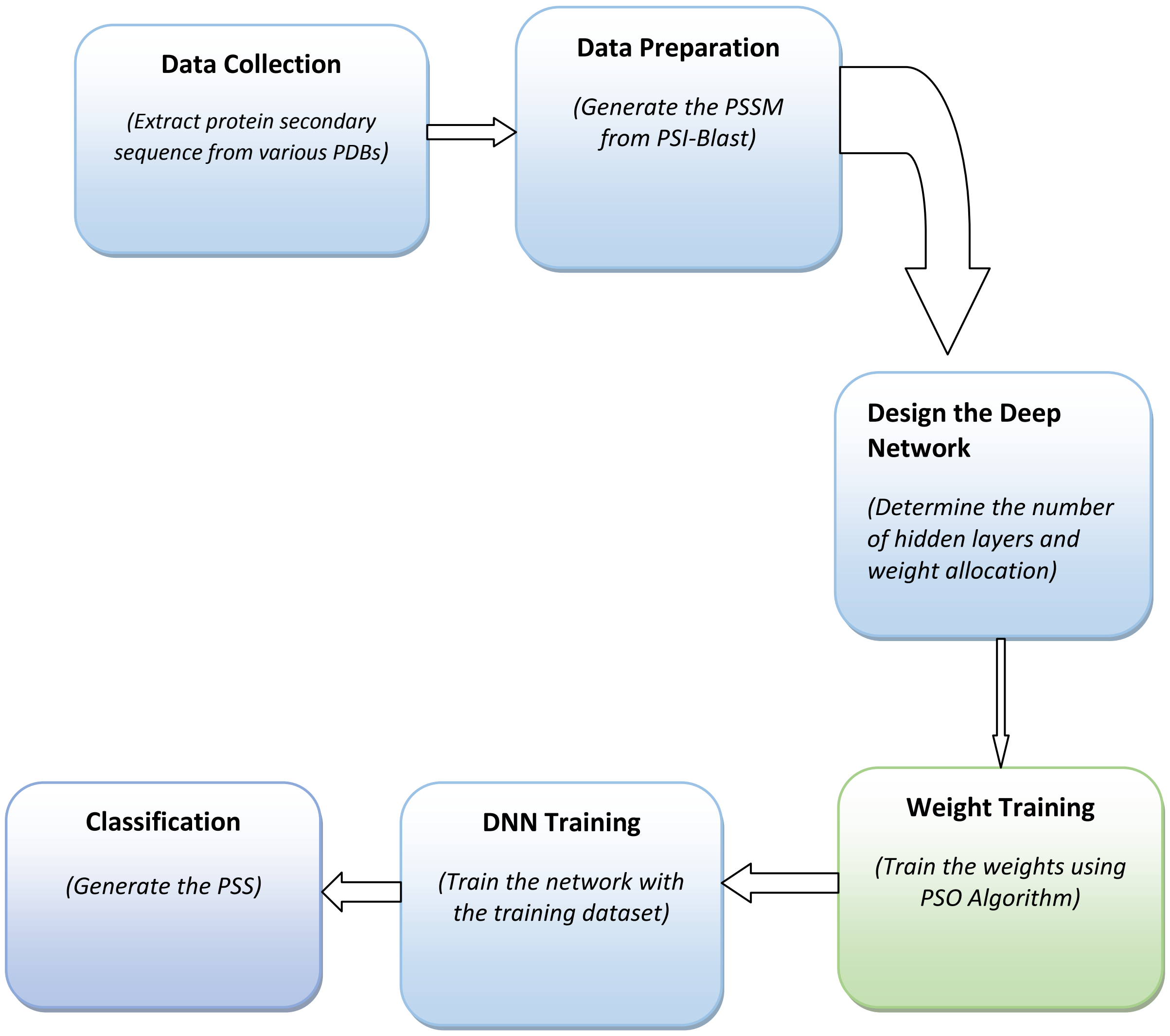
Data Collection
(Extract protein secondary sequence from various PDBs)
Data Preparation
(Generate the PSSM from PSI-Blast)
Design the Deep Network
(Determine the number of hidden layers and weight allocation)
Weight Training
(Train the weights using PSO Algorithm)
DNN Training
(Train the network with the training dataset)
Classification
(Generate the PSS)

Fig 3: Training a deep neural network with particle swarm optimization algorithm

Initially, the position of these particles is randomly assigned and the error is calculated with a fitness function to know how far they are from the target solution. The velocity of the particle is then updated based on its present position, present velocity, personal best position, global best position, inertial weight, cognitive constant and some random values r between 0 and 1.

3.2.1 Particle Swarm Optimization (PSO) Algorithm

```
1. Initialize each particle to a random state (position, velocity, error, personal best error, personal best position)
2. Initialize the global best position and error to the best position and error of any of the particles
3. Loop until done
    a. For each particle in the swarm,
        i. Calculate the velocity of the new particle
        ii. Use new velocity calculated to generate the new position of the particle,
        iii. calculate the error of the new position
        iv. If the new error is better than the particle's personal best error, then
            1. Set personal best error equals new error
        v. If the new error of particle is better than global best error
```

```
            1. Set global best error equals to the new error
            2. Set global best position equals new position
      b. End for
  4. End loop
Return global best position
```

Mathematically,

$v_{t+1} = \omega * v_t + c_1 * r_1 * (p_b - x_t) + c_2 * r_2 * (p_g + xt$ .... (3.2)

$x_{t+1} = x_t + v_{t+1}$ .... (3.3)

$$0 \leq r_1, r_2 \leq 1$$

$\omega: InertialWeigth$

$v_t: velocityattimet$

$p_b: PersonalBest$

$p_g: GlobalBest$

$x_t: Positionofparticleattimet$

$c_1, c_2: CognitiveandGloalConstatnts$

$r_1, r_2: randomnumbers$

As the training continues, the particle gradually moves towards the optimum position (Fig 4).

The output of the network is coded as follows

$$Coil(-) = 0,0,1$$

$$Helix(H) = 0,1,0$$

$$Sheet(E) = 1,0,0$$

The protein Sequence profile is a (n x 20) vector. Each row calculates the probability of the availability of a particular amino acid in the general twenty amino acids thereby the residue is coded to be a 20-dimensional vector which serves as input to the deep neural network.

The layers of this network are connected with a set of weight and bias. The output of each layer is calculated using an activation function. The sigmoid function was adopted for this project which is defined in (3.4).

$S(x) = \frac{1}{1+e^{-x}}$.. (3.4)

Here, x is the sum output generated from the weights, bias and the value of the previous neuron given by equation (3.5).

$x_n = \theta + \sum w_{kn} + o_k$ .... (3.5)

Thereby, it can be said that the activation function then generates equation (3.6).

$S(x_n) = \frac{1}{1+e^{-(\theta+\sum w_{kn}+o_k)}}$.................................... (3.6)

$\theta = biasterm$

$x_n = sumofaparticularneuroninalayer$

$w_{kn} = Weightconnectingneurononthepreviouslayertothepresentneuron$

$o_k = outputofthepreviousneuron$

After these outputs are generated, a fitness function (Mean Squared Error) is then used to calculate the deviation of the result from the target and then training begins.

Mean Squared Error (MSE) is the mean of the squared of the

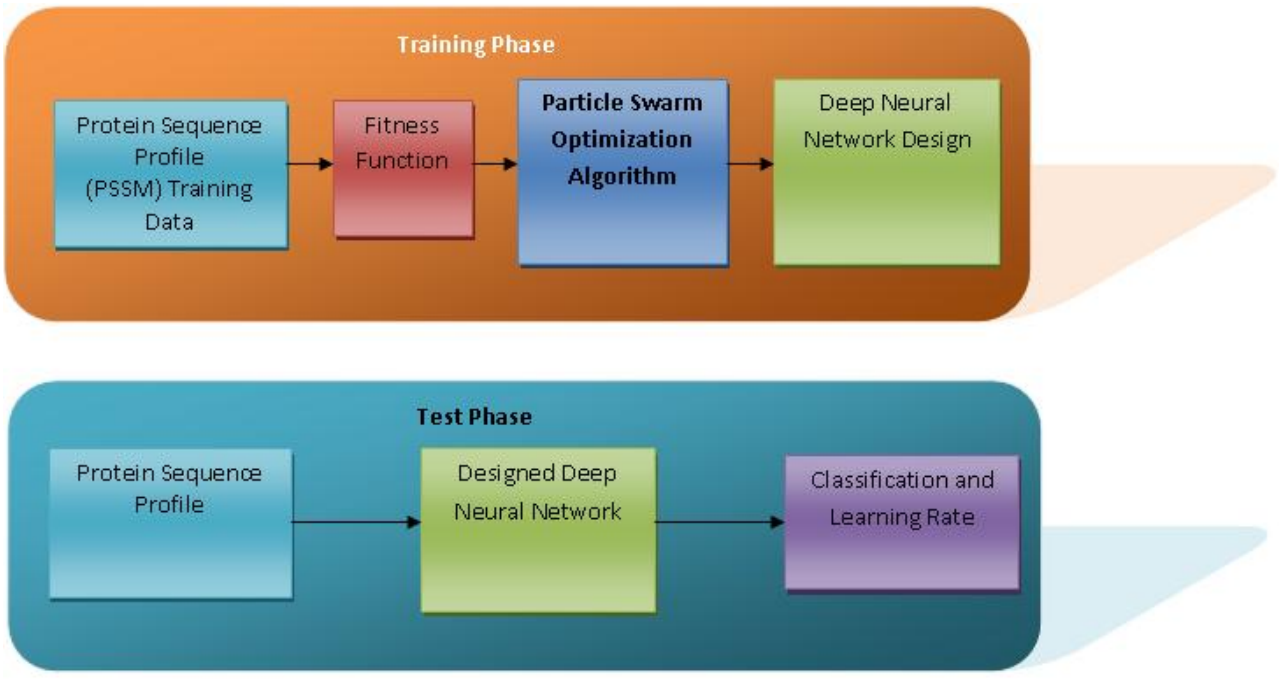
Training Phase
Protein Sequence Profile (PSSM) Training Data
Fitness Function
Particle Swarm Optimization Algorithm
Deep Neural Network Design
Test Phase
Protein Sequence Profile
Designed Deep Neural Network
Classification and Learning Rate

Fig 4: System Model

error in the system as demonstrated in equation (3.1). The smaller the error term, the best the solution predicted.

For this model, the weights and bias of the system are randomly generated at first and then poured into the swarm by assigning this generated weights and bias to the global best position (solution) and the MSE of the random generated weights and bias as the global best error. The swarm then generates various particles (potential solutions) and then train and modifies the global best based on the number of epoch.

3.2.2 Architecture Algorithm

```
1. Randomly assign weights(w) and bias(b) to the network
2. for each neuron in the network
      a. compute the output $o_i$
3. end for
4. Calculate the Error (E) of the system with the present weights using the Mean Squared Error
5. Initialize the Global Best Position (Solution) of the Swarm to the weights and bias generated
6. Initialize the Global Best Error of the Swarm to the Error Calculated
7. Create a swarm with the appropriate number of particles
8. Initialize the epoch
9. Initialize each of the Swarm particles to a random state (position, velocity, error, personal best error, personal best position)
10. Loop until done (until epoch exhausted)
      a. For each particle in the swarm,
            i. Compute new particle velocity
            ii. Use new velocity to compute new position,
            iii. Compute error of the new position
            iv. If new error better than personal best error then
                  1. Set personal best error equals new error
            v. If new error better than global best error
                  1. Set global best error equals to the new error
                  2. Set global best position equals new position
      b. End for
11. End loop
12. Update the weights and bias with the global best position
13. Test the system with the test dataset
```

3.3 PSSP Accuracy Measure

This work uses the Q_3 protein secondary structure performance accuracy measure which is calculated by the percentage of the proportion of the correctly predicted residue to the total residue in a class.

$$Q_3 = \sum \frac{R_{CP}}{R_T} * 100 \quad \text{(3.7)}$$

$R_{CP} = Correctly\ Predicted\ Residue$

$R_T = Total\ Residue\ in\ class$

3.4 Dataset

This work used a total of 1507 protein sequences retrieved from [9]. These data were carefully picked from each protein superfamily defined by the Structural Classification of Proteins (SCOP). This is done to ensure that the bias caused by sequence profile similarities is reduced. Furthermore, sequence with residues lesser than thirty (30) and residues greater than eight hundred (800) was filtered out to aid the system result and time.

Multiple Sequence Alignment (MSA) of the dataset was then generated with Psi-blast using UniRef90 dataset as the reference sequence. This MSA was then used to produce the profile of the dataset which is then used for training and testing of the system.

3.5 Result Presentation and Discussion

With the use of Particle Swarm Optimizer in training a deep neural network, we achieved an accuracy of 53.18% on epoch 180 for Q_3 state prediction. This was due to the early convergence of the PSO at local minimal.

Table 1. Accuracy with Sigmoid and Softmax

Epoch	**20**	**40**	**80**	**100**	**120**	**140**	**160**	**180**	**200**
Accuracy (%)	43.16	41.01	52.04	52.63	52.93	53.18	53.04	53.18	51.94

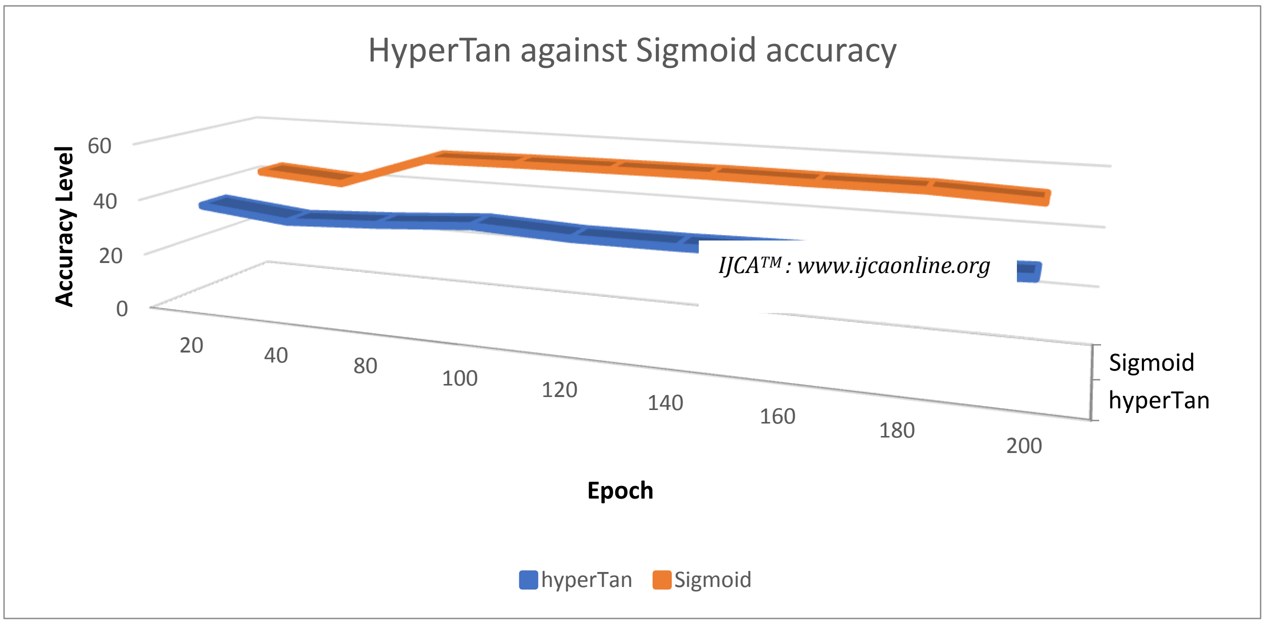

Fig 5: Accuracy of hyperbolic tangent against sigmoid function based on the number of epoch

4. CONCLUSION

This work investigates the use of swarm intelligence particularly the particle swarm optimizer in training a deep neural network for predicting protein secondary structure from its primary structure. It was discovered that the performance of the proposed method didn’t show high prediction accuracy compared to the current state of art due to the early convergence of the optimizer at local minimal.

In future works, it is of interest to investigate the use of the Comprehensive Learning Particle Swarm Optimizer by [10] which considerably minimizes the problem of early convergence.

5. REFERENCES

[1] Bordoloi H &Sarma K. K (2014). Protein Structure Prediction using Artificial Neural Network. Available at: https://www.researchgate.net/publication/252067753_Protein_Structure_Prediction_using_Artificial_Neural_Network. Access on: 4th October, 2017.

[2] Hoijat R, Amin R &Effat D (2016). Cuckoo Search Algorithm and Its Application for Secondary Structure Protein Structure Prediction. Journal of Informatics and Computer Engineering (JICE) Vol. 2(3), pp. 134 – 139. DOI: 649123/220124. Accessed on: 17th September, 2017.

[3] Islam, MdNasrul (2015). A Balanced Secondary Structure Predictor. University of New Orleans Theses and Dissertations. Paper 1995 Available at: http://scholarworks.uno.edu/cgi/viewcontent.cgi?article=3100&context=td. Accessed on: 29th Aug., 2017

[4] Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization, Proceedings of the 1995 IEEE International Conference on Neural Networks, pp. 1942-1948, IEEE Press ISBN: 0-7803-2768-3 doi: 10.1109/ICNN.1995.488968

[5] Wang, S., Peng, J., Ma, J., & Xu, J. (2016). Protein Secondary Structure Prediction Using Deep Convolutional Neural Fields. Scientific Reports, 6, 18962. http://doi.org/10.1038/srep18962. Accessed on: 5th October, 2017

[6] Sonderby S. K &Winther O (2014). Protein Secondary Structure Prediction with Long Short Term Memory Networks. arXiv:1412.7828 [q-bio.QM] Available at: https://arxiv.org/pdf/1412.7828.pdf. Accessed on: 5th October, 2017.

[7] Xiaohui Hu (2010). Particle Swarm Optimization Tutorial. Available: http://www.swarmintelligence.org/tutorials.php. Accessed on: 10th September, 2017

[8] Hamashree B & Kandarpa K. S (2011). Protein Structure Prediction Using Artificial Neural Network. Special Issue of International Journal of Computer Applications (0975 – 8887) on Electronics, Information and Communication Engineering - ICEICE No.3

[9] Drozdetskiy, A., Cole, C., Procter, J. & Barton, G. J. (2015). JPred4: a protein secondary structure prediction server. Nucleic Acids Res., gkv332.

[10] Liang J. J., Qin A. K. & Baskar S. (2006). Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions. IEEE transactions on evolutionary computation, vol. 10, no. 3