A Neural Network-Based Multi-Label Classifier for Protein Function Prediction

Shahab Tahzeeb
Dept. of Computer & Information Systems Engineering
NED University of Engineering & Technology
Karachi, Pakistan
stahzeeb@neduet.edu.pk

Shehzad Hasan
Dept. of Computer & Information Systems Engineering
NED University of Engineering & Technology
Karachi, Pakistan
shasan@neduet.edu.pk

Abstract-Knowledge of the functions of proteins plays a vital role in gaining a deep insight into many biological studies. However, wet lab determination of protein function is prohibitively laborious, time-consuming, and costly. These challenges have created opportunities for automated prediction of protein functions, and many computational techniques have been explored. These techniques entail excessive computational resources and turnaround times. The current study compares the performance of various neural networks on predicting protein function. These networks were trained and tested on a large dataset of reviewed protein entries from nine bacterial phyla, obtained from the Universal Protein Resource Knowledgebase (UniProtKB). Each protein instance was associated with multiple terms of the molecular function of Gene Ontology (GO), making the problem a multilabel classification one. The results in this dataset showed the superior performance of single-layer neural networks having a modest number of neurons. Moreover, a useful set of features that can be deployed for efficient protein function prediction was discovered.

Keywords-gene ontology; molecular function term; multi-label classification; neural network; protein function prediction

I. INTRODUCTION

Understanding proteins' functions plays a vital role in acquiring insights of the molecular mechanisms operating in both physiological and ailing medical conditions. As a result, this understanding substantiates the discovery of drugs in different diseases [1]. However, predicting protein functions is an arduous task. The fact is markedly implied by the incredibly large number of unannotated protein entries hosted by the most comprehensive protein database, the Universal Protein Resource Knowledgebase (UniProtKB) [2]. This is mainly due to the reliance on traditional experimental annotation techniques carried out by molecular biologists. The gap between reviewed and unreviewed protein sequences is widening due to the data deluge from high-throughput state-of-the-art sequencing techniques [1, 3-5]. The pressing demands for computational methods on the functional annotation of proteins have paved the way for significant contributions by computer science researchers. Many computational techniques employing machine learning for functional annotation of proteins have been utilized in the literature. The principal difference between various approaches lies in the set of features pursued by different investigators. This section presents a brief summary of some of the most prominent efforts in this area.

An ensemble of Deep Neural Networks (DNNs) was proposed in [1], where each DNN worked on a different set of features from the dataset. The predictions of different DNNs were then voted to arrive at the final protein function prediction. A DNN for the hierarchical multilabel classification of protein functions designed to perform well even with a limited number of training samples was presented in [3]. In [4], a DNN was introduced to learn features from word embedding of protein sequences, based on the concept of Natural Language Processing (NLP), using sequence similarity profiles as additional features to locate proteins. Authors in [5] established the efficacy of exploiting any interrelationships among different functional terms. For instance, different functional classes were found to coexist with some proteins suggesting a mutual relationship. Furthermore, a quantification model of these relations was proposed, using a functional similarity measure and a framework to capitalize on it for the eventual prediction of protein functions. A classification technique based on a neural network coupled with a Support Vector Machine (SVM) was demonstrated in [6], utilizing a bi-directional Long Short-Term Memory (LSTM) network to generate fixed-length protein vectors out of variable-sized sequences and deal with the challenges posed by the variable length of protein sequences. In [7], protein sequence motifs were used to build a deep convolutional network and predict protein function, while the authors claimed to have built the best performing model for the cellular component classes. The significance of Protein-Protein Interaction (PPI) and time-course gene expression data as powerful predictors for the prediction of protein function was shown in [8]. A method, called Dynamic Weighted Interactome Network (DWIN), was proposed, that in addition to PPI and gene expression data, took also into account information related to protein domains and complexes to improve the prediction performance. In [9], clustering was applied on a PPI network for the prediction of protein function. A protein graph model was shown in [10], constructed of protein structure, with each node representing a cluster of amino acid residues. However, the idea of using an accuracy metric for evaluation is generally misleading. In [11], an active learning approach was explored for the prediction of

Corresponding author: Shahab Tahzeeb

protein function using a PPI network. This method operated in two phases: Spectral clustering was used to cluster the PPI network followed by the application of the betweenness centrality measure for labeling within each cluster, and then the labeled protein data were used by a classification algorithm. Associations between functions in a PPI network were used in [12], stating that multiple function labels assigned to proteins were not independent and their coexistence could be used effectively to predict protein function. A deep semantic text representation was presented in [13], with various pieces of information extracted from protein sequences such as homology, motifs, and domains. Protein function prediction was carried out using a consensus between text-based and sequence-based methods. In [14], a classifier using cumulative iterations was proposed, based on its semantic similarity with the term Gene Ontology (GO). Each prediction was followed by updating and optimizing scores of characteristic terms in the set of GO annotations, which, in turn, led to improved future predictions. The dissimilarity of protein functions, rather than conventional similarity measures, was used in [15] to segregate rare and frequently occurring classes of functions. This technique worked well for imbalanced datasets.

The notable contributions cited above are just a handful of numerous praiseworthy efforts towards the prediction of protein function. These endeavors differ in terms of the protein information utilized by the corresponding systems and the computational or time complexities of the classification models. The current paper presents a neural network-based multi-label classifier for the prediction of protein function by training and testing several neural networks on a large dataset [16]. The results indicate that a neural network with a single hidden layer achieved remarkable prediction performance with nominal computational complexity. This makes its implementation viable on systems with modest hardware capabilities. Consequently, the time required for the classification task is in the order of seconds.

II. MATERIALS AND METHODS

A. Dataset

The dataset adopted from [16] includes 121,378 protein instances. These labeled protein examples were extracted from UniProtKB [2], a comprehensive worldwide repository of protein information. These protein entries pertain to 9 bacterial phyla, namely Actinobacteria, Bacteroidetes, Chlamydiae, Cyanobacteria, Firmicutes, Fusobacteria, Proteobacteria, Spirochaetes, and Tenericutes. Each instance in the dataset had 9,890 features. These features included the sequence of amino acids making up the corresponding protein, compositions of amino acids, dipeptides and tripeptides; compositions of five groups of amino acids, i.e. aliphatic, aromatic, positively charged, negatively charged, uncharged, and various structural and physiochemical properties derived from the amino acid sequence. In addition, some features quantify conjoint triads. A conjoint triad is a unit of three successive amino acids such that each amino acid in the unit belongs to one of the seven groups formed on the basis of the dipole and volume scale [17]. These characteristic values indicate the strength of interaction between the amino acids of these 7 groups. The feature set also contains pseudo amino acid compositions for the corresponding protein. As suggested in [18], these numbers overcome the loss of sequence order effect in a protein caused by considering just plain amino acid compositions. Moreover, there are also 541 motifs included in the features. These are small segments in proteins' tertiary structure that are frequently found in different proteins. These similar patterns are associated with the structural or functional roles of proteins.

There are 1,739 binary labels associated with each protein instance. These labels correspond to GO terms belonging to the Molecular Function (MF) category. The GO is a categorization of biological functions using three broad classes, i.e. Molecular Function (MF), Cellular Component (CC), and Biological Process (BP), generally referred to as GO terms [19]. The molecular function term specifies a biochemical activity performed by a gene product, without taking into account the time and space dimensions of this activity. The enzyme is an example of the MF term. The CC refers to the location of the biochemical activity of a gene product in the cell. Ribosome and nuclear membrane are two such examples. BP, an all-encompassing term, defines a biological objective to which activities of various gene products contribute. Cell growth and maintenance serve as examples the BP term.

B. Data Preprocessing

The Comma Separated Values (CSV) files for 9 different bacterial phyla were combined to obtain a single Pandas' data frame object using the Pandas data analysis library in Python [20]. Duplicate rows were removed from the data frame, which was then converted to an array using the scientific computing library NumPy in Python [21]. The feature values were then scaled using the standard scaler available in the scikit-learn library in Python [22]. Data scaling was investigated using normalization and robust scaler, but these data scaling techniques proved inferior to the standard scaling technique.

C. Features Partitioning

The neural networks were trained on 3 sets of features. The objective of partitioning features into various subsets was to test the hypothesis that compositions of amino acids, dipeptides, and tripeptides are sufficient to predict protein functions. $F = \{F_1, F_2, F_3\}$ represented the set of features used to train different models, where F_1 was the entire set of 9,890 features, and F_2 was the set of 8,420 features that contained only compositions of amino acids, dipeptides, and tripeptides. The set $F_3=F_1–F_2$ contained 1,470 features consisting of various properties and characteristics derived from proteins as described in subsection A.

D. Neural Networks

A variety of neural networks was selected, differing in the number of hidden layers and neurons in each layer to train the protein function classification system on datasets corresponding to each feature set F_1, F_2, and F_3. The experimental results are given in Section III. It was observed that the simplest neural network containing a single hidden layer demonstrated better performance on this dataset compared to neural networks having more hidden layers. The optimal number of neurons in this single hidden layer was experimentally determined to be just 5% of the total input and output neurons for feature sets F_1 and F_2 for the best performing neural network. However, for the F_3 feature set, the optimal number of neurons in the single hidden layer of the best performing neural network turned out to be 50% of the total of input and output neurons.

Once the optimal number of neurons in a single hidden layer was determined, the addition of another hidden layer was utilized to observe any potential boost in performance. The number of neurons in the second hidden layer was chosen to be 50% of the first hidden layer. This was done to ensure that the network captured the most important features for prediction. Table I summarizes various single-hidden layer neural networks trained and tested on the F_1 feature set, i.e. the entire set of features from the dataset. The reference computer for all time and memory size measurements presented is a Core i7-8700 at 3.2 GHz 6-core processor.

TABLE I. SINGLE-HIDDEN LAYER MODELS TRAINED ON THE F_1 FEATURE SET

Model	Neurons[1]	Size (MB)	Training time (sec)	Predictiontime (sec)
M1	1	16	1950	2.25
M2	5	77	2655	6.03
M3	10	154	3920	7.51
M4	15	232	4860	9.75
M5	25	386	8820	13.6
M6	50	773	12175	21.8
M7	75	1130	15400	31.6

1. Expressed as percentage of input + output neurons

Table II presents the M8 neural network with two hidden layers trained on F_1. This model was constructed by adding another hidden layer to the best performing single-hidden layer neural network M2 to explore any performance gain. The second hidden layer had 50% neurons of the first hidden layer in an attempt to capture the optimal features best suited for the prediction task.

TABLE II. TWO-LAYER MODEL TRAINED ON THE F_1 FEATURE SET

Model	Size (MB)	Training time (sec)	Prediction time (sec)
M8	74	2120	5.52

Table III summarizes various single-hidden layer neural networks trained and tested on the F_2 feature set, i.e. the compositions of amino acids, dipeptides, and tripeptides in the protein sequence.

TABLE III. SINGLE-HIDDEN LAYER MODELS TRAINED ON THE F_2 FEATURE SET

Model	Neurons[1]	Size (MB)	Training time (sec)	Prediction time (sec)
M9	5	60	2760	5
M10	10	118	3480	6.31
M11	25	295	7920	11.1
M12	50	590	15000	17.8
M13	60	708	15855	20.4

1. Expressed as a percentage of input + output neurons

Table IV presents the M14 neural network containing two hidden layers and trained on F_2. This model was developed by adding another hidden layer to the best performing single hidden layer neural network M9, to investigate any improvements in the classifier performance. The second hidden layer had 50% neurons of the first hidden layer to exploit the predictors best suited for the prediction task.

TABLE IV. TWO-LAYER MODEL TRAINED ON THE F_2 FEATURE SET

Model	Size (MB)	Training time (sec)	Prediction time (sec)
M14	56	2050	4.76

Table V summarizes several single-hidden layer neural networks trained and tested on the F_3 feature set, i.e. features consisting of various properties and characteristics derived from proteins.

TABLE V. SINGLE-HIDDEN LAYER MODELS TRAINED ON F_3 FEATURE SET

Model	Neurons[1]	Size (MB)	Training time (sec)	Predictiontime (sec)
M15	5	6	2750	1.33
M16	10	12	3200	1.58
M17	25	30	5200	2.92
M18	30	35	5270	3.05
M19	50	60	6630	4.20
M20	60	71	7320	4.59

1. Expressed as a percentage of input + output neurons

Table VI presents the M21 neural network having two hidden layers and trained on F_3. This model was generated by adding another hidden layer to the best performing single hidden layer neural network M19 to discover any potential performance enhancement. The number of neurons in the second hidden layer was chosen to be 50% of the first hidden layer to capitalize on the features best suited for the classification.

TABLE VI. TWO-LAYER MODEL TRAINED ON THE F_3 FEATURE SET

Model	Size (MB)	Training time (sec)	Prediction time (sec)
M21	58	5720	4.64

For each network, we employed the *relu* activation for the hidden layers, the sigmoid activation for the output layer, the *he_uniform* kernel initializer for the hidden layers, and the Adaptive moment estimation (Adam) optimizer with a learning rate of 0.00001.

E. Performance Evaluation

Since a protein example in this dataset can be mapped to more than one binary label, the prediction of protein function is a multilabel classification problem. The dataset is also highly imbalanced due to the overwhelming number of negative examples for each label. Evaluation of such a classification model cannot simply rely on the accuracy of prediction [23, 24]. For example, if a negative class is abundantly prevalent among all examples in an imbalanced dataset, then a naive classifier predicting this class for all examples will easily achieve very high accuracy. The challenge of this inflated accuracy measure becomes aggravated in the case of multilabel classification of imbalanced datasets. This problem was addressed by defining more meaningful performance measures, namely *precision*, *recall*, *F1 score*, *zero-one loss*, *hamming loss*, and *Matthews Correlation Coefficient*. These measures are defined here.

1) Precision

Precision is defined as the fraction of positively classified instances that are, in effect, positive. This gives a clear picture of a classifier's strength in predicting positive classes. Letting *TP* and *FP* respectively denote the count of true and false positives, *Precision* is calculated as:

$$Precision = \frac{TP}{TP+FP} \quad (1)$$

The precision of *predict-majority-class-for-all* classifier is thus 0 judiciously penalizing for its shortcoming at predicting the positive minority class. However, any classifier that makes just one positive prediction and ensures its correctness would have 100% precision despite its failure to predict other positive examples. This calls for another classification metric, called *Recall*, also known as *sensitivity*.

2) Recall

Recall is defined as the fraction of positive examples in the dataset classified as positive. Letting *FN* denote the number of false negatives, the *Recall* is given by:

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

This measure penalizes a classifier that attempts to achieve high precision simply by making a few correct positive predictions.

3) F1 Score

Precision and recall are combined in a single performance measure called *F1 score*, which is their harmonic mean.

$$F1\,score = 2 * \frac{Precision*Recall}{Precision+Recall} \quad (3)$$

As the harmonic mean is biased towards lower values, *F1 score* can have a higher value only in the case when both precision and recall have high values. In multi-label classification, there are several ways to average the aforementioned performance metrics on all labels [25, 26]. These are the *micro* average, *macro* average, *weighted* average, and *samples* average as defined below. As usual, the *F1 score* is the harmonic mean of the corresponding precision and recall in each case.

4) Micro Average

This is calculated by counting the number of True Positives (TPs) across the entire set of target labels. If there are *N* samples in the dataset and each sample has *L* binary target labels, then the micro averages of *Precision* and *Recall* are calculated as:

$$Precision_{micro} = \frac{\sum_{i=1}^{N*L}(Y\_pred_i \wedge Y\_true_i)}{\sum_{i=1}^{N*L} Y\_pred_i} \quad (4)$$

$$Recall_{micro} = \frac{\sum_{i=1}^{N*L}(Y\_pred_i \wedge Y\_true_i)}{\sum_{i=1}^{N*L} Y\_true_i} \quad (5)$$

where *Y_pred* and *Y_true* are the predicted and actual target labels, respectively. The conjunction operator ∧ ensures the inclusion of only those label instances that are positive in both *Y_pred* and *Y_true*, i.e. TPs.

5) Macro Average

This averages the *Precision* and *Recall* scores of the individual target labels, giving equal weights to all of them.

$$Precision_{macro} = \frac{1}{L}\sum_{j=1}^{L}\frac{\sum_{i=1}^{N}(Y\_pred_{ij} \wedge Y\_true_{ij})}{\sum_{i=1}^{N} Y\_pred_{ij}} \quad (6)$$

$$Recall_{macro} = \frac{1}{L}\sum_{j=1}^{L}\frac{\sum_{i=1}^{N}(Y\_pred_{ij} \wedge Y\_true_{ij})}{\sum_{i=1}^{N} Y\_true_{ij}} \quad (7)$$

6) Weighted Average

This averages the *Precision* and *Recall* scores of the individual target labels, using the number of positive instances of each label in the set *Y_true* as their weight.

$$Precision_{weighted} = \frac{1}{\sum_{j=1}^{L} w_j}\sum_{j=1}^{L}\frac{\sum_{i=1}^{N}(Y\_pred_{ij} \wedge Y\_true_{ij})*w_j}{\sum_{i=1}^{N} Y\_pred_{ij}} \quad (8)$$

$$Recall_{weighted} = \frac{1}{\sum_{j=1}^{L} w_j}\sum_{j=1}^{L}\frac{\sum_{i=1}^{N}(Y\_pred_{ij} \wedge Y\_true_{ij})*w_j}{\sum_{i=1}^{N} Y\_true_{ij}} \quad (9)$$

where w_j denotes the weight, also known as the support of the *j-th* label.

7) Samples Average

It averages the *Precision* and *Recall* scores across the samples.

$$Precision_{samples} = \frac{1}{N}\sum_{i=1}^{N}\frac{\sum_{j=1}^{L}(Y\_pred_{ij} \wedge Y\_true_{ij})}{\sum_{j=1}^{L} Y\_pred_{ij}} \quad (10)$$

$$Recall_{samples} = \frac{1}{N}\sum_{i=1}^{N}\frac{\sum_{j=1}^{L}(Y\_pred_{ij} \wedge Y\_true_{ij})}{\sum_{j=1}^{L} Y\_true_{ij}} \quad (11)$$

This is the most faithful as well as the most conservative performance indicator of the multi-label classifier as it reflects, on the average, how well the classifier performed on each sample. Therefore, the sample averages were used to gauge the performance of the models.

8) Zero-One Loss

For a multi-label classification problem, this measure credits a prediction as correctly classified only when all labels are correctly classified. The loss is zero for a correct prediction. However, if the classifier fails to make a correct prediction even for just one target label, the corresponding loss is 1. It follows that the *zero-one* loss is truly a conservative and highly penalizing performance measure.

$$Loss_{0/1} = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{L}(Y\_pred_{ij} \oplus Y_{true\,ij}) \quad (12)$$

The combination of the product operator Π and the exclusive-OR operator ⊕ ensures that any mismatch between *L* predicted and target labels generates a loss of 1 for any given sample. Otherwise, the loss is zero for a complete match between all predicted and target labels for a given sample.

9) *Hamming Loss*

This gives the fraction of all incorrectly predicted labels by quantifying the number of incorrect predictions of all labels rather than penalizing individual examples. Hence, if a multi-label classifier incorrectly predicts 1 out of 10 labels for a given instance, the hamming loss for that example is just 1/10 as compared to 1 in the case of zero-one loss. It follows that hamming loss is lenient compared to the stringent zero-one loss.

$$HL = \frac{1}{N*L}\sum_{i=1}^{NL}(Y\_pred_i \oplus Y\_true_i) \quad (13)$$

10) *Matthews Correlation Coefficient*

The Matthews Correlation Coefficient (MCC) is a binary version of Pearson's correlation coefficient [27]. However, multiclass classification problems can also benefit from its extended version [28]. MCC compares ground truth and predicted vectors, considering all possibilities of prediction, i.e. True Positives (TP), True Negatives (TN), False Positives (FP), and False negatives (FN). Therefore, it gives a balanced evaluation of the performance of the classifier. The correlation coefficient lies in the range [-1, +1], with -1 for false prediction, 0 for random prediction, and +1 for correct prediction.

$$MCC = \frac{TP*TN-FP*FN}{\sqrt{(TP+TN)(TP+FN)(FP+TN)(FP+FN)}} \quad (14)$$

The MCC was calculated for every example and its average was used to assess the performance of the classifier on the entire dataset.

11) *Consolidated Performance Metric*

For the sake of an all-encompassing and more realistic comparison of performance, the aforementioned metrics were combined in a single Consolidated Performance Metric (CPM) as follows:

$$CPM = \frac{F1\,score*MCC}{Loss_{0/1}*HL} \quad (15)$$

The CPM was constructed in the higher, the better way.

III. RESULTS

Table VII shows the performance details of the neural networks (M1 to M21) as the samples averages of *Precision* (P), *Recall* (R), *F1 score* (F1), *Zero-One Loss* (ZOL), *Hamming Loss* (HL), MCC, and CPM.

A wide variety of neural networks was trained and tested on a large dataset of proteins. For feature sets F_1 and F_2, it was observed that 5% of the total input and output neurons in the single hidden layer networks exhibited better prediction performance than other single-layer models. These models were designated as M2 and M9, respectively. However, for the feature set F_3, the optimal count of neurons in the hidden layer emerged as 50% of the total input and output neurons. This model was designated as M19. The bar graphs in Figures 1-3 compare the CPM for various neural networks that work on a specific feature set. In each case, the best performing single-layer network was extended by adding a second hidden layer to assess any performance edge. Neurons in the second hidden layer were the 50% of the first hidden layer to ensure that the most relevant features for prediction play their due role. The blue bars in Figures 1-3 represent the performance of the best performing single-layer networks, while yellow bars show the performance of 2-layer neural networks.

TABLE VII. PERFORMANCE OF NEURAL NETWORKS

Model	P	R	F1	ZOL	HL	MCC	CPM
M1	0.96	0.95	0.95	12.25	0.0159	0.9518	4.6423
M2	0.96	0.96	0.96	11.36	0.0120	0.9576	6.7437
M3	0.96	0.96	0.95	11.64	0.0122	0.9571	6.4028
M4	0.96	0.96	0.95	11.76	0.0122	0.9566	6.3341
M5	0.96	0.96	0.95	11.90	0.0123	0.9568	6.2100
M6	0.96	0.96	0.95	12.55	0.0129	0.9546	5.6016
M7	0.96	0.96	0.95	12.61	0.0132	0.9539	5.4442
M8	0.96	0.95	0.95	12.21	0.0128	0.9535	5.7959
M9	0.94	0.94	0.93	15.85	0.0173	0.9341	3.1681
M10	0.94	0.94	0.93	16.47	0.0179	0.9329	2.9429
M11	0.93	0.94	0.93	16.70	0.0180	0.9318	2.8828
M12	0.93	0.94	0.93	17.30	0.0186	0.9298	2.6873
M13	0.93	0.93	0.93	17.07	0.0183	0.9297	2.7678
M14	0.93	0.93	0.92	17.17	0.0188	0.9279	2.6446
M15	0.96	0.95	0.95	12.43	0.0129	0.9535	5.6492
M16	0.96	0.96	0.96	11.34	0.0117	0.9591	6.9396
M17	0.97	0.96	0.96	10.43	0.0108	0.9614	8.1935
M18	0.97	0.96	0.96	10.66	0.0110	0.9614	7.8709
M19	0.97	0.96	0.96	10.34	0.0107	0.9625	8.3516
M20	0.97	0.96	0.96	10.39	0.0108	0.9621	8.2310
M21	0.97	0.96	0.96	10.95	0.0114	0.9593	7.3775

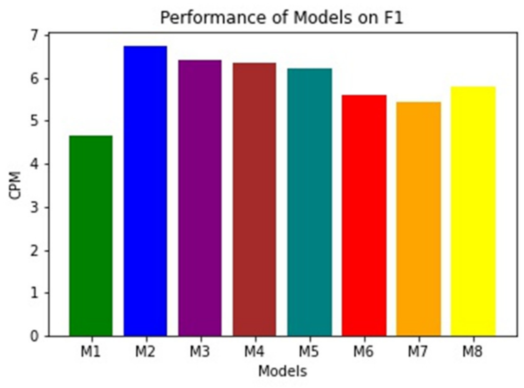

Fig. 1. Neural networks' comparison on F_1 feature set.

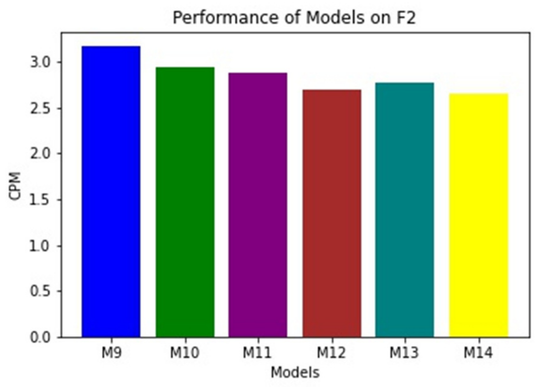

Fig. 2. Neural networks' comparison on F_2 feature set.

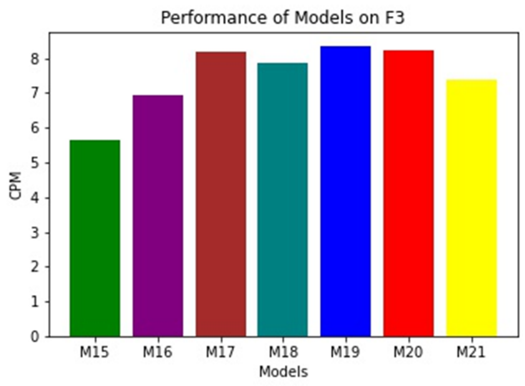

Fig. 3. Neural networks's comparison on F_3 feature set.

F. Optimal Feature Set

The experiments also focused on exploring an optimal set of features for the prediction of protein function. As it was noticed, the F_3 feature set proved to be the best predictor for this multi-label classification. Figure 4 shows a comparison of the best-performing models for each feature set, where M19 on F_3 achieved the best performance.

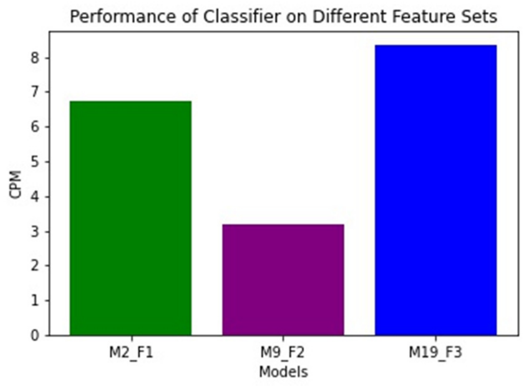

Fig. 4. Performance comparison of the best performing single-layer models on different feature sets. F_3 proves to be the best predictor set.

G. Classification Threshold

The impact of the classification threshold on the performance of a classifier was examined. The models predict the probability of each target label associated with every instance. These probability values quantify the chance for a given instance to belong to a particular class. These probability values should be translated into binary labels 0 and 1 before the final evaluation of the model. This conversion to binary labels required a threshold or probability cutoff value below which all values are classified as class 0 and equal or greater values are classified as class 1. Classifier performance metrics are profoundly influenced by the choice of this threshold. The impact of the threshold is more pronounced for imbalanced datasets. As the examined dataset is skewed towards more negative examples of each label, the performance of the models was evaluated for various values of thresholds. Figures 5 and 6 show two example plots of samples averages of P, R, and F1 score for models M2 and M19, respectively, against a range of classification thresholds.

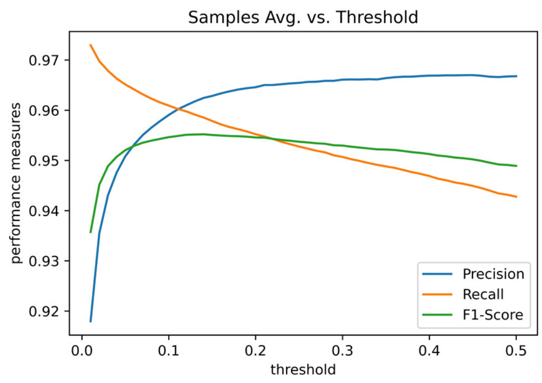

Fig. 5. Performance curves of M2 for the F_1 feature set. The choice of classification threshold plays an important role in improving the performance of the classifier.

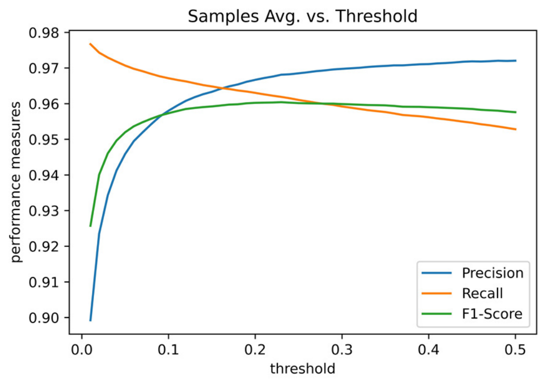

Fig. 6. The Performance curves of model M19 for the F_3 feature set.

H. Confusion Matrix

The confusion matrix is a visualization of a classifier's performance, as it gives the count of TP, FP, TN, and FN class predictions.

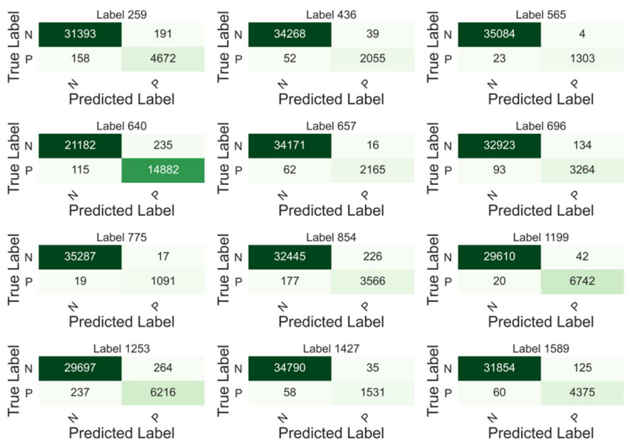

Fig. 7. Confusion matrices of the best-performing model M19 for F_3. Only labels having support more than 1,000 are included.

Figure 7 presents the confusion matrices of select labels in the test dataset for the model M19 that performed best in the F_3 feature set. To highlight the strength of the classifier, only labels whose support exceeded 1,000 in the dataset were chosen

IV. DISCUSSION

The performance comparison of different neural networks on a large protein dataset showed that neural networks having a single hidden layer and a modest number of neurons achieved superior performance on this specific dataset than relatively more complex networks. The number of neurons in the single hidden layer was empirically determined. The rigorous experimentation revealed that only 5% of the total input and output neurons were adequate in single-hidden-layer models operating on F_1 and F_2 feature sets. However, this count was 50% for a single-layer model on F_3. This disparity in the number of neurons was because these models for the F_1 and F_2 feature sets had 9,890 and 8,420 neurons, respectively, in their input layers. Therefore, even 5% of the total input and output neurons were adequate to effectively train the model for 1,739 labels. However, for F_3, the number of input neurons was barely 1,470, and consequently, more neurons were needed for better prediction performance. This justifies a 50% count of neurons in the single hidden layer for this network. In any case, the training time, prediction time, and model size of these models were much better than those of other competing models. These models showed much better performance (F1 score: 0.96) compared to the deep learning ensemble [1] (F1 score: 0.79) on the same dataset. This was also achieved with a much lower computational complexity.

A. Best Predictors

The findings highlight the impressive role of the physiochemical properties and motifs in proteins, pseudo amino acid compositions, and other properties derived from the protein sequences in predicting protein functions. The proposed model for this feature set was extremely efficient as it had better performance and lower computational complexity.

B. Sufficiency of Amino Acid, Dipeptide, and Tripeptide Compositions

The results were suggestive of the sufficiency of amino acid, dipeptide, and tripeptide compositions in predicting protein functions. Although the performance metrics for this particular feature set had lower values than other feature sets, it can be used for a sufficient and tolerable approximation. This could save time spent in engineering features from existing features of the dataset consisting of bare compositions of amino acids, dipeptides, and tripeptides.

V. CONCLUSIONS

This study culminated in two significant findings regarding the examined protein dataset. The first one pertains to the exceptional performance of single-layer neural networks on this dataset, alhough the number of neurons in this single hidden layer must be empirically determined as a percentage of the total input and output neurons in the network. The simple design of this single-layer model requires minimal computing resources. This model showed a performance improvement of more than 16% over two-layer neural networks operating on the F_1 feature set. The corresponding performance improvements for the F_2 and F_3 features set were 20% and 13%, respectively. This study could play a substantial role in the prediction of protein function, due to the tremendous predictive power of some physiochemical properties of proteins, their pseudo-amino acid compositions, motifs in proteins, and some other significant characteristics. The bare compositions of amino acids, dipeptides, and tripeptides provide a reasonably high level of approximation of protein functions. This could be useful in cases where researchers want to have an approximate idea of protein functions just from the amino acid sequence rather than extracting and relying on many other properties of proteins.

REFERENCES

[1] S. Mishra, Y. P. Rastogi, S. Jabin, P. Kaur, M. Amir, and S. Khatun, "A deep learning ensemble for function prediction of hypothetical proteins from pathogenic bacterial species," *Computational Biology and Chemistry*, vol. 83, Dec. 2019, Art. no. 107147, https://doi.org/10.1016/j.compbiolchem.2019.107147.

[2] "UniProtKB - UniProt Knowledgebase." https://www.uniprot.org/help/uniprotkb (accessed Dec. 04, 2021).

[3] X. Yuan, W. Li, K. Lin, and J. Hu, "A Deep Neural Network Based Hierarchical Multi-Label Classifier for Protein Function Prediction," in *2019 International Conference on Computer, Information and Telecommunication Systems (CITS)*, Aug. 2019, pp. 1–5, https://doi.org/10.1109/CITS.2019.8862034.

[4] Z. Du, Y. He, J. Li, and V. N. Uversky, "DeepAdd: Protein function prediction from k-mer embedding and additional features," *Computational Biology and Chemistry*, vol. 89, Dec. 2020, Art. no. 107379, https://doi.org/10.1016/j.compbiolchem.2020.107379.

[5] X. F. Zhang and D. Q. Dai, "A Framework for Incorporating Functional Interrelationships into Protein Function Prediction Algorithms," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 3, pp. 740–753, May 2012, https://doi.org/10.1109/TCBB.2011.148.

[6] A. Ranjan, M. S. Fahad, D. Fernández-Baca, A. Deepak, and S. Tripathi, "Deep Robust Framework for Protein Function Prediction Using Variable-Length Protein Sequences," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 17, no. 5, pp. 1648–1659, Sep. 2020, https://doi.org/10.1109/TCBB.2019.2911609.

[7] M. Kulmanov and R. Hoehndorf, "DeepGOPlus: improved protein function prediction from sequence," *Bioinformatics*, vol. 36, no. 2, pp. 422–429, Jan. 2020, https://doi.org/10.1093/bioinformatics/btz595.

[8] B. Zhao *et al.*, "A New Method for Predicting Protein Functions From Dynamic Weighted Interactome Networks," *IEEE Transactions on NanoBioscience*, vol. 15, no. 2, pp. 131–139, Mar. 2016, https://doi.org/10.1109/TNB.2016.2536161.

[9] M. Modi, N. G. Jadeja, and K. Zala, "FMFinder: A Functional Module Detector for PPI Networks," *Engineering, Technology & Applied Science Research*, vol. 7, no. 5, pp. 2022–2025, Oct. 2017, https://doi.org/10.48084/etasr.1347.

[10] M. A. Alvarez and C. Yan, "A new protein graph model for function prediction," *Computational Biology and Chemistry*, vol. 37, pp. 6–10, Apr. 2012, https://doi.org/10.1016/j.compbiolchem.2012.01.003.

[11] W. Xiong, L. Xie, S. Zhou, and J. Guan, "Active learning for protein function prediction in protein–protein interaction networks," *Neurocomputing*, vol. 145, pp. 44–52, Dec. 2014, https://doi.org/10.1016/j.neucom.2014.05.075.

[12] P. Sun *et al.*, "Protein Function Prediction Using Function Associations in Protein–Protein Interaction Network," *IEEE Access*, vol. 6, pp. 30892–30902, 2018, https://doi.org/10.1109/ACCESS.2018.2806478.

[13] R. You, X. Huang, and S. Zhu, "DeepText2GO: Improving large-scale protein function prediction with deep semantic text representation,"

Methods, vol. 145, pp. 82–90, Aug. 2018, https://doi.org/10.1016/j.ymeth.2018.05.026.

[14] K. Taha, P. D. Yoo, and M. Alzaabi, "iPFPi: A System for Improving Protein Function Prediction through Cumulative Iterations," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 12, no. 4, pp. 825–836, Jul. 2015, https://doi.org/10.1109/TCBB.2014.2344681.

[15] M. Frasca and N. C. Bianchi, "Multitask Protein Function Prediction through Task Dissimilarity," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, no. 5, pp. 1550–1560, Sep. 2019, https://doi.org/10.1109/TCBB.2017.2684127.

[16] S. Mishra, Y. P. Rastogi, S. Jabin, P. Kaur, M. Amir, and S. Khatoon, "A bacterial phyla dataset for protein function prediction," *Data in Brief*, vol. 28, Feb. 2020, Art. no. 105002, https://doi.org/10.1016/j.dib.2019.105002.

[17] J. Shen *et al.*, "Predicting protein–protein interactions based only on sequences information," *Proceedings of the National Academy of Sciences*, vol. 104, no. 11, pp. 4337–4341, Mar. 2007, https://doi.org/10.1073/pnas.0607879104.

[18] K. C. Chou, "Prediction of protein cellular attributes using pseudo-amino acid composition," *Proteins: Structure, Function, and Bioinformatics*, vol. 43, no. 3, pp. 246–255, 2001, https://doi.org/10.1002/prot.1035.

[19] M. Ashburner *et al.*, "Gene Ontology: tool for the unification of biology," *Nature Genetics*, vol. 25, no. 1, pp. 25–29, May 2000, https://doi.org/10.1038/75556.

[20] "pandas - Python Data Analysis Library." https://pandas.pydata.org/ (accessed Dec. 04, 2021).

[21] "NumPy - The fundamental package for scientific computing with Python." https://numpy.org/ (accessed Dec. 04, 2021).

[22] "scikit-learn: machine learning in Python — scikit-learn 1.0.1 documentation." https://scikit-learn.org/stable/ (accessed Dec. 04, 2021).

[23] D. Virmani, N. Jain, A. Srivastav, M. Mittal, and S. Mittal, "An Enhanced Binary Classifier Incorporating Weighted Scores," *Engineering, Technology & Applied Science Research*, vol. 8, no. 2, pp. 2853–2858, Apr. 2018, https://doi.org/10.48084/etasr.1962.

[24] M. Alghobiri, "A Comparative Analysis of Classification Algorithms on Diverse Datasets," *Engineering, Technology & Applied Science Research*, vol. 8, no. 2, pp. 2790–2795, Apr. 2018, https://doi.org/10.48084/etasr.1952.

[25] X. Z. Wu and Z. H. Zhou, "A Unified View of Multi-Label Performance Measures," in *Proceedings of the 34th International Conference on Machine Learning*, Jul. 2017, pp. 3780–3788, Accessed: Dec. 04, 2021. [Online]. Available: https://proceedings.mlr.press/v70/wu17a.html.

[26] T. Li, C. Zhang, and S. Zhu, "Empirical Studies on Multi-label Classification," in *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, Nov. 2006, pp. 86–92, https://doi.org/10.1109/ICTAI.2006.55.

[27] J. Gorodkin, "Comparing two K-category assignments by a K-category correlation coefficient," *Computational Biology and Chemistry*, vol. 28, no. 5, pp. 367–374, Dec. 2004, https://doi.org/10.1016/j.compbiolchem.2004.09.006.

[28] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, May 2000, https://doi.org/10.1093/bioinformatics/16.5.412.