A DEEP NEURAL NETWORK APPROACH FOR THE PREDICTION OF PROTEIN SUBCELLULAR LOCALIZATION

Samson A.B.P.[*], *Chandra S.R.A.*[*], *Manikant M.*[*]

Abstract: The subcellular localization of proteins is an essential characteristic of human cells, which plays a vital part in understanding distinct functions and cells' biological processes. The abnormal protein subcellular localization affects protein functionality and may cause many human diseases ranging from metabolic disorders to cancer. Therefore, the prediction of subcellular locations of the proteins is an important task. Artificial neural network has become a popular research topic in machine learning that can achieve remarkable results in learning high-level latent traits. This paper proposes a deep neural network (DNN) model to predict the human protein subcellular locations. The DNN automatically learns high-level representations of abstract features and proteins by examining nonlinear relationships between different subcellular locations. The experimental results have shown that the proposed method gave better results compared with the classical machine learning techniques such as support vector machine and random forest. This model also outperformed the similar model, which uses stacked auto-encoder (SAE) with a softmax classifier.

Key words: *protein subcellular localization, feature representation, classification, machine learning, deep learning*

Received: September 16, 2019 **DOI:** 10.14311/NNW.2021.31.002
Revised and accepted: February 28, 2021

1. Introduction

The protein subcellular localization (PSL) prediction relies heavily on handcrafted feature descriptors to specify proteins. Predicting the subcellular localization of a protein is a critical task in bioinformatics studies, and the protein function relies upon the compartment or organelle in which it is located because it affords a physiological context for its purpose [1, 2]. Drug development, therapeutic target identification, the study of protein function, and biological data research require an understanding of subcellular localization of proteins [3].

*Chandra Sekhara Rao Annavarapu – Corresponding author; Samson Anosh Babu P.; Manikant Mani; Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines) Dhanbad, Jharkand – 826004, India, E-mail: acsrao@iitism.ac.in, samson.enosh.17dr000327@cse.ism.ac.in, mnkntmn6@gmail.com

In the past decade, many researchers have focused on extensive applications of many sequencing analysis techniques. These sequencing technologies generate a large amount of data. This generated data is collected and analyzed with various data analysis pipelines, which are used to identify diseases and perform sample classification between different disease phenotypes for diagnostic and prognostic purposes [4]. In this advanced era, facing such vast numbers of sequences, their PSL experimental conclusions are extraordinarily ineffective and costly. Therefore, fast, flexible, and memory-efficient computational methods are required to approach these problems. In recent years, many computational methods have been proposed for the prediction of PSL. Many of these analysis pipelines consist of tools, which use machine learning algorithms in conjunction with various feature representations to provide predictions [5, 6].

Most of the machine learning techniques for subcellular localization prediction models are classified into two types: (1) sequence-based and (2) annotation-based; The sequence-based models employ the information of the primary protein sequences, and the annotation-based models use most of the homology information from different types of annotation including gene ontology (GO). These terms hold the knowledge of cellular elements and molecular functions of genes [7,8]. Although many machine learning techniques are proposed to improve localization prediction accuracy, feature representation has still become a research challenge, which requires handcrafted designation [9].

In the past few years, deep learning has shown the capability to learn useful feature representations from the input data automatically. Based on depth, artificial neural networks (ANNs) can be broadly classified into two types: shallow networks and deep networks. The networks with more hidden layers are more capable of learning complex nonlinear relationships between input and output [22]. ANNs with more than one hidden layer in their architecture is called deep neural networks (DNNs). The DNNs have multiple layers between the input and output layers, which is highly potential in efficiently describing the linear, nonlinear, and complex relationships [10]. Moreover, deep learning has been applied in many fields of computational biology. For example, in [12, 13], deep learning was applied to predict eukaryotic PSL.

The authors Wen et al. [11] designed a deep learning-based framework for the prediction of drug-target interaction. Further, the authors in [14] used deep neural networks such as LeNet, ALEXNET, and VGG16 for underwater acoustic signal processing. In writings, an embedding-based method is presented for predicting the subcellular localization of proteins [25]. On the other hand, the authors in [26] presented a critical evaluation of web-based prediction tools against the same benchmark data set for human protein subcellular localization. The authors Sitao et al. [27] designed a tunable recognition unit and developed an aptamer-based near-infrared (NIR) light-responsive nanoplatform for manipulating the subcellular localization of specific proteins in their native states. Furthermore, the authors Yijie et al. [28] proposed a fuzzy support vector machine based on kernelized neighborhood representation (FSVM-KNR) to predict the subcellular localization of protein. Moreover, the authors Xin et al. proposed a deep protein subcellular localization predictor, consisting of a linear classifier and a deep feature extractor of convolution neural network (CNN) [29]. In this approach, the deep CNN feature extractor

is first shared and pre-trained in a deep gene ontology annotation predictor, and then is transferred to the subcellular localization predictor with fine-tuning using protein localization samples. In this way, it enhanced the deep protein subcellular localization predictor using transfer learning of GO annotation.

This paper proposes a deep neural network model for predicting and classifying the localization sites of human proteins. The proposed work investigates existing SAE performance, traditional techniques such as SVM and RF, and DNN in context with the PSL dataset. It uses a multilayered neural network, which could overcome few limitations of SAE. Fig. 1 shows a summary of the proposed workflow.

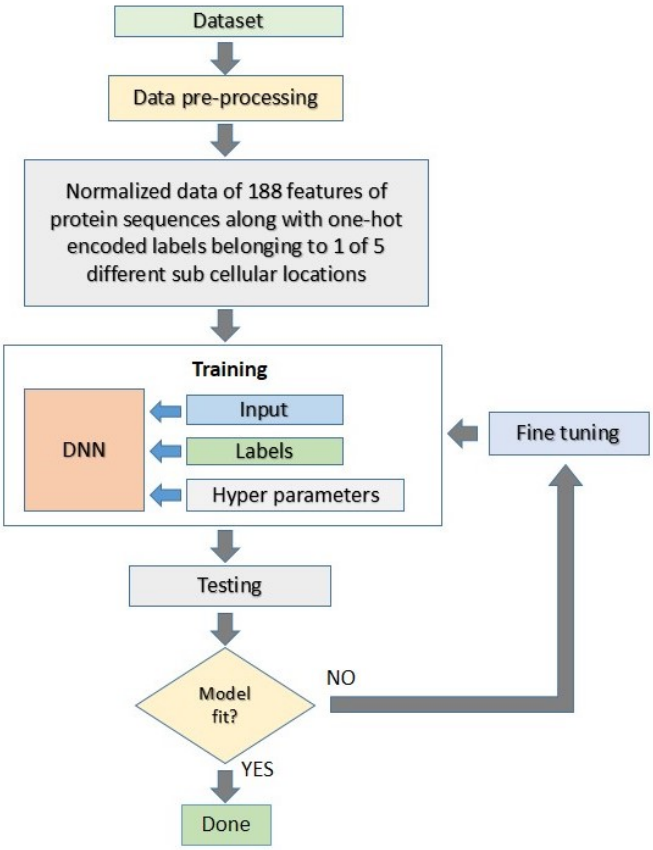

Fig. 1 *Graphical representation of proposed workflow.*

1.1 Roadmap

This paper is organized as follows: Section 2 provides the related work. The dataset preparation and preprocessing details are explained in Section 3. Section 4 describes proposed model. Next, Section 5 provides the experimental results and comparisons. Further, a detailed discussion is given in Section 6. Finally, Section 7 concludes the work.

2. Related work

Auto encoder (AE) [19] is a type of neural network that learns to recreate the input provided to the network. It achieves this by learning the mapping between the input and output fed to the network, where the output of the network is the input itself. The mapping is essentially a layer which *'encodes'* the input in a different dimension. The layers up to the encoding layer are collectively called as an *encoder*, and the layers after the encoding layer up to the output layer are collectively called as a *decoder*. The dimension of the encoding layer of an AE is kept smaller than the input dimension. It is one of the methods for dimensionality reduction. The learning process is almost similar to other neural network models, with the exception that it employs unsupervised learning.

The AE encoder can be used to extract relevant low-level features of input since it has already been trained in an unsupervised manner. This trained encoder can further be used to train a different encoder of another AE. The decoder of AE is only used to train encoders, and they do not take part in the final classification task. The encoding layer of this new AE is of a smaller dimension than the encoding layer of the previous AE. Similarly, a set of encoding layers are derived after training multiple AEs. It is important to note that while training new AE for encoders with encoding dimensions less than the previous AE, the input provided to the new AE is the output of the previous encoder. These encoders are arranged in decreasing order of their encoding dimensions and are fully connected to create a multi-layer neural network. A fully connected softmax layer, which is a classifier with output neurons, is connected to the output of the absolute encoder. The final classifier network is then trained in a supervised manner using a backpropagation method similar to other neural networks. This model is called as SAE because of the way it is created.

3. Methods

This part briefly describes the methods used in the proposed work, such as dataset preparation, preprocessing, and input feature representation.

3.1 Dataset preparation and preprocessing

For the dataset construction, a total of 11,898 human protein sequence data covering about 200 subcellular locations were collected from the famous UniProtKB database[1]. The proposed method utilized the following top five locations after counting the number of proteins at each site: membrane, nucleus, cytoplasm, cell membrane, and secreted. Then, CD-HIT program [15] version-4 was applied to exclude repetitive sequences by fixing the threshold to 0.7, whereas other parameters were set to default. By following the procedure in [12], the primary dataset with 5,780 protein samples was retained. Tab. I describes the pattern details of the protein subcellular locations and the 5 locations marked as five classes.

[1] https://www.uniprot.org/downloads

Location	Class label	#Proteins
Nucleus	0	1504
Cytoplasm	1	1489
Cell membrane	2	753
Membrane	3	1619
Secreted	4	415
Total number of proteins:		5780

Tab. I *Protein dataset details.*

3.2 Input feature representation method

The proposed approach considers a popular feature descriptor based on protein physicochemical properties to obtain features from the protein sequence data as the input for the deep learning model. This robust feature representation captures the following physicochemical properties holding the division of amino acids into three different groups to represent peptide sequences [16]:

1. Hydrophobicity
   - group-1 $\{RKEDQN\}$
   - group-2 $\{GASTPHY\}$
   - group-3 $\{CVLIMFW\}$
2. Normalized van der waals colume
   - group-1 $\{GASCTPD\}$
   - group-2 $\{NVEQIL\}$
   - group-3 $\{MHKFRYW\}$
3. Polarity
   - group-1 $\{LIFWCMVY\}$
   - group-2 $\{PATGS\}$
   - group-3 $\{HQRKNED\}$
4. Polarizability
   - group-1 $\{GASDT\}$
   - group-2 $\{CPNVEQIL\}$
   - group-3 $\{KMHFRYW\}$
5. Charge
   - group-1 $\{KR\}$
   - group-2 $\{ANCQGHILMFPSTWYV\}$

- group-3 $\{DE\}$

6. Surface tension
   - group-1 $\{GQDNAHR\}$
   - group-2 $\{KTSEC\}$
   - group-3 $\{ILMFPWYV\}$
7. Secondary structure
   - group-1 $\{EALMQKRH\}$
   - group-2 $\{VIYCWFT\}$
   - group-3 $\{GNPSD\}$
8. Solvent accessibility
   - group-1 $\{ALFCGIVW\}$
   - group-2 $\{RKQEND\}$
   - group-3 $\{MPSTHY\}$

Using these properties, we elicited a collection of 188 useful features that represent the protein sequence based on amino acids (AAs) distribution with certain physicochemical properties [17, 18]. For each property, we assumed $S = P_1, P_2, P_3, P_4, \ldots, P_N$ represents a protein sequence, where P_i is the amino acid in position i, and N denotes the number of amino acids, (i.e., sequence length). The 20 amino acids can be expressed as follows.

$$AA = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}. \tag{1}$$

In the first step, the respective quantities of 20 AAs such as $AA_1, AA_2, AA_3, \ldots, AA_{20}$ were calculated as $m_1, m_2, m_3, \ldots, m_{19}, m_{20}$. By using these values to represent a specific protein sequence, the feature vector $(FV)(1-20)$ is denoted as:

$$(FV_1, FV_2, FV_3, \ldots, FV_{20}) = \left(\frac{m_1}{N}, \frac{m_2}{N}, \ldots, \frac{m_{20}}{N}\right), \tag{2}$$

where N represents the sequence length, m_i $(i = 1, 2, 3, 4, \ldots, 20)$ represents the quantity of an AA in the protein sequence, and $\sum_{i=1}^{20} FV_i = 1$.

Further, the peptide sequence is encoded by considering the AA composition of the content(C), distribution (D), and bivalent frequency (F) of AAs as three aspects of descriptors for each physicochemical property. The AAs descriptors were used to describe the protein properties. For example, by considering hydrophobicity (H) property:

- The AAs were classified as $RKEDQN$, $GASTPHY$, and $CVLIMFW$ classes. By utilizing the size of 3 classes (CH_1, CH_2, and CH_3), the calculated $FV(21-23)$ as follows:

$$(FV_{21}, FV_{22}, FV_{23}) = \left(\frac{CH_1}{N}, \frac{CH_2}{N}, \frac{CH_3}{N}\right). \tag{3}$$

- Then, the chain length is calculated as $DH_{ij}(i = 1,2,3; j = 1,2,3,4,5)$, wherein the first, 25 %, 50 %, 75 %, and 100 % of AAs chain belongs to a specific property were located, respectively. Then the defined $FV(24-38)$ was denoted as follows:

$$(FV_{24},\ldots,FV_{28};FV_{29},\ldots,FV_{33};FV_{34},\ldots,FV_{38}) = \left(\frac{DH_{11}}{N},\ldots,\frac{DH_{15}}{N};\frac{DH_{21}}{N},\ldots,\frac{DH_{25}}{N};\frac{DH_{31}}{N},\ldots,\frac{DH_{35}}{N}\right). \tag{4}$$

- In the final step, the number of bivalent seeds were represented as $(N-1)$. Then counted the number of respective bivalent seeds that contain 2 AAs from various classes and obtained the defined FH_1, FH_2, and FH_3 parameters as follows:

$$(FV_{39},FV_{40},FV_{41}) = \left(\frac{FH_1}{N-1},\frac{FH_2}{N-1},\frac{FH_3}{N-1}\right). \tag{5}$$

All these 21 feature vectors were calculated for each property and finally extracted all 188 feature vectors (i.e., $20 + (21 * 8) = 188$) after completing the complete analysis of properties.

4. Proposed method

The proposed method comprises of three stages: 1. Model creation, 2. Model training, and 3. Model testing.

4.1 Model creation

The proposed model has ten layers, including a 1-input layer, 8-hidden layers, and 1-output layer. Fig. 2 shows the architecture of the proposed model. The input layer has a dimension of 188 for the normalized float values of a sample from the dataset processed during the data preprocessing step. All eight layers are fully connected layers with dimensions of 256, 512, 1024, 2048, 512, 128, 64, and 32. These hidden layers also employ dropout as a regularization method, and the dropout probability used is 0.15. These layers use ReLU (Rectified Linear Unit) [23] as an activation function. The output layer is a fully-connected layer with five output neurons representing 5 class labels. It uses the softmax activation function as given in Eq. 6 to normalize the outputs of these five neurons and therefore give 5 class probabilities.

$$\sigma(x)_j = \frac{\mathbf{e}^{x_j}}{\sum_{n=1}^{N}\mathbf{e}^{x_n}}, \tag{6}$$

where x is the input vector to the output layer. j is the index ranging from 1 to N.

The categorical cross-entropy loss function was used for the calculation of loss in classification tasks. The method used for model optimization is Adam (Adaptive moment estimation) [24]. Adam is one of the latest algorithms in the family

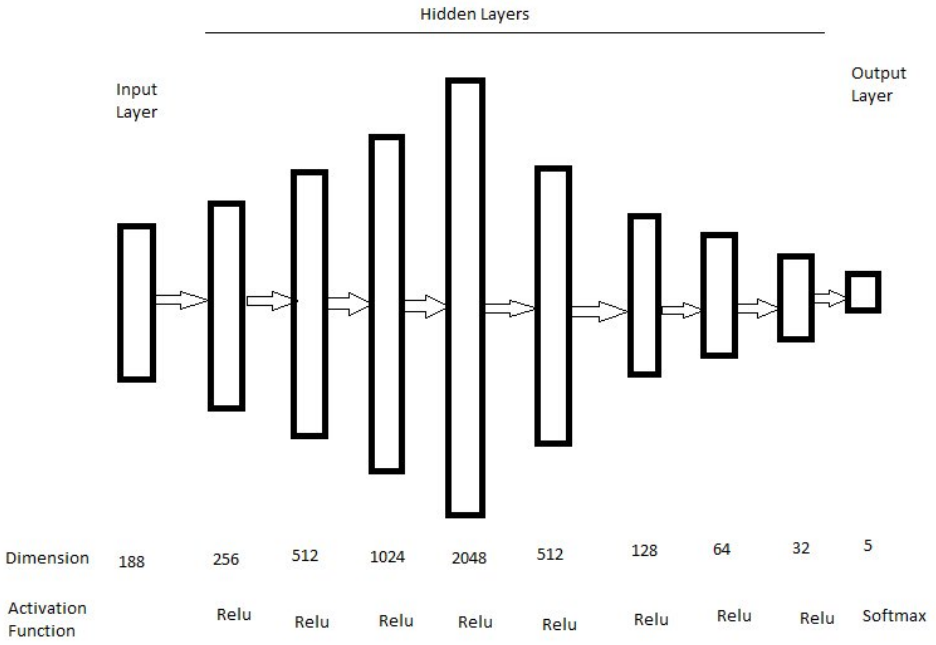

Fig. 2 *Architecture of the proposed deep neural network model.*

of optimizers for model training. It combines two powerful optimizers, namely: RMSProp (Root Mean Square Propagation) and AdaGrad (Adaptive Gradient). Unlike other optimizers, Adam uses a different learning rate for every parameter in the network and then adjusts it along with the parameter as training proceeds. The proposed model summary with a layerwise number of parameters is described in Fig. 3.

4.2 Model training

The dataset is divided into two sets after random shuffling, with 80 % samples belonging to the training set and 20 % samples belonging to the test set. The corresponding training labels were converted to 5-dimensional vectors using the one-hot encoding method. The model was trained using the training set for 250 epochs having a batch size of 500 samples.

4.3 Model testing

Testing is necessary for measuring the classification accuracy on a set of testing data. The test set consisting of 20 % of the data was used to test the model. The model was also tested with another reliable performance measure, such as the k-fold cross-validation method.

Layer (type)	Output Shape	Param #
dense_55 (Dense)	(None, 256)	48384
dropout_43 (Dropout)	(None, 256)	0
dense_56 (Dense)	(None, 512)	131584
dropout_44 (Dropout)	(None, 512)	0
dense_57 (Dense)	(None, 1024)	525312
dropout_45 (Dropout)	(None, 1024)	0
dense_58 (Dense)	(None, 2048)	2099200
dropout_46 (Dropout)	(None, 2048)	0
dense_59 (Dense)	(None, 512)	1049088
dropout_47 (Dropout)	(None, 512)	0
dense_60 (Dense)	(None, 128)	65664
dropout_48 (Dropout)	(None, 128)	0
dense_61 (Dense)	(None, 64)	8256
dropout_49 (Dropout)	(None, 64)	0
dense_62 (Dense)	(None, 32)	2080
dense_63 (Dense)	(None, 5)	165

Total params: 3,929,733
Trainable params: 3,929,733
Non-trainable params: 0

Fig. 3 *The proposed model summary with layer wise number of parameters.*

5. Results

The proposed model is implemented in Python version 3, using Keras with TensorFlow backend and Scikit Learn. The average results were taken after executing the models ten times. The obtained proposed model's training and testing accuracies are 49.58 % and 45.24 %. Furthermore, k-fold cross-validation with $k = 3$ and $k = 5$ were used to evaluate the model. Mean 3-fold cross-validation accuracy of the model was 45.78 % with a standard deviation of 0.44 and mean 5-fold cross-validation accuracy of the model was 45.76 % with a standard deviation of 0.51.

The other model used to compare the proposed model's performance on the dataset is the SAE classifier used in DeepPSL [12]. The proposed model showed better results when compared with DeepPSL. This model had a training and testing accuracy of 45.13 % and 44.64 %, respectively. The mean k-fold ($k = 3$) validation accuracy came out to be 45.73 % with a standard deviation of 0.85, and the mean k-fold ($k = 5$) validation accuracy came out to be 45.74 % with a standard deviation of 0.44.

The proposed model was also compared with two other traditional machine learning methods. At first, random forest (RF) with bootstrapping and a maximum depth of six were used to perform the dataset classification. The accuracy obtained for the training set was 12.59 %, and for the testing set was 10.47 %. The obtained mean k-fold ($k = 3$) validation accuracy was 9.72 % with a standard de-

viation of 0.33, and the mean k-fold ($k = 5$) validation accuracy was 9.83 % with a standard deviation of 0.31. The other method used was the support vector machine (SVM) with RBF (radial basis function) kernel. Its accuracy for the training set is 33.41 %, and for the testing set is 36.07 %. The mean k-fold ($k = 3$) validation accuracy is 29.07 % with a standard deviation of 0.07, and the mean 5-fold cross-validation accuracy is 31.31 % with a standard deviation of 0.45. Tab. II shows the classification results summary, and Fig. 4 visualizes the comparison of results. The performance of the classifier models was evaluated after getting the true positive (TP), false positive (FP), true negative (TN), and false negative (FN) values. The formulae to calculate the performance measures are as follows:

$$Precision = \frac{TP}{TP + FP}, \tag{7}$$

$$NPV = \frac{TN}{TN + FN}, \tag{8}$$

$$Recall = \frac{TP}{TP + FN}, \tag{9}$$

$$Specificity = \frac{TN}{TN + FP}, \tag{10}$$

$$FPR = \frac{FP}{TN + FP}, \tag{11}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{12}$$

$$f1 - score = 2 * \frac{precision * recall}{precision + recall}. \tag{13}$$

Tab. III depicts the confusion matrix of the test data for the proposed model. In Tabs. IV, V and VI, the confusion matrices, precision, recall, f1-score, and support of the test data for SAE, SVM, and RF models were displayed, respectively. Support represents the number of samples present in test data. Moreover, the area under curve (AUC) is also calculated, which is the trade-off between sensitivity and specificity and provides a valuable parameter for comparing test performance. In Tab. VII the details of sensitivity, specificity, positive predictive value (PPV), negative predictive value (NPV), false positive rate (FPR), and AUC of classifier models for a single test run were displayed.

6. Discussion

The SAE method was applied in DeepPSL [12] for the classification of PSL with ten classes. For a comparison between models, this method is applied to our dataset with 5 classes. The number of AEs is not mentioned in [12]. However, for our dataset, which has 188 input dimensions, two AEs were trained successively to derive two encoders. Moreover, dropout [20] was used as regularizer in all the

Method (Total # samples:5780)	Avg. training acc. (%) (# samples:4624)	Avg. testing acc. (%) (# samples:1156)	k-fold ($k = 3$) acc. (%)	k-fold ($k = 5$) acc. (%)
Proposed method (DNN)	49.58	45.24	45.78±0.44	45.76±0.51
Stacked autoencoder (Deep PSL)	45.13	44.64	45.73±0.85	45.74±0.44
Random Forest	12.59	10.47	9.72±0.33	9.83±0.31
SVM	33.41	36.07	29.07±0.07	31.31±0.45

Tab. II *Results summary of classification accuracy.*

Location	Class label	TN	FP	FN	TP	Precision	Recall	f1-score	Support
Nucleus	0	748	109	174	125	0.53	0.42	0.47	299
Cytoplasm	1	568	294	119	175	0.37	0.60	0.46	294
Cell membrane	2	1005	0	151	0	0.0	0.0	0.0	151
Membrane	3	623	199	146	188	0.49	0.56	0.52	334
Secreted	4	1044	34	46	32	0.48	0.41	0.44	78
Average (or) Weighted Average (or) Total:		797.6	127.2	127.2	104	0.41	0.45	0.42	1156

Tab. III *Confusion matrix of the test data for proposed model.*

Location	Class label	TN	FP	FN	TP	Precision	Recall	f1-score	Support
Nucleus	0	670	194	114	178	0.48	0.61	0.54	292
Cytoplasm	1	670	202	160	124	0.38	0.44	0.41	284
Cell membrane	2	1012	0	144	0	0.0	0.0	0.0	144
Membrane	3	575	233	136	212	0.48	0.61	0.53	348
Secreted	4	1057	11	86	2	0.15	0.02	0.04	88
Average (or) Weighted Average (or) Total:		796.8	128	128	103.2	0.37	0.45	0.40	1156

Tab. IV *Confusion matrix of the test data for SAE classifier model.*

Location	Class label	TN	FP	FN	TP	Precision	Recall	f1-score	Support
Nucleus	0	748	116	194	98	0.46	0.34	0.39	292
Cytoplasm	1	872	0	284	0	0.0	0.0	0.0	284
Cell membrane	2	1012	0	144	0	0.0	0.0	0.0	144
Membrane	3	185	623	29	319	0.34	0.92	0.49	348
Secreted	4	1068	0	88	0	0.0	0.0	0.0	88
Average (or) Weighted Average (or) Total:		777	147.8	147.8	83.4	0.22	0.36	0.25	1156

Tab. V *Confusion matrix of the test data for SVM classifier model.*

Location	Class label	TN	FP	FN	TP	Precision	Recall	f1-score	Support
Nucleus	0	117	747	2	290	0.28	0.99	0.44	292
Cytoplasm	1	872	0	284	0	0.0	0.0	0.0	284
Cell membrane	2	1012	0	144	0	0.0	0.0	0.0	144
Membrane	3	760	48	280	68	0.59	0.2	0.29	348
Secreted	4	1068	0	85	3	1	0.03	0.07	88
Average (or) Weighted Average (or) Total:		765.8	159	159	72.2	0.32	0.31	0.20	1156

Tab. VI *Confusion matrix of the test data for Random Forest classifier model.*

Method	Sensitivity	Specificity	PPV	NPV	FPR	AUC
Proposed method	0.4498	0.8625	0.4498	0.8625	0.1375	0.6561
Stacked autoencoder (Deep PSL)	0.4463	0.8615	0.4463	0.8615	0.1384	0.6539
Random Forest	0.3123	0.8281	0.3123	0.8281	0.1719	0.5702
SVM	0.3607	0.8402	0.3607	0.8402	0.1598	0.6005

Tab. VII *Performance measures of classifier models.*

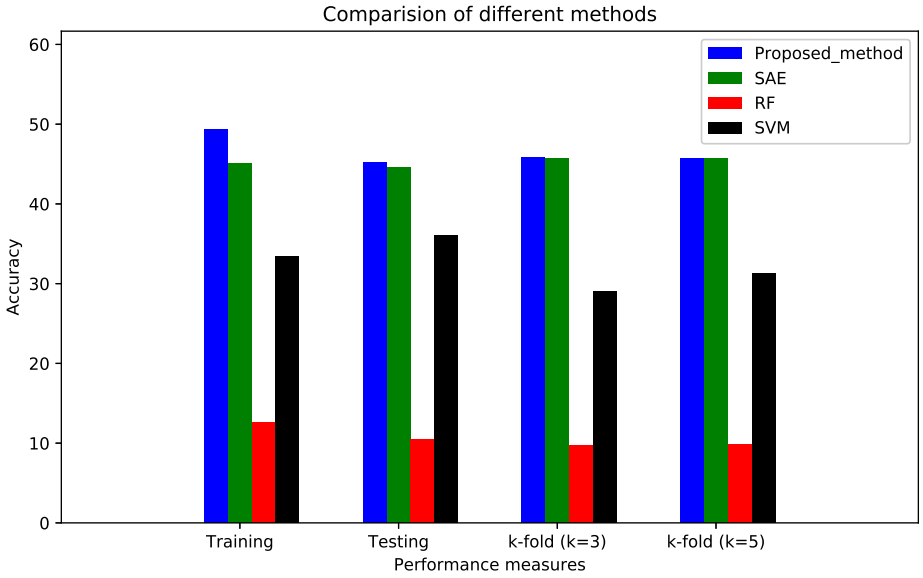

Fig. 4 *Comparision of results using different performance measures.*

layers constructed in the SAE. We used 0.15 as dropout probability. The encoders were then connected, and a fully-connected softmax output layer was connected to the output of the second encoder. This final layer was trained to create an SAE classifier with multiple layers. The whole training process visualized in Fig. 5.

The DeepPSL model used SAE as a feature extraction method, which is combined with a classifier at the last layer to perform classification. The number of

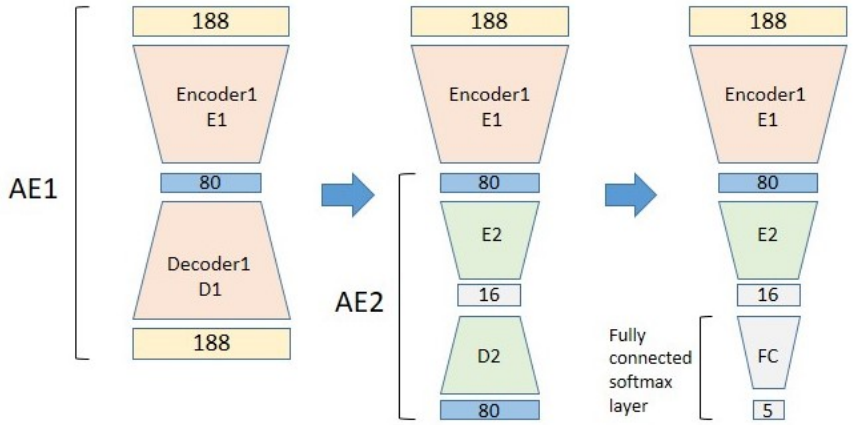

Fig. 5 *Training process of Stacked Auto Encoder using two Auto Encoders.*

hidden layers constructed with SAE is equal to the number of AEs used in constructing the model. In our SAE implementation, we used two AEs (i.e., two hidden layers) for this purpose. However, the layers were individually trained. The proposed model has eight hidden layers, and that can learn more complex input-output relationships provided the number of epochs, batch size, and other hyperparameters is appropriately set. As shown in Fig. 3, the convergence of the proposed model is comparatively slow and requires many parameters (weights).

The data used in the proposed work has data points with 188-dimension size. However, the paper [12] uses a 588-dimension input size and has a number of instances= 13978. Therefore, the SAE used in the proposed work is undoubtedly different from the one used by the authors of the paper [12]. Furthermore, the proposed work dataset has five classes compared to 10 classes in the paper [12]. However, the dataset used should not affect the generalizability of the model used. It established that the proposed work model for simulating DeepPSL (SAE) is different from the SAE used in the proposed work. The dimensions of the SAE model used in the proposed work as follows. An AE of a single layer with a latent dimension of 80 and input/output dimension of 188 is used using the process as discussed before to create the first layer of the model. The next layer of dimension 16 is created similarly, followed by the softmax layer of dimension 5. Number of parameters (weights and bias) are as follows:

- The network has $(188 * 80 + 80 * 16 + 16 * 5)$ weights + $(80 + 16 + 5)$ bias = 16501 parameters.

Moreover, the proposed work investigates the performance of SAE, traditional techniques such as SVM and RF, and Deep Neural Networks in context with the PSL dataset. The results of the experiments provide clarity on their classification performance. Upon comparing the results, the following two limitations of SAE are found, which affects its performance:

- The layer-wise pre-training of the SAE is done in an unsupervised manner, and therefore label of any given instance in the dataset is not used for feature learning.
- The learned features are indiscriminative, and it affects the classification accuracy.

The proposed work uses a multilayered neural network that is devoid of the limitations mentioned above of SAE.

7. Conclusion

This paper proposed a DNN model that automatically learns high-level representations of abstract input features and predicts the human protein subcellular locations. The proposed method extracts 188 features for 5,780 proteins and attempted to classify them into five different categories. The proposed method is evaluated, and the results are compared with the DeepPSL model and traditional machine learning techniques: SVM and RF. The proposed method's performance showed better results due to its usage of more layers and less complexity than the existing DeepPSL model, which uses SAE with a softmax classifier.

References

[1] EMANUELSSON O., BRUNAK S., VON HEIJNE G., NIELSEN H. Locating proteins in the cell using TargetP, SignalP and related tools. Nature protocols, 2007, 2(4), 953.

[2] IMAI K., NAKAI K. Prediction of subcellular locations of proteins: where to proceed?. Proteomics, 2010, 10(22), pp. 3970–3983.

[3] EISENHABER F., BORK P. Wanted: subcellular localization of proteins based on sequence. Trends in cell biology, 1998, 4(8), pp. 169–170.

[4] MOROZOVA O., MARRS M.A. Applications of next-generation sequencing technologies in functional genomics. Genomics, 2008, 92(5), pp. 255–264.

[5] JHA A., KHARE A., SINGH R. Features' compendium for machine learning in NGS data Analysis, 2015.

[6] WEI L., LIAO M., GAO Y., JI R., HE Z., ZOU Q. Improved and promising identification of human microRNAs by incorporating a high-quality negative set. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 2014, 11(1), pp. 192–201.

[7] PARK K.J., KANEHISA M. Prediction of protein subcellular locations by support vector machines using compositions of amino acids and amino acid pairs. Bioinformatics, 2003, 19(13), pp. 1656–1663.

[8] WAN S., MAK M.W., KUNG S.Y. HybridGO-Loc: Mining hybrid features on gene ontology for predicting subcellular localization of multi-location proteins. PLoS One, 2014, 9(3), e89545.

[9] WEI L., XING P., SU R., SHI G., MA Z.S., ZOU Q. CPPred-RF: a sequence-based predictor for identifying cell-penetrating peptides and their uptake efficiency. Journal of Proteome Research, 2017, 16(5), pp. 204-4-2053.

[10] LECUN Y., BENGIO Y., HINTON G. Deep learning. nature, 2015, 521(7553), p. 436.

[11] WEN M., ZHANG Z., NIU S., SHA H., YANG R., YUN Y., LU H. Deep-learning-based drug–target interaction prediction. Journal of proteome research, 2017, 16(4), pp. 1401–1409.

[12] WEI L., DING Y., SU R., TANG J., ZOU Q. Prediction of human protein subcellular localization using deep learning. Journal of Parallel and Distributed Computing, 2018, 117, pp. 212–217.

[13] ALMAGRO ARMENTEROS J.J., SØNDERBY C.K., SØNDERBY S.K., NIELSEN H., WINTHER O. DeepLoc: prediction of protein subcellular localization using deep learning. Bioinformatics, 2017, 33(21), pp. 3387–3395.

[14] WU H., SONG Q., JIN G. Underwater acoustic signal analysis: Preprocessing and classification by deep learning. Neural Network World, 2020, 30(2), pp. 85–96.

[15] HUANG Y., NIU B., GAO Y., FU L., LI W. CD-HIT Suite: a web server for clustering and comparing biological sequences. Bioinformatics, 2010, 26(5), pp. 680–682.

[16] LIN C., ZOU Y., QIN J., LIU X., JIANG Y., KE C., ZOU Q. Hierarchical classification of protein folds using a novel ensemble classifier. PloS one, 2013, 8(2), e56499.

[17] ZOU Q., WANG Z., GUAN X., LIU B., WU Y., LIN Z. An approach for identifying cytokines based on a novel ensemble classifier. BioMed research international, 2013.

[18] CAI Y.D., LIU X.J., XU X.B., CHOU K.C. Prediction of protein structural classes by support vector machines. Computers & chemistry, 2002, 26(3), pp. 293–296.

[19] BALDI P. Autoencoders, unsupervised learning, and deep architectures. In Proceedings of ICML workshop on unsupervised and transfer learning, 2012, pp. 37–49.

[20] SRIVASTAVA N., HINTON G., KRIZHEVSKY A., SUTSKEVER I., SALAKHUTDINOV R. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 2014, 15(1), pp. 1929–1958.

[21] KRIZHEVSKY A., SUTSKEVER I., HINTON G.E. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, 2012, pp. 1097–1105.

[22] SUN S., CHEN W., WANG L., LIU X., LIU T.Y. On the depth of deep neural networks: A theoretical view. In Thirtieth AAAI Conference on Artificial Intelligence, 2016.

[23] GLOROT X., BORDES A., BENGIO Y. Deep sparse rectifier neural networks. In Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, pp. 315–323.

[24] KINGMA D.P., BA J. Adam: A method for stochastic optimization. arXiv preprint arXiv: 1412.6980, 2014.

[25] PAN X., LI H., ZENG T., LI Z., CHEN L., HUANG T., CAI Y.D. Identification of protein subcellular localization with network and functional embeddings. Frontiers in Genetics, 2021, 11, p. 1800.

[26] SHEN Y., DING Y., TANG J., ZOU Q., GUO F. Critical evaluation of web-based prediction tools for human protein subcellular localization. Briefings in bioinformatics, 2020, 21(5), pp. 1628–1640.

[27] XIE S., DU Y., ZHANG Y., WANG Z., ZHANG D., HE L., TAN W. Aptamer-based optical manipulation of protein subcellular localization in cells. Nature communications, 2020, 11(1), pp. 1–10.

[28] DING Y., TANG J., GUO F. Human protein subcellular localization identification via fuzzy model on kernelized neighborhood representation. Applied Soft Computing, 2020, 96, 106596.

[29] YUAN X., PANG E., LIN K., HU J. Deep Protein Subcellular Localization Predictor Enhanced with Transfer Learning of GO Annotation. IEEJ Transactions on Electrical and Electronic Engineering, 2021.