A Semi-Supervised Autoencoder-Based Approach for Protein Function Prediction

Richa Dhanuka, Anushree Tripathi, and Jyoti P. Singh, *Senior Member, IEEE*

***Abstract*—After the development of next-generation sequencing techniques, protein sequences are abundantly available. Determining the functional characteristics of these proteins is costly and time-consuming. The gap between the number of protein sequences and their corresponding functions is continuously increasing. Advanced machine-learning methods have stepped up to fill this gap. In this work, an advanced deep-learning-based approach is proposed for protein function prediction using protein sequences. A set of autoencoders is trained in a semi-supervised manner with protein sequences. Each autoencoder corresponds to a single protein function only. In particular, 932 autoencoders corresponding to 932 biological processes and 585 autoencoders corresponding to 585 molecular functions are trained separately. Reconstruction losses of each protein sample for every autoencoder are used as a feature to classify these sequences into their corresponding functions. The proposed model is tested on test protein samples and achieves promising results. This method can be easily extended to predict any number of functions having an ample amount of supporting protein sequences. All relevant codes, data and trained models are available at https://github.com/richadhanuka/PFP-Autoencoders.**

***Index Terms*—Autoencoder, bioinformatics, machine learning, multi-label classification, protein function.**

I. Introduction

PROTEINS are responsible for performing many critical biological functions such as metabolism, cell growth, and cell differentiation, which makes proteins an essential component of life. However, as per the UniProt [1], [2] data, only 0.5% of all the known proteins have well-characterized functions. Hence, there is a lot of scope to unearth the unknown functions of protein. The enhanced ability to predict the protein functions accurately can potentially speed up research in the field of drug discovery and personalized medicine, leading to better human health.

Protein Function Prediction (PFP) is a method of assigning functions to proteins. There are approximately 28758 biological processes (BP) terms, 11148 molecular functions (MF) terms, and 4185 cellular components (CC) terms [3] as categorised by the Gene Ontology (GO) Consortium[1] [4], [5]. The initial traditional approaches for PFP are manual and experimental, and which mainly target specific proteins [6]. The traditional approaches are low throughput and require huge human effort as most of the experiments are conducted in laboratories. With the advent of next-generation sequencing techniques, more biological sequences are more readily available, and has thus become cumbersome to annotate the proteins using traditional approaches [7]. The increased gap between unannotated and annotated proteins has opened up space to develop computational approaches for solving the problems of PFP. Computational approaches are used to suggest relevant functions for the newly discovered proteins which biologists can later validate through experimental methods. These suggestions reduce the search space for biologists to experiment with, which in turn increases the efficiency of experimental methods.

Various computational methods have been developed so far to predict functions of proteins [8]–[13]. The foremost approach is homology-based transfer, in which new proteins are annotated with functional annotations of similar proteins in the database [14], [15]. The Basic Local Alignment Search Tool (BLAST) [8] is one such sequence alignment tool. BLAST gained popularity and many variations of the original algorithm were built with modifications like PSI-BLAST and Gapped BLAST [9]. All these tools are based on string matching algorithms with a common drawback of being complex and computationally expensive. In [11], the authors presented an alignment-free approach for comparing sequences which showed that a sequence can be represented in the form of numerical vectors, which created a huge space for feature engineering in PFP. In their work, they achieved comparable results to the traditional alignment-based approaches.

Recently, the use of deep-learning techniques has grown in the field of protein function prediction, since these techniques can learn from a broad range of data [16]–[22]. Most machine-learning techniques in PFP use features generated from protein sequences to train a model and predict protein functions [23]–[29]. As protein sequences are considered analogous to human languages except that human languages have known semantics [30], many natural language processing (NLP) techniques have been developed to embed protein sequences into numerical features. Du *et al.* [31] have used NLP based k-mer word embeddings to predict protein functions. They have utilized the Word2Vec [32] method to define the set of features

Manuscript received 25 November 2021; revised 22 February 2022; accepted 23 March 2022. Date of publication 29 March 2022; date of current version 5 October 2022. *(Corresponding author: Richa Dhanuka.)*

The authors are with the National Institute of Technology Patna, Patna 800005, India (e-mail: richa.cse17@nitp.ac.in; anushree.cs@nitp.ac.in; jps@nitp.ac.in).

Digital Object Identifier 10.1109/JBHI.2022.3163150

[1]http://www.geneontology.org/ as of Jan, 2021

for protein sequences. In [33], researchers have proposed that simple encodings from NLP like one-hot encoding and random embeddings that do not need any pre-training achieve similar performance when compared to pre-trained embeddings of protein sequences. Villegas-Morcillo *et al.* [34] utilized the plethora of unannotated proteins to create embeddings. They proposed an unsupervised protein embedding in which proteins with unknown functions are used to create the features and are then fed to the supervised model to predict functions. They outperformed the performances of various hand-crafted features including 3D structural information, using a simple two-layer perceptron model. Wan and Jones [29] proposed a novel work called FFPred-GAN, a data augmentation technique to generate high-quality synthetic protein samples that can accurately learn the high-dimensional distributions of protein sequence-based features using generative adversarial networks (GAN) and improve the prediction accuracy of protein functions. They ran tests on various classifiers like support vector machines, random forests and k-nearest neighbours showing an improvement in the predictive performance in all the tested classifiers. They integrated their proposed model with NetGO [35] to achieve maximum predictive performance. Feature extraction in other proteomic data like protein-protein interaction data are also being used [36]–[38]. In [36], the authors proposed a deep network fusion model to extract features from heterogeneous protein networks. They have used multi-modal deep autoencoders in which separate layers were for different networks, and later connected the layers together into a single bottleneck layer. It integrates various heterogeneous protein interaction networks into a low-dimensional feature representation common to all networks. Fan *et al.* [39] proposed a multi-modal graph-based architecture, Graph2GO, based on feedforward neural network model. They have used various types of protein data like protein network data (protein-protein interaction data and sequence similarity networks), protein sequence information, protein subcellular location, and protein domains to predict protein functions. The integration of these heterogeneous protein data enhanced the learning capability of the model.

Apart from feature extraction techniques, many more works using deep-learning models such as convolution neural networks (CNNs), recurrent neural networks (RNNs), autoencoders, etc. have been proposed [21], [35], [40]–[43]. In one such study [44] researchers built a language translation model using a recurrent neural network where a protein sequence was considered one language, and the functions were considered another language. Kulmanov *et al.* [45] proposed a work on the use of CNNs to predict functions based on sequences and interaction data of proteins achieving an $F1\text{-}score$ of 0.40 for BP, 0.50 for MF and 0.64 for CC on data from Swiss-Prot and STRING databases. Miranda and Hu [46] proposed an autoencoder-based model that uses deep-stacked denoising autoencoders to extract features, and multi-labelled SVMs to make classifications on Yeast [47] and Genbase [48] datasets achieving an $F1\text{-}score$ of 0.593 for the Yeast dataset and 0.993 for the Genbase dataset.

The emergence of machine-learning techniques in PFP has brought advances in the predictability of protein functions. However, these techniques require retraining of models with each and every change in the input/output of the training data, which leads to increased computational time and enhanced cost.

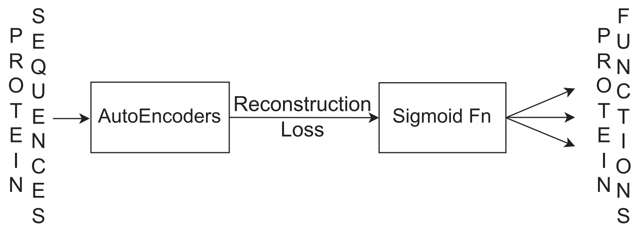


Fig. 1. Block diagram of the proposed model.

In the present work, we propose a semi-supervised autoencoder-based deep-learning technique to predict protein functions. An autoencoder is an unsupervised learning model which tries to reconstruct its own inputs. It is a variant of feed-forward artificial neural networks in which input and target output is same. The hidden layers in the network create a bottleneck, forcing the data to compress. It is mainly used for the compact representation of the data [49]. As far as we know, none of the works on PFP have used autoencoders in a semi-supervised manner. All the works which are found for autoencoders [46], [36] are trained in an unsupervised manner and are mainly used for feature generation. We have used autoencoders that are trained on protein sequences performing the same functions in a semi-supervised manner by preserving the information about the function (i.e. the target label) with which the autoencoder is trained. Preserving the target label for each autoencoder gives the training a shade of supervision, hence the autoencoders are termed as semi-supervised autoencoders. The individual autoencoder is trained on sequences corresponding to a single function and which are capable of predicting if a protein performs that function. Proteins that will perform a function, say F_1, will give lesser reconstruction loss when passed through the autoencoder trained for F_1 as compared to other autoencoders. The losses obtained from all the autoencoders for each protein will then be used as a feature vector to train a dense sigmoid layer to classify the proteins into their respective functions. Since proteins are able to perform multiple functions, their prediction is a multi-label classification [50]. Hence, a sigmoid function at the output layer helps in determining the probability of a protein sequence lying in each class. The proposed model is evaluated in terms of F_{max}, average recall, average precision, and area under the precision-recall curve (AUPR) for the test data. The proposed model outperforms other state-of-the-art deep-learning methods including DeepGO [45] and MLDA [51].

II. Materials and Methods

In order to predict protein functions, a model is trained using protein sequences as input and target functions as output. Fig. 1 shows the high-level block diagram of the proposed model, in which protein sequences are passed through pre-trained autoencoders to get the corresponding reconstruction losses, which are then used by the dense sigmoid layer to predict the functions.

Fig. 2 gives a detailed flow of the proposed model. The complete model is divided into two stages: a) Stage-1 and b)

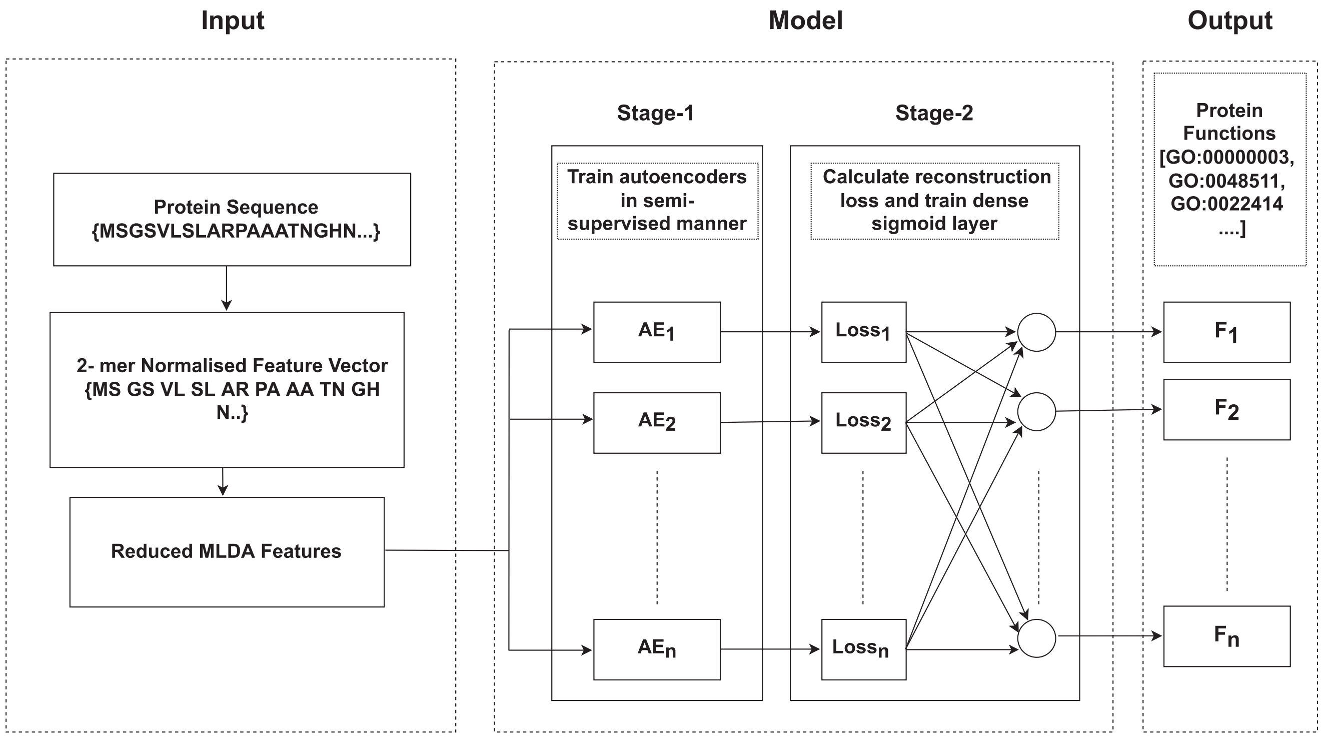

Input
Model
Output
Stage-1
Stage-2
Protein Sequence {MSGSVLSLARPAAATNGHN...}
2- mer Normalised Feature Vector {MS GS VL SL AR PA AA TN GH N..}
Reduced MLDA Features
Train autoencoders in semi-supervised manner
Calculate reconstruction loss and train dense sigmoid layer
AE1
AE2
AEn
Loss1
Loss2
Lossn
Protein Functions [GO:00000003, GO:0048511, GO:0022414 ....]
F1
F2
Fn

Fig. 2. Detailed flow of the proposed system.

TABLE I
PROTEIN SEQUENCE DATA STATISTICS

Protein Function	No. of training data	No. of testing data	No. of functions
Biological Processes	36380	9096	932
Molecular Functions	25223	6306	585

TABLE II
PERFORMANCE MEASURES OF VARIOUS K-MERS

k-mer	F_{max}	Average Precision	Average Recall
1-mer	0.29	0.51	0.20
2-mer	0.36	0.48	0.29
3-mer	0.33	0.39	0.28

Stage-2. Stage-1 comprises the training of autoencoders with protein sequences in such a way that each autoencoder is trained only on the protein sequences corresponding to one function. Stage-2 classifies proteins into their respective protein functions by using the reconstruction losses calculated through the pre-trained autoencoders. To calculate the losses, each protein sequence is passed through the pre-trained autoencoders and reconstruction losses are calculated for every autoencoder. Later, the dense sigmoid layer classifier is trained with input as the reconstruction losses and output as the corresponding protein functions. Lastly, the proposed model is tested on test data.

The rest of this section provides a detailed description of the dataset used, data pre-processing, features generation, and the architecture of the proposed model.

A. Data Preparation

We have used the data[2] used in [45] to train and test the model. The dataset has 585 functions for MF and 932 functions for BP. The statistics of the data used in the proposed work are presented in Table I. Swiss-Prot's manually reviewed and annotated protein sequences that correspond to the following experimental evidence codes (EXP, IPI, IDA, IEP, IMP, IGI, TAS, and IC) are contained in the dataset downloaded from UniProt. No sequences include any ambiguous amino acids (Z, X, U, O, J, B).

In the proposed model, each autoencoder corresponding to a protein function is trained on protein sequences. Protein sequences are converted into a feature map which is input into the autoencoders. The subsequent sections elaborate the pre-processing of input/output data and proposed model architecture.

B. Feature Extraction and Reduction

A protein sequence is created using different combinations of 20 amino acids. There can be at most 20^k combinations of these amino acids, taken k at a time. As a first step, we investigated the performance of the model with features generated through 1-mer (k=1), 2-mer (k=2) and 3-mer (k=3). The perfomance measures for various tested k-mers are represented in Table II. Since the performance of 2-mer is maximum, we continued our experiments with 2-mer features itself. Considering 2-mer, the total number of features generated for a sequence are 400. For

[2]Downloaded https://github.com/bio-ontology-research-group/deepgo

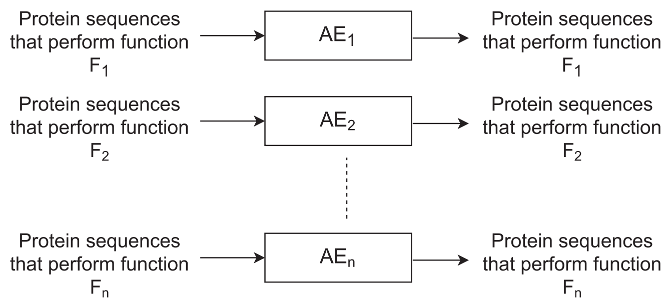

Protein sequences that perform function F1
AE1
Protein sequences that perform function F1
Protein sequences that perform function F2
AE2
Protein sequences that perform function F2
Protein sequences that perform function Fn
AEn
Protein sequences that perform function Fn

Fig. 3. Individual training of autoencoders.

every i^{th} protein sample P_i, a normalized feature vector V_i of size 400 is created [52]. As a result, the entire dataset is represented by a $n * 400$ matrix (M), where n is the number of protein samples, and 400 is the feature size. This matrix is a sparse matrix with 400 columns. This representation may suffer from curse of dimensionality as it has a large number of features. Hence, these features are reduced by using MLDA [51] to get a more compact representation.

C. Proposed Model

We propose a semi-supervised autoencoder-based deep-learning model which takes protein sequences as input and predicts the functions being performed through sequences. Protein sequences are represented as 2-mer features, and protein functions are represented as a multi-hot vector as PFP is a multi-label classification. The complete model is divided into two stages. In Stage-1, a set of autoencoders are trained with protein sequences in which each autoencoder corresponds to a single protein function. Autoencoders are unsupervised models, however since each autoencoder corresponds to an individual function, it gets the shade of supervision. Later in Stage-2, reconstruction loss is calculated for each protein sequence for every autoencoder. The reconstruction losses are normalized and scaled between 0 and 1. A dense sigmoid layered classifier is used to classify the proteins based on the reconstruction losses into their respective functions. An overview of the proposed work is shown in Algorithm 1.

1) Stage-1: AutoEncoders: We train a set of autoencoders corresponding to each function. Let's assume that we have collected data corresponding to k functions, so that we will have k autoencoders corresponding to k functions. An autoencoder AE_1 is trained with protein sequences which performs function F_1, autoencoder AE_2 is trained with protein sequences which performs function F_2 and so on as shown in Fig. 3. In general, an autoencoder AE_j is trained with protein sequences that perform function F_j. The motive behind this way of training is to expose each autoencoder to proteins performing a specific function only, which helps in capturing the information valid for the same function. This gives the autoencoders a shade of supervision. Now, we calculate the reconstruction loss of each protein sequence P_i for each autoencoder. Reconstruction loss (L) is the measure of closeness between reconstructed input by the model and the original input, which can be mathematically represented as (1)

Algorithm 1: Stepwise description of the proposed work.

```
Input:
  • Reduced 2-mer based features constructed from protein sequences
  • Multi-hot vector of protein functions
Output:
  • Predicted functions
STEP 1: Calculate 2-mer normalized feature vectors for all the protein sequences
1:  for each protein sequence $P_i$ (i = 1 to n) do
2:    for each 2-mer (mer = 1 to 400) do
3:      $count_{i,mer}$ = count 2-mers
4:      $NF_{i,mer}$ = Normalize $count_{i,mer}$ by length of $P_i$
STEP 2: Reduce the features using MLDA to get lower dimension features
STEP 3: Represent protein functions into multi-hot vectors
1:  for each protein sequence $P_i$ (i = 1 to n) for
2:    for each protein function $F_j$ (j = 1 to k) for
3:      if $P_i$ performs $F_j$ then
4:        $Vec_{ij} = 1$
5:      else
6:        $Vec_{ij} = 0$
STEP 4: Train set of autoencoders corresponding to each function
1:  for each protein function $F_j$ (j = 1 to k) do
2:    for each protein sequence $P_i$ (i = 1 to n) do
3:      if $P_i$ performs $F_j$ then
4:        add $P_i$ to $TrainingSet_j$
5:    Train autoencoder $AE_j$ with $TrainingSet_j$
6:    Save model $AE_j$
STEP 5: Calculate reconstruction losses of each protein sequence for every autoencoder, normalize and scale them between 0 and 1
1:  for each trained autoencoder $AE_j$ (j = 1 to k)
2:    for each protein sequence $P_i$ (i = 1 to n)
3:      evaluate $ReconstructionLoss_{ij}$
4:  Normalize and scale reconstruction losses to 0 and 1
STEP 6: Train a dense sigmoid layer with reconstruction losses as input and protein functions as output
STEP 7: Test the protein sequences for their respective functions
```

$$L = \frac{1}{m}\sum_{k=1}^{m}(x_k - \hat{x_k})^2, \quad (1)$$

where x_k is original input, $\hat{x_k}$ is the reconstructed input, and m is the number of input features.

If we have k autoencoders, we will have k reconstruction losses for each protein sequence P_i. These losses will be used as a feature to predict protein functions. Each protein sample P_i will be represented as $P_i = \{L_{1i}, L_{2i}, L_{3i}, \ldots, L_{ki}\}$.

The new feature dimension will be $n * k$, where n is the total number of protein sequences and k is the total number of functions (or autoencoders). In this work, 932 and 585 functions are taken for BP and MF respectively. For BP, 932 autoencoders are trained with 351 features. For MF, 585 autoencoders are trained with 317 features. Each autoencoder has three hidden layers with 300, 200, and 300 neurons respectively, and tanh activation function. The autoencoders are trained for 100 epochs, batch size of 8, rmsprop optimizer and loss function as mean_squared_error.

2) Stage-2: Classifier: In the proposed model, a dense sigmoid layer is used as a classifier to classify the protein sequences into their respective functionality. It takes input as calculated reconstruction losses (refer to section II-C1) and output as protein functions. The final output of sigmoid layer is the probability of a sequence lying in each class i.e. probability of a protein sequence performing that function. The classifier works on the ideology that the reconstruction loss of the protein sequence that performs a given function is less when it is passed through the autoencoder trained on the given protein function. Since each protein sequence is passed through every autoencoder to calculate the corresponding reconstruction losses, it is expected to have low loss values for the functions which are being performed by the protein and high loss values for the functions which are not being performed by the sequences. The same learning process is automated in this classifier. A sigmoid layer ensures that the prediction probability for each function is independently computed. The classifier comprises one hidden layer of 500 neurons with ReLU activation function and an output layer of 585 neurons with sigmoid activation function in MF. For BP, the classifier contains a hidden layer of 810 neurons with ReLU activation function and an output layer of 932 neurons with sigmoid activation function. In both MF and BP, the classifier is trained for 500 epochs, batch size of 32, rmsprop optimizer and loss function as mean_squared_error.

D. Integration of Sequences With Protein Interaction Data

We have integrated the features generated for the classification of protein sequences with the protein interaction data to achieve better predictability of the model. The protein interaction data is initially downloaded from STRING database. The knowledge graph embeddings of size 256 are generated for each protein [45]. These features are concatenated with the reconstruction losses for each protein sequence. We train the dense sigmoid layer with these concatenated features and evaluate the performance of the model. We found that integration of network topology data added diversified information to the model and helped in better learning, resulting in an increase in the performance of the model.

E. Integration of Features Based on Physicochemical Properties

We have further integrated our model with the features extracted based on the physicochemical properties of amino acids. Amino acid residues exhibit different physicochemical properties. The amino acids are grouped into three groups (Group I, Group II, Group III) based on their seven physicochemical properties (hydrophobicity, Van der Waals volume, polarity, polarizability, charge, secondary structure, and solvent accessibility), such that those that belong to the same group are considered similar. For example, based on the physicochemical property 'hydrophobicity,' 20 amino acids can be divided into three groups: Group I (RKEDQN), Group II (GASTPHY), and Group III (CVLIMFW). These groups are represented in Table III [53]. Similar to normalised feature vectors of sequences, the normalised frequency features are extracted for members of every group, ie. amino acids characterised by their physicochemical properties [54].

TABLE III
AMINO ACID GROUPING BASED ON PHYSICOCHEMICAL PROPERTIES

Physicochemical Property	Group I	Group II	Group III
Hydrophobicity	RKEDQN	GASTPHY	CLVIMFW
Van der Waals volume	GASCTPD	NVEQIL	MHKFRYW
Polarity	LIFWCMVY	PATGS	HQRKNED
Polarizibility	GASDT	CPNVEQIL	KMHFRYW
Charge	KR	ANCQGHIL MFPSTWYN	DE
Secondary structure	EALMQKRH	VIYCWFT	GNPSD
Solvent accessibility	ALFCGINW	RKQEND	MPSTHY

$$f_{i,j} = \frac{count(AAG_{ij})}{N} \tag{2}$$

where

- $f_{i,j}$ represents normalized frequency of amino acids characterised by their physicochemical properties
- $i = 1, 2,.., 7$ (for each property)
- $j = 1, 2, 3$ (for each group)
- $count(AAG_{ij})$ represents total count of grouped amino acids in the protein sequence
- N = length of the protein sequence

Here, we get 21 dimensional features for each protein sequence. Along with the reconstruction losses and knowledge graph embeddings, we concatenate these 21 features for each protein sequence. We train the dense sigmoid classifier with the concatenated features and evaluate the performance of the model. We found that integration of these features improved the model's learning, resulting in enhanced performance of the model.

F. Predictability of Standalone Autoencoders

The proposed semi-supervised autoencoders are individually used to predict the protein functions. Reconstruction losses corresponding to each autoencoder are calculated for every protein. The proteins are classified into their respective functions based on the assumption that reconstruction loss is less for the proteins performing a given function and vice-versa. We empirically calculate threshold values for each autoencoders' reconstruction loss. If the reconstruction loss is less than the threshold value, then the protein is supposed to perform that function. For example, a protein P_i is passed through a pre-trained autoencoder AE_j (which is trained for function f_j), if the reconstruction loss L_i is less than a threshold value t_j, then the protein performs function f_j and vice-versa.

G. Evaluation Metrics

The proposed model's performance is evaluated with evaluation measures of Critical Assessment of Functional Annotation (CAFA) [55]: F_{max}, average recall, average precision and area under the precision-recall curve (AUPR). We calculate these metrics for an empirically calculated threshold $t \in [0, 1]$.

Average Precision: It is a protein-centric measurement of correct positive predictions among all the positive predictions, averaged over all the protein samples. Equation 3 computes average precision.

$$avgPrecision = \frac{1}{n} * \sum_{i=1}^{n} \frac{Pred_i \cap True_i}{Pred_i}. \tag{3}$$

Average Recall: It is a protein-centric measurement of correct positive predictions among all the actual positive labels, averaged over all the protein samples. Equation 4 computes average recall.

$$avgRecall = \frac{1}{n} * \sum_{i=1}^{n} \frac{Pred_i \cap True_i}{True_i}. \tag{4}$$

F_{max} : F_{max} is the maximum protein-centric $F1$-$score$ calculated over all thresholds. F1 score is the harmonic mean between precision and recall. Equation 5 shows the mathematical representation of F_{max}.

$$F_{max} = \max_t \left(\frac{2 * avgPrecision * avgRecall}{(avgPrecision + avgRecall)} \right), \tag{5}$$

where $Pred_i$ is the set of positive prediction for protein sample P_i, $True_i$ is the set of true annotations for protein sample P_i, n is the total number of samples, and t is the threshold.

AUPR: Area under the precision-recall curve summarizes the precision-recall curve. It represents the tradeoff between precision and recall. It is a measure for a highly imbalanced dataset.

III. Results and Discussion

A semi-supervised autoencoder-based model is built to predict protein functions. Autoencoders are trained with protein sequences corresponding to individual protein functions. Each protein sequence is then passed through the trained autoencoders to compute the corresponding reconstruction losses. The classifier comprising a dense sigmoid layer is trained with the computed reconstruction losses as input and protein functions as output. The model's performance is evaluated on test data.

We compare the proposed model with three different kinds of existing approaches that are state-of-the-art in their respective categories of PFP. First is BLAST, which is homology-based transfer PFP, second is DeepGO [45], which is a deep learning-based PFP, and third is MLDA [51], which is feature-based PFP. The same dataset is used across the comparisons. We have evaluated the proposed work in terms of CAFA-based evaluation measures: average precision, average recall, F_{max}, and AUPR. Table IV shows the comparative performances. The proposed model outperforms all three kinds of approaches (BLAST, DeepGO, MLDA). The proposed model works well due to the focussed training of individual autoencoders that learns about the characteristics of a single protein function at a time and learns to differentiate between the learned protein function and others. Also, integration of other protein data types like protein-protein interaction and physicochemical property-based features added diversified information to the classifier that resulted in an improved predictive performance.

Fig. 4 depicts the term-centric performance of the proposed model. As per the graph, it is found that the functions which have a higher number of performing proteins are predicted with a high $F1$-$score$ in both the cases of biological processes and molecular functions. In case of biological processes, most of the functions have less than 10000 annotated proteins and $F1$-$score$ varies from 0 to 0.45. In case of molecular functions, most of the functions have less than 2000 annotated proteins in the training data and $F1$-$score$ varies from 0 to 0.65. In MF, the proposed model is able to predict functions with lesser annotations too.

Fig. 5 shows the term-centric performance of standalone autoencoders when tested on the basis of reconstruction loss itself. The linear relationship is observed between the $F1$-$score$ and the number of annotated proteins with which the autoencoders are trained. The high number of annotated proteins resulted in better predictability of the functions. The performance of standalone autoencoders is in line with the proposed model. However, in the case of molecular functions, the proposed model was able to predict a few functions with lesser annotations, while standalone autoencoders are not able to do so due to the unoptimized training of individual autoencoders.

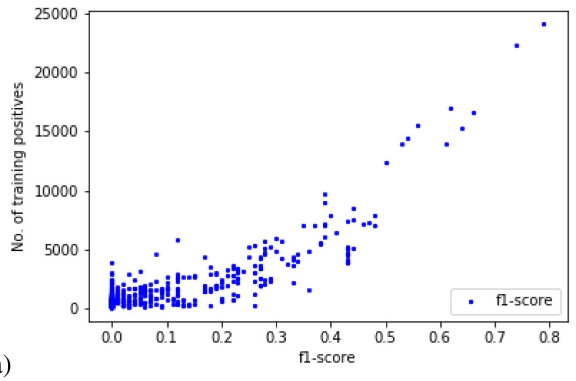
The main contributions of the proposed model are as follows:

- *Semi-supervised autoencoders:* In this work autoencoders are trained in a semi-supervised manner such that they know what they have been trained for. This improves the predictability and generalizability of the model.
- *Scalable model:* Every supervised model needs to be trained again if the target changes. However, in the proposed model, the pre-trained autoencoders need not be trained again for a function. For every new function, a new autoencoder may be trained and added to the existing model. Hence, the proposed model is easily scalable to any number of functions to be predicted.
- *Efficient standalone autoencoders:* Each autoencoder can be used standalone to predict a function. Reconstruction loss of the protein sequences is computed for every autoencoder. If the reconstruction loss is less than a threshold value, then the protein sequence performs the function for which the autoencoder is trained.

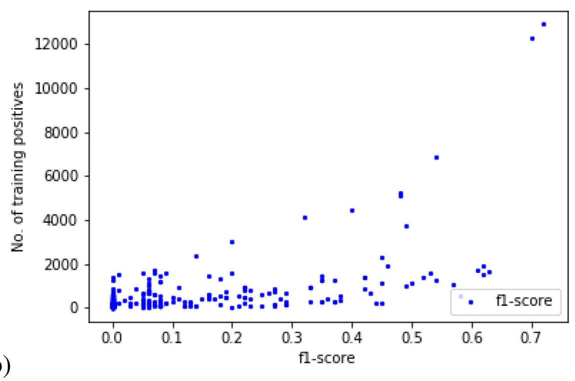
Statistical significance test: We have performed a statistical hypothesis test called Welch's t-test (parametric t-test) to show the statistical significance of our best performing model. The system was executed for 15 runs. Table V tabulates the p-values for different tests at $5\%(0.05)$ significance level. All the p-values

TABLE IV
PERFORMANCE OVERVIEW OF THE PROPOSED WORK AND COMPARISON TO OTHER EXISTING APPROACHES

Methods	Biological Process				Molecular Function			
	Fmax	Average Precision	Average Recall	AUPR	Fmax	Average Precision	Average Recall	AUPR
BLAST [9]	0.314	0.302	0.327	-	0.372	0.367	0.377	-
DeepGOSeq [45]	0.293	0.304	0.282	-	0.364	0.453	0.304	-
DeepGO [45]	0.395	**0.412**	0.379	-	0.47	**0.577**	0.397	-
MLDA (seq) [51]	0.339	0.3332	0.3455	0.26	0.3591	0.365	0.3541	0.25
MLDA (seq & interaction) [51]	0.4158	0.4074	0.4251	0.38	0.4312	0.4693	0.399	0.35
Proposed Model (seq)	0.3487	0.331	0.3684	0.27	0.3837	0.4347	0.3435	0.3
Proposed Model (seq & interaction)	0.4211	0.4019	0.4423	0.38	0.4598	0.5078	0.4201	0.38
Proposed Model (seq & interaction & physicochemical)	**0.4216**	0.4013	**0.4441**	**0.40**	**0.4749**	0.5129	**0.4421**	**0.43**

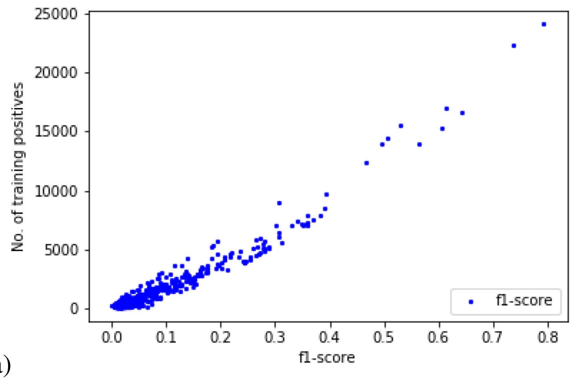


(a)

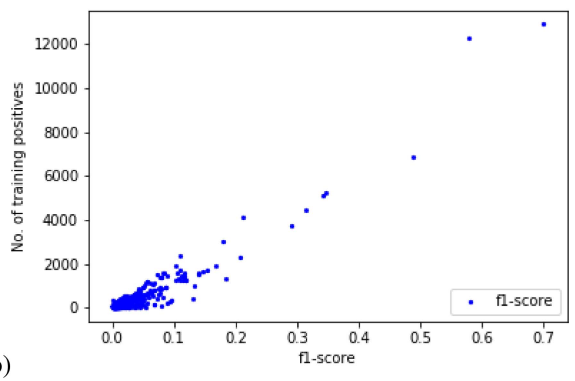


(b)

Fig. 4. Term-centric performance plots illustrating the behaviour of $F1\text{-}score$ with the number of positives in the training data (a) performance of proposed model (dense sigmoid layer) for biological processes (b) performance of proposed model (dense sigmoid layer) for molecular functions.



(a)



(b)

Fig. 5. Term-centric performance plots illustrating behaviour of $F1\text{-}score$ with the number of positives in the training data (a) performance of individual autoencoders for biological processes (b) performance of individual autoencoders for molecular functions.

are less then 0.05, showing that the results achieved are statistically significant, and thus the best results have not occurred by chance.

IV. Conclusion and Future Works

We have proposed a novel approach in which autoencoders are trained in a semi-supervised manner to predict protein functions. These trained autoencoders can be reused by any classifier and need not be trained again for any additional new function. Reconstruction losses are obtained from the autoencoders signifying whether the protein performs that specific function. The classifier with dense sigmoid layer is trained upon the idea that lower reconstruction loss from an autoencoder indicates that the protein performs the function on which the autoencoder is trained, while higher reconstruction loss from the autoencoder indicates that the protein sample does not perform that function. Our experiments demonstrate that the proposed model can

TABLE V
STATISTICAL SIGNIFICANCE TEST RESULTS: P-VALUE AT 5% SIGNIFICANCE VALUE

Methods	Biological Process	Molecular Function
MLDA (seq)	5.99E-27	2.08E-42
MLDA (seq & interaction)	5.42E-08	3.25E-20
DeepGOSeq	1.99E-20	2.45E-26
DeepGO	1.32E-09	6.45E-08
Proposed Model (seq)	9.92E-17	4.37E-25
Proposed Model (seq & interaction)	0.000687	4.28E-12

efficiently predict protein functions outperforming various state-of-the-art models. The experiments with standalone pre-trained autoencoders showed that each autoencoder is individually capable of predicting the corresponding functions in which it was trained, making them scalable to any number of functions. This reduces the effort of retraining the model with every change in target labels. The experiments with sequences and protein interaction data showed that the diversified information about proteins adds value to better learning of the model. Inclusion of features based on physicochemical properties of amino acids added significant information about functional characteristics of proteins that enhanced the performance of the model.

We can further extend the proposed work by leveraging the various outcomes that are observed during the experiments. First, all the autoencoders are trained with the same hyper-parameters. Different autoencoders can be individually optimized for every function, which will give better exposure to the features associated with specific functions. Second, the model takes each autoencoder (trained on individual functions) as an independent entity. Hence, it does not consider the hierarchical relationships among functions neither during training the autoencoders nor while computing the reconstruction losses. The model can be extended further to incorporate these relationships to enhance predictability. Third, the proposed model focuses only on protein sequences to predict the protein functions. A test involving integration of protein interaction data enhanced the model's performance. The proposed model can be extended to incorporate other proteomic data.

REFERENCES

[1] E. Boutet *et al.*, "UniProtKB/Swiss-Prot, the manually annotated section of the UniProt knowledgebase: How to use the entry view," in *Plant Bioinformatics*. Berlin, Germany: Springer, 2016, pp. 23–54.

[2] U. Consortium, "UniProt: A worldwide hub of protein knowledge," *Nucleic Acids Res.*, vol. 47, no. D1, pp. D506–D515, 2019.

[3] S. Carbon *et al.*, "AmiGO: Online access to ontology and annotation data," *Bioinformatics*, vol. 25, no. 2, pp. 288–289, 2008.

[4] M. Ashburner *et al.*, "Gene ontology: Tool for the unification of biology," *Nature Genet.*, vol. 25, no. 1, pp. 25–29, 2000.

[5] G. O. Consortium, "The gene ontology resource: 20 years and still going strong," *Nucleic Acids Res.*, vol. 47, no. D1, pp. D330–D338, 2018.

[6] G. Pandey, V. Kumar, and M. Steinbach, "Computational approaches for protein function prediction: A survey," Dept. Comput. Sci. Eng., Univ. Minnesota, Twin Cities, MN, USA, Tech. Rep. 06-028, 2006.

[7] A. Godzik, M. Jambon, and I. Friedberg, "Computational protein function prediction: Are we making progress?," *Cellular Molecular Life Sci.*, vol. 64, no. 19–20, 2007, Art. no. 2505.

[8] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *J. Mol. Biol.*, vol. 215, no. 3, pp. 403–410, 1990.

[9] S. F. Altschul *et al.*, "Gapped BLAST and PSI-BLAST: A new generation of protein database search programs," *Nucleic Acids Res.*, vol. 25, no. 17, pp. 3389–3402, 1997.

[10] B. Rost, J. Liu, R. Nair, K. O. Wrzeszczynski, and Y. Ofran, "Automatic prediction of protein function," *Cellular Mol. Life Sci. CMLS*, vol. 60, no. 12, pp. 2637–2650, 2003.

[11] H.-J. Yu and D.-S. Huang, "Normalized feature vectors: A novel alignment-free sequence comparison method based on the numbers of adjacent amino acids," *IEEE/ACM Trans. Comput. Biol. Bioinf. (TCBB)*, vol. 10, no. 2, pp. 457–467, Mar./Apr. 2013.

[12] H. Song, B. J. Bremer, E. C. Hinds, G. Raskutti, and P. A. Romero, "Inferring protein sequence-function relationships with large-scale positive-unlabeled learning," *Cell Syst.*, vol. 12, no. 1, pp. 92–101, 2021.

[13] J. Yan, J. Cheng, L. Kurgan, and V. N. Uversky, "Structural and functional analysis of 'non-smelly' proteins," *Cellular Mol. Life Sci.*, vol. 77, no. 12, pp. 2423–2440, 2020.

[14] R. D. Sleator and P. Walsh, "An overview of in silico protein function prediction," *Arch. Microbiol.*, vol. 192, no. 3, pp. 151–155, 2010.

[15] J. Chen, M. Guo, X. Wang, and B. Liu, "A comprehensive review and comparison of different computational methods for protein remote homology detection," *Brief. Bioinf.*, vol. 19, no. 2, pp. 231–244, 2016.

[16] 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, 2018.

[17] J. Hou, R. Cao, and J. Cheng, "Deep convolutional neural networks for predicting the quality of single protein structural models," 2019, *bioRxiv:590620*.

[18] S. K. Sønderby, C. K. Sønderby, H. Nielsen, and O. Winther, "Convolutional LSTM networks for subcellular localization of proteins," in *Proc. Int. Conf. Algorithms Comput. Biol.*, Springer, 2015, pp. 68–80.

[19] Z. Zhang *et al.*, "Deep learning in omics: A survey and guideline," *Brief. Funct. Genomic.*, vol. 18, no. 1, pp. 41–57, 2018.

[20] T. Sun, B. Zhou, L. Lai, and J. Pei, "Sequence-based prediction of protein protein interaction using a deep-learning algorithm," *BMC Bioinf.*, vol. 18, no. 1, 2017, Art. no. 277.

[21] Z. Guo, J. Hou, and J. Cheng, "DNSS2: Improved ab initio protein secondary structure prediction using advanced deep learning architectures," *Proteins: Struct., Function, Bioinf.*, vol. 89, no. 2, pp. 207–217, 2021.

[22] B. Lai and J. Xu, "Accurate protein function prediction via graph attention networks with predicted structure information," *Brief. Bioinf.*, vol. 23, no. 1, 2022, Art. no. bbab502.

[23] L. Wang, Y. Wang, and Q. Chang, "Feature selection methods for big data bioinformatics: A survey from the search perspective," *Methods*, vol. 111, pp. 21–31, 2016.

[24] X. Jing, Q. Dong, D. Hong, and R. Lu, "Amino acid encoding methods for protein sequences: A comprehensive review and assessment," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 17, no. 6, pp. 1918–1931, Nov./Dec. 2019.

[25] W. Li, B. Ma, and K. Zhang, "Optimizing spaced k-mer neighbors for efficient filtration in protein similarity search," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 11, no. 2, pp. 398–406, Mar./Apr. 2014.

[26] D. Cozzetto, F. Minneci, H. Currant, and D. T. Jones, "FFPred 3: Feature-based function prediction for all Gene Ontology domains," *Sci. Rep.*, vol. 6, no. 1, pp. 1–11, 2016.

[27] K.-L. Lin *et al.*, "Feature selection and combination criteria for improving accuracy in protein structure prediction," *IEEE Trans. Nanobiosci.*, vol. 6, no. 2, pp. 186–196, Jun. 2007.

[28] M. Littmann, M. Heinzinger, C. Dallago, T. Olenyi, and B. Rost, "Embeddings from deep learning transfer go annotations beyond homology," *Sci. Rep.*, vol. 11, no. 1, pp. 1–14, 2021.

[29] C. Wan and D. T. Jones, "Protein function prediction is improved by creating synthetic feature samples with generative adversarial networks," *Nature Mach. Intell.*, vol. 2, no. 9, pp. 540–550, 2020.

[30] E. Asgari and M. R. Mofrad, "Continuous distributed representation of biological sequences for deep proteomics and genomics," *PLoS One*, vol. 10, no. 11, 2015, Art. no. e0141287.

[31] Z. Du, Y. He, J. Li, and V. N. Uversky, "Deepadd: Protein function prediction from k-mer embedding and additional features," *Comput. Biol. Chem.*, vol. 89, 2020, Art. no. 107379.

[32] Y. Goldberg and O. Levy, "word2vec Explained: Deriving Mikolov et al.'s negative-sampling word-embedding method," 2014, *arXiv:1402.3722*.

[33] T. Lu, A. X. Lu, and A. M. Moses, "Random embeddings and linear regression can predict protein function," 2021, *arXiv:2104.14661*.

[34] A. Villegas-Morcillo, S. Makrodimitris, R. C. van Ham, A. M. Gomez, V. Sanchez, and M. J. Reinders, "Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function," *Bioinformatics*, vol. 37, no. 2, pp. 162–170, 2021.

[35] R. You *et al.*, "NetGO: Improving large-scale protein function prediction with massive network information," *Nucleic Acids Res.*, vol. 47, no. W1, pp. W379–W387, 2019.

[36] V. Gligorijević, M. Barot, and R. Bonneau, "deepNF: Deep network fusion for protein function prediction," *Bioinformatics*, vol. 34, no. 22, pp. 3873–3881, 2018.

[37] M. Barot, V. Gligorijević, K. Cho, and R. Bonneau, "NetQuilt: Deep multispecies network-based protein function prediction using homology-informed network similarity," *Bioinformatics*, vol. 37, no. 16, pp. 2414–2422, 2021.

[38] S. Yao, R. You, S. Wang, Y. Xiong, X. Huang, and S. Zhu, "NetGO 2.0: Improving large-scale protein function prediction with massive sequence, text, domain, family and network information," *Nucleic Acids Res.*, vol. 49, no. W1, pp. W469–W475, 2021.

[39] K. Fan, Y. Guan, and Y. Zhang, "Graph2GO: A multi-modal attributed network embedding method for inferring protein functions," *GigaScience*, vol. 9, no. 8, 2020, Art. no. giaa081.

[40] A. S. Rifaioglu, T. Doğan, M. J. Martin, R. Cetin-Atalay, and V. Atalay, "DEEPred: Automated protein function prediction with multi-task feed-forward deep neural networks," *Sci. Rep.*, vol. 9, no. 1, pp. 1–16, 2019.

[41] M. Kulmanov and R. Hoehndorf, "DeepGOPlus: Improved protein function prediction from sequence," *Bioinformatics*, vol. 36, no. 2, pp. 422–429, 2020.

[42] Y. Cai, J. Wang, and L. Deng, "SDN2GO: An integrated deep learning model for protein function prediction," *Front. Bioeng. Biotechnol.*, vol. 8, p. 391, 2020, doi: 10.3389/fbioe.2020.00391.

[43] M. Tanveer *et al.*, "Machine learning techniques for the diagnosis of Alzheimer's disease: A review," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 16, no. 1s, pp. 1–35, 2020.

[44] R. Cao, C. Freitas, L. Chan, M. Sun, H. Jiang, and Z. Chen, "ProLanGO: Protein function prediction using neural machine translation based on a recurrent neural network," *Molecules*, vol. 22, no. 10, 2017, Art. no. 1732.

[45] M. Kulmanov, M. A. Khan, and R. Hoehndorf, "DeepGO: Predicting protein functions from sequence and interactions using a deep ontology-aware classifier," *Bioinformatics*, vol. 34, no. 4, pp. 660–668, 2018.

[46] L. J. Miranda and J. Hu, "A deep learning approach based on stacked denoising autoencoders for protein function prediction," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf.*, 2018, vol. 1, pp. 480–485.

[47] A. Elisseeff and J. Weston, "A Kernel method for multi-labelled classification," *Adv. Neural Inf. Process. Syst.*, vol. 14, pp. 681–687, 2001.

[48] S. Diplaris, G. Tsoumakas, P. A. Mitkas, and I. Vlahavas, "Protein classification with multiple algorithms," in *Panhellenic Conference Informatics*. Berlin, Germany: Springer, 2005, pp. 448–456.

[49] A. Gupta, H. Wang, and M. Ganapathiraju, "Learning structure in gene expression data using deep architectures, with an application to gene clustering," in *Proc. IEEE Int. Conf. Bioinf. Biomed.*, 2015, pp. 1328–1335.

[50] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1819–1837, Aug. 2014.

[51] H. Wang, L. Yan, H. Huang, and C. Ding, "From protein sequence to protein function via multi-label linear discriminant analysis," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 14, no. 3, pp. 503–513, May/Jun. 2017.

[52] R. Dhanuka and J. P. Singh, "Protein function prediction using functional inter-relationship," *Comput. Biol. Chem.*, 2021, Art. no. 107593.

[53] D. Chen, X. Tian, B. Zhou, and J. Gao, "ProFold: Protein fold classification with additional structural features and a novel ensemble classifier," *BioMed Res. Int.*, vol. 2016, pp. 1–10, 2016.

[54] X. Qu, D. Wang, Y. Chen, S. Qiao, and Q. Zhao, "Predicting the subcellular localization of proteins with multiple sites based on multiple features fusion," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 13, no. 1, pp. 36–42, Jan./Feb. 2016.

[55] N. Zhou *et al.*, "The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens," *Genome Biol.*, vol. 20, no. 1, pp. 1–23, 2019.