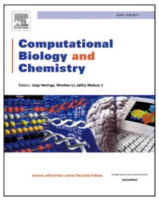
Contents lists available at ScienceDirect

Computational Biology and Chemistry

journal homepage: www.elsevier.com/locate/cbac



Research Article

Deep_CNN_LSTM_GO: Protein function prediction from amino-acid sequences

Mohamed E.M. Elhaj-Abdou [a,*], Hassan El-Dib [a], Amr El-Helw [a], Mohamed El-Habrouk [b]

[a] *Faculty of Engineering, Arab Academy for Science and Technology and Maritime Transport, Alexandria, Egypt*
[b] *Faculty of Engineering, Alexandria University, Alexandria 21544, Egypt*

A R T I C L E I N F O

Keywords:
Deep learning
Gene ontology
CNN
LSTM
Protein function prediction
MF
BP
CC
UniProt-SwissProt
CAFA

A B S T R A C T

Protein amino acid sequences can be used to determine the functions of the protein. However, determining the function of a single protein requires many resources and a tremendous amount of time. Computational Intelligence methods such as Deep learning have been shown to predict the proteins' functions. This paper proposes a hybrid deep neural network model to predict an unknown protein's functions from sequences. The proposed model is named Deep_CNN_LSTM_GO. Deep_CNN_LSTM_GO is an Integration between Convolutional Neural network (CNN) and Long Short-Term Memory (LSTM) Neural Network to learn features from amino acid sequences and outputs the three different Gene Ontology (GO). The gene ontology represents the protein functions in the three sub-ontologies: Molecular Functions (MF), Biological Process (BP), and Cellular Component (CC). The proposed model has been trained and tested using UniProt-SwissProt's dataset. Another test has been done using Computational Assessment of Function Annotation (CAFA) on the three sub-ontologies. The proposed model outperforms different methods proposed in the field with better performance using three different evaluation metrics (Fmax, Smin, and AUPR) in the three sub-ontologies (MF, BP, CC).

1. Introduction

In the bioinformatics field, one of the most essential and critical motifs is proteins (Li et al., 2002). Proteins provide many essential functions and responsibilities in the organism's body, such as DNA structure, muscle building, antibody support, immunity against diseases, and much more. Their function characterizations and annotations serve many sensitive biological, computational applications, such as new drug discovery that could help in global pandemics like HIV/Aids, Cholera, COVID-19 viruses, and many more uses. Protein function prediction can help find the relations between such genes and phenotypes or genetic diseases by understanding the mechanisms, patterns, and relations between these diseases and the genes located in that organism (Lindsay, 2003). There are massive, exponentially increasing needs for discovering these functions for such applications. However, traditional experimental procedures for discovering these functions in the genomics laboratories are very slow compared to these needs (Saeidnia et al., 2015).

On the other hand, computational-intelligent methods and algorithms were proposed and introduced to predict the proteins' function using amino acid sequences faster. These use the complete information and a plethora of protein functions that have been published in biological research. This biological information is digitized and saved in online public databases that can be accessed and downloaded without limitations.

Protein function prediction has been a hot topic application in bioinformatics for the last two decades. Providing information and understanding of protein functions can increase the speed of drug discovery. This is possible due to many factors (Keedwell and Narayanan, 2005), such as the diversity of the computational-intelligent techniques, the possibility of merging between them to overcome the computational bottlenecks, fast and accurate predictions, in addition, the strong hardware support.

Protein function prediction methods can be classified according to the biological information data types, such as protein amino acid sequences, 3D structure, protein folding information, protein-protein interaction networks, gene expression, protein family, integration between these various sources, and more. As mentioned earlier, many databases have been proposed and published online to describe the biological information of proteins in its digital format suitable for

* Corresponding author.
E-mail addresses: mohamed_elhajabdou@student.aast.edu (M.E.M. Elhaj-Abdou), hassaneldib@aast.edu (H. El-Dib), elhelw@aast.edu (A. El-Helw), eepgmme1@yahoo.co.uk (M. El-Habrouk).

https://doi.org/10.1016/j.compbiolchem.2021.107584
Received 5 May 2021; Received in revised form 8 September 2021; Accepted 21 September 2021
Available online 24 September 2021

computation.

UniProtKB (The UniProt Consortium, 2019) database was proposed for amino acid sequences. It is an extensive database published to provide rich annotations and information on proteins and their functions.

Many global initiatives have been provided and proposed in order to find the unknown functions for a known sequence such as Critical Assessment of Function Annotations (CAFA) challenges (Radivojac et al., 2013; Jiang et al., 2016; Zhou et al., 2019). Full details for UniProtKB and CAFA will be further explored and discussed in Section III.

Protein Information Resource (PIR) (Wu et al., 2003) database is an online public resource that provides information for both genome and protein levels to represent the sequence information for each one of them.

For 3D structure protein representation, Protein Data Bank (PDB) (Berman et al., 2000), Nucleic acid Receptors DataBase (NucleaRDB) (Vroling et al., 2012), and Structural Classification Of Proteins (SCOP) (Murzin et al., 1995) databases are available. PDB and NucleaRDB deal with 3D shapes of both proteins and nucleic acids. NucleaRDB provides 2D and 3D structural information, alignment, the chromosomal location of nuclear receptor genes, potential Nuclear Localization Signals (NLS), and binding partners, in addition to sequence-based information such as cDNA, multiple alignments, and phylogenetic trees. SCOP provides detailed information about evolutional relationships between PDB-database proteins dealing with family, superfamily, folding, and protein types.

The interactions between proteins are provided in many different databases, such as the Search Tool for the Retrieval of Interacting Genes/ Proteins (STRING) (Szklarczyk et al., 2016) database. STRING uses a spring model to generate the network images. Nodes are modeled as masses and edges as springs. The nodes' final position in the image is computed by minimizing the system sub-cellular locations 'energy'. It contains 24.5 million proteins from more than 5000 organisms.

Clusters of Orthologous Groups (COG) protein database (Tatusov et al., 2000) is an attempt to reach a phylogenetic classification of the proteins encoded in 21 complete genomes of bacteria, archaea, and eukaryotes. Phylogenetic classification represents how species are related to each other through a common ancestor.

Some other databases deal with one or a group of specific proteins such as LIPASE (Fischer and Pleiss, 2003), G Protein-Coupled Receptors (GPCRDB) (GPCRdb, 2020), and Transporter Classification Databases (TCDB) (Saier et al., 2006). LIPASE deals with the enzyme LIPASE, responsible for the breakdown of fats in food during the digestion process. GPCRDB deals with the family of G proteins acting as molecular switches inside cells. TCDB classifies more than 1500 families of membrane transport proteins in a different organism.

The rest of this paper is organized as follows: Section II presents the background and related work. In section III materials and methods are introduced and illustrated with full details. In section IV, explains the proposed model. While in Section V is the theoretical part explanation of the used evaluation metrics. the results were discussed and evaluated with multiple evaluation metrics. Section VI. In Section I discuss and mention the advantageous of the proposed model. And finally, Section II concludes the paper as well as pointing out suggested future work.

2. Related work

Another way to classify the protein function prediction methods is according to the types of computational approaches used:

- Conventional
- Machine Learning
- Deep Learning

BLAST (Altschul et al., 1990), the primary technique from the conventional approaches, is a tool proposed for local alignment search to compare and find the similarities between two sequences for protein function prediction. Many updates and improvements have been proposed on BLAST, such as PSI-BLAST (Altschul et al., 1997), for performance enhancement.

ProMK (Yu et al., 2015) as a machine learning approach uses a combination of five different kernels from the K-Nearest Neighbors (KNN) algorithm (Euclidean, Standardized Euclidean, Cosine, Correlation, and Spearman) targeting different orgasms such as yeast, human, and mouse) for predicting the function of the protein.

A variety of deep learning approaches have been proposed for protein function prediction. Some of these techniques used UniProtKB and CAFA datasets. These techniques are briefly summarized in the following paragraphs for the sake of comparison with the proposed technique.

In Go-FDR (Gong et al., 2016) was proposed for protein function prediction from sequences. GO-FDR uses the PSI-BLAST sequence alignment algorithm to generate position-specific scoring matrix PSSM (Alejandro et al., 1999) to score the relevant GO term depends on the relative entropy. The evaluation is done against the CAFA2 competition and ranked from the top methods in this compition.

In FFPred 3.0 (Cozzetto et al., 2016), the SVM algorithm is selected to generate predictions by scanning the input sequences. SVM library was trained using the GO annotations and UniProtKB. The training set was extended to cover the three domains MF, BP, CC. SVM with Matthews correlation coefficient (MCC) is used, candidate functional classes were identified based on the availability of sufficiently large and confident positive and negative instances.

In Go-labeler (You et al., 2018a), logistic regression combined with BLAST-KNN was used for protein function prediction. The model is trained on the UniProtKB dataset. To extract the information and the features from the sequences, K-mers (Kawulok and Deorowicz, Apr 17, 2015), InterPro (Mitchell et al., 2015), and ProFET (Ofer and Linial, 2015) algorithms were used and conducted. The evaluation is done using Fmax, S-min, AUPR metrics on CAFA1 and CAFA2.

In Deep_GO (Kulmanov and Khan, 2018), a convolutional neural network (CNN) is used and trained for protein function prediction using two different sources, the amino acid sequences from UniProtKB-Swissport and the protein-protein interaction network from the STRING databases. The model was evaluated using Fmax, AUPR, and MCC. The CNN model was constructed with around 20 layers included pooling, embedding, and activation functions.

MTDNN (Fa et al., 2018) proposed a model for predicting human proteins using a malti-task feedforward neural network. The model is constructed from 6 feedforward layers with activation functions: RELU, sigmoid, and softmax in the output layer. The proposed method uses both the shared representations of all tasks and specific characteristics of individual tasks. MTDNN trained the model using the GO dataset to predict and annotate the amino acid sequence against only five different classes. The evaluation is done using the F1 score and the CAFA challenge.

Deep_Go_Plus (Kulmanov and Hoehndorf, 2020) predicts the functions of the protein from sequences. A CNN and similarity-based method BLAST are combined using the weighted sum approach. The proposed model was constructed from 49 layers activated with sigmoid and RELU activation functions. The model used UniProtKB-Swissport as the training dataset. While the test set used CAFA challenge datasets and subset samples from the UniProtKB-Swissport. The proposed model was evaluated using Fmax, Smin, and AUPR.

From the previous deep learning discussed methods, Deep-GO-Plus (Kulmanov and Hoehndorf, 2020) and DeepLoc (José Juan Almagro Armenteros et al., 2017) used a combination of multiple techniques. Deep_Go_Plus (Kulmanov and Hoehndorf, 2020) combined BLAST with a CNN using a weighted sum method, while DeepLoc (José Juan Almagro Armenteros et al., 2017) combined LSTM and CNN. In addition, for the machine learning approach, ProMK (Yu et al., 2015) combined multiple kernels of KNN.

The approaches (Taju et al., 2018; Nauman et al., 2019; José Juan

Almagro Armenteros et al., 2017; Wei et al., 2018; Clark and Radivojac, 2011; Minneci et al., 2013; Yu et al., 2015; Yunes and Babbitt, 2019) were also proposed in the literature. However, they will not be discussed because of these approaches are:

- used different datasets (Taju et al., 2018; Nauman et al., 2019; Minneci et al., 2013)
- applied their techniques to different targets (such as subcellular localization prediction (José Juan Almagro Armenteros et al., 2017; Wei et al., 2018) and proteins family function prediction (Taju et al., 2018)).
- The model was designed and Performed on a subset of the domains (Clark and Radivojac, 2011; Minneci et al., 2013; Yunes and Babbitt, 2019) (MF, BP, and CC).

All these reasons rendered their comparison with the proposed technique in this paper rather difficult.

Among these other approaches, DeepLoc (José Juan Almagro Armenteros et al., 2017) used a combination of two deep learning architectures (LSTM and CNN) for subcellular localization prediction using amino-acid sequences. The training was performed on the UniProtKB database against ten subcellular locations. The CNN model was designed with two layers, while the RNN model was designed with two LSTM layers. The maximum length of the input sequence in this approach was limited to 1000.

This paper proposes the combination of two cascaded deep learning algorithms (CNN and LSTM) to improve the results and reduce the computational burden in protein function prediction.

3. materials and methods

3.1. Training and test datasets

This paper is focused on the primary Amino-Acid sequence. To train, validate and test the model UniProtKB-SwissProt database was used. In addition, CAFA datasets were selected for testing the proposed model for different organisms.

3.1.1. UniProtKB dataset

UniProtKB (The UniProt Consortium, 2019) database was proposed for amino acid sequences. It is an extensive database published to provide rich annotations and information on proteins and their functions. Some of the UniProtKB database's core data are amino-acid sequences, protein names, protein descriptions, cross-references, taxonomy data, and citation information. UniProtKB consists of two main sub-datasets, UniProtKB-SwissProt and UniProtKB-TrEMBL.

UniProtKB-SwissProt is a high-quality, fully experimentally proven annotations. This information was manually extracted from both computational methods and biological literature and then verified with different experimental techniques performed in biological labs.

The second section is UniProtKB-TrEMBL, which is the information that was extracted from computational methods only and still waiting to be manually reviewed in biological literature.

as shown in Fig. 1 and Fig. 2 the number of entries in both UniProtKB Swissport and TrEMBL. For the UniProtKB/TrEMBL is remarkably increasing over 200 M entries, while on the other hand UniProtKB/ Swissport over 500 K entries.

In order to take a look inside the distribution of the sequences lengths of the overall entries in the database, the following Fig. 3 shows a statistical result from lengths range from 1 to > 2600 in UniProtKB/ TrEMBL. While on the other hand UniProtKB/ Swissport length distribution shown in Fig. 4.

The sequences of proteins employed in this research were conducted and downloaded from the UniProtKB database (The UniProt Consortium, 2019). UniProtKB-SwissProt's was used in the proposed method to construct the training and the test set. These include:

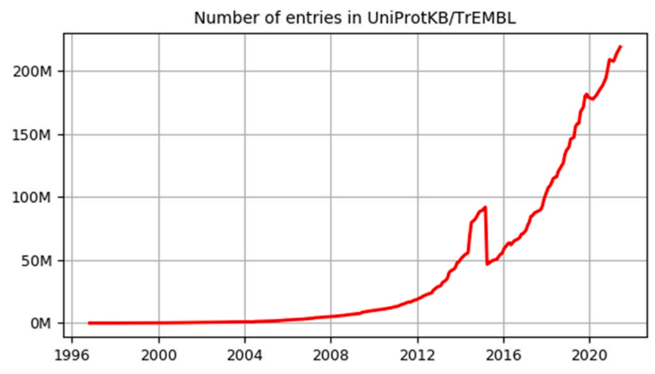


Fig. 1. Number of Entries in UniProtKB/TrEMBL.

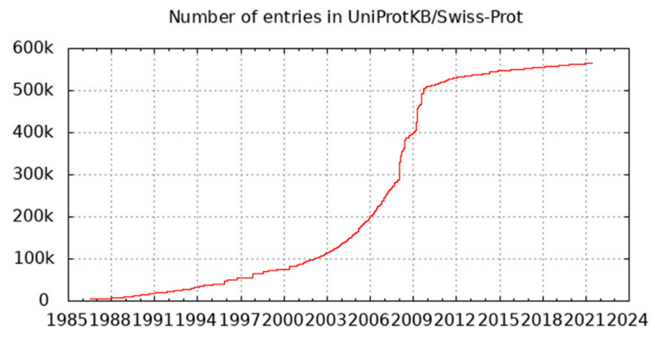


Fig. 2. Number of Entries in UniProtKB/SwissProt's.

- Protein names
  - o Protein name is a string code constructed from chars and numbers such as "1433B_HUMAN" for protein name (14-3-3) protein beta/ alpha. its Homo sapiens protein with CC Function type.
- Accessions entity
  - o It's a list of numbers associated with the entity, this numbers provide a stable way of identifying entries from the release dates, this helps in the mapping procedure between the Gene Ontology dataset which is the target and the amino acid sequences.
- Amino-acid sequences
  - o Represent the sequence of the alphabets chars each alpha bit represent a specific amino acid.
- Annotations (GO ID's)
  - o Gene-Ontology represent the target functions of that specific protein.
- Organism's name
  - o Represent the kingdom of the species such as bacteria species or homo- sapiens species.

The training and test sets were filtered using two criteria. First, the datasets were filtered with the annotations that have biological experimental evidence codes (IGI, IEP, TAS, IC, EXP, IDA, IPI, IMP) targeting human proteins. Second, a filter was performed by removing the proteins that have no GO annotations. The number of samples in the training, validation, and test sets were 51344, 5705, and 71, respectively as shown in Fig. 5.

3.1.2. CAFA dataset

CAFA is a yearly global challenge proposed in the field of bioinformatics aimed to predict the function of proteins from their sequences or structures. CAFA challenge series was introduced in 2010/11 in order to find functions of proteins that rapidly increased in online databases and biological literature. Table 1 presents a comparison between the various CAFA datasets available online.

To evaluate the proposed method, three different test sets were

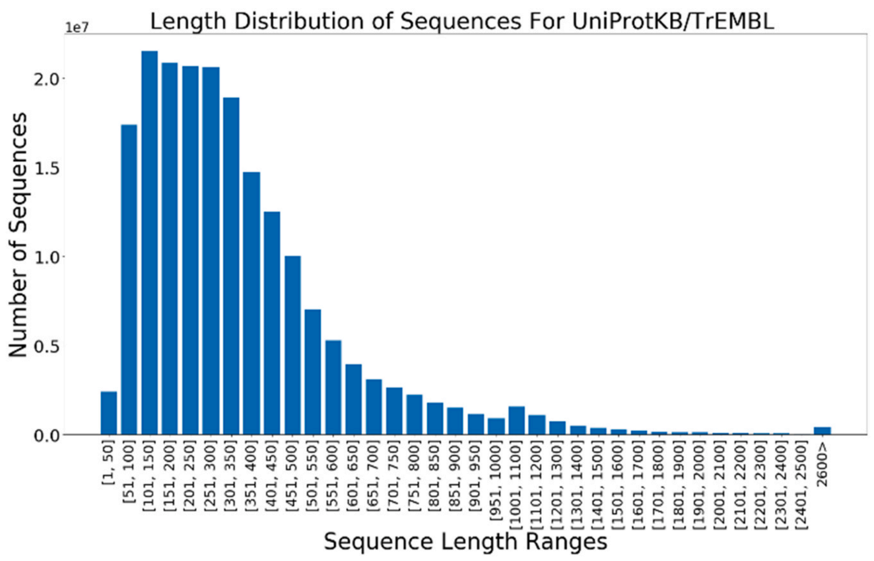


Fig. 3. Length Distribution of sequences in UniProtKB/TrEMBL.

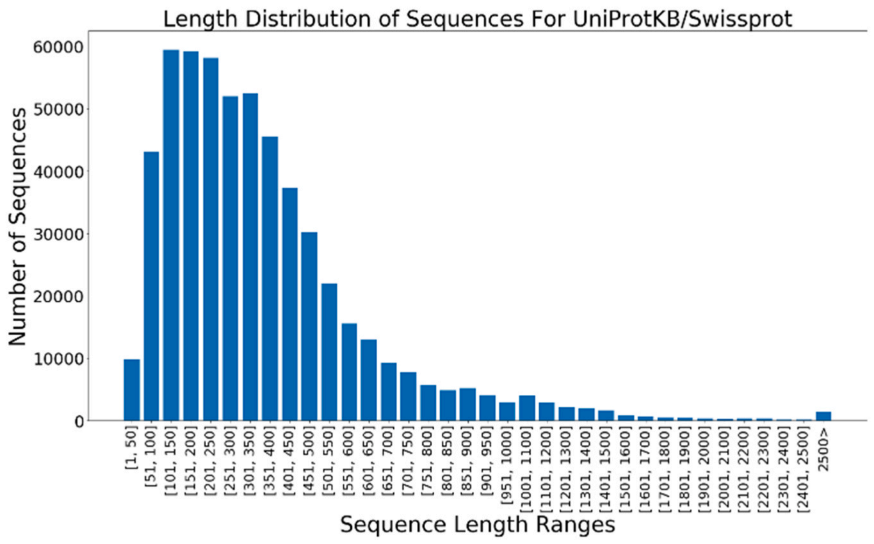


Fig. 4. Length Distribution of sequences in UniProtKB/SwissProt's.

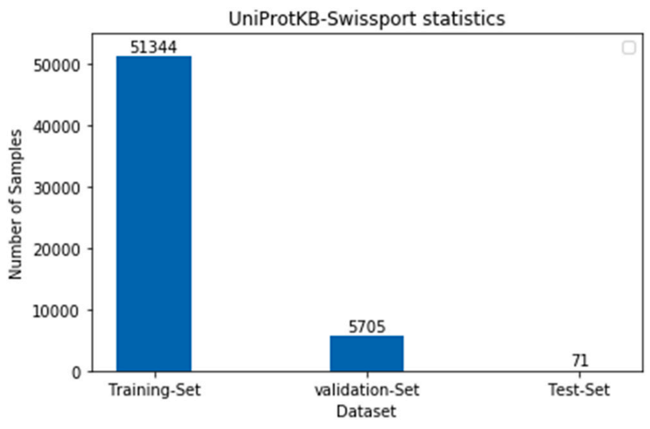


Fig. 5. The training validation and test sets statistics.

Table 1
CAFA Data Sets Comparison.

Dataset	Introduced	Number of Organisms	Tasks
CAFA4	2019/20	18	1-Phenotype prediction. 2-Protein function prediction.
CAFA3	2017/18	23	1-Phenotype prediction. 2-Protein function prediction
CAFA2	2013/14	28	Protein function prediction
CAFA1	2010/11	18	Protein function prediction

selected from the CAFA series. From CAFA4, a number of samples were selected as test cases from eight different organisms, including human to test if the proposed model can adapt with other species sequences. From CAFA3, a specific number of samples were selected from the human organism. The number of test sets selected from each CAFA dataset is presented in Table 2. While in CAFA2, 281 samples from bacteria organism were selected.

Table 2
CAFA datasets used.

Dataset	Organism Name	Number of Test Sets
CAFA4	Homo-Sapian (Human)	122
	Arabidopsis thaliana	535
	Danio rerio	18
	Mus musculus Linnaeus	136
	Rattus norvegicus	42
	Dictyostelium Discoideum	39
	Escherichia coli K-12	53
	Schizosaccharomyces Pombe 972 h-	56
CAFA3	Homo-Sapian (Human)	1131
CAFA2	Bacteria	281

3.2. Predicted (target) functions

Gene-Ontology (GO) (Ashburner et al., 2000) is a hierarchical description of protein functions that describe three categories. The Growth of the Gene-Ontology annotations in the scientific publications research over time is remarkably increasing as shown in Fig. 6 the number of annotations over the past three years. The GO annotations expected to reach over 200 K annotations in 2022 which a huge contribution in this resource.

Gene-Ontology represents the target of the predicted functions of this research. These protein functions can be categorized as follows:

- **Molecular Function (MF)** are those functions belonging to the activities that occur at the molecular level (simple processes). MF describes the action of the activities. However, they do not describe where or when the activity takes place. An example of MF is the catalytic activity or binding activity.
- **Biological Process (BP)** represents the largest and complex processes. Those processes represent MF activities or chemical reactions that are involved inside and/or outside cells. An example of BP is DNA repair and signal transduction.
- **Cellular Component (CC)** are those structures of which cells are composed. CC represents complex places that exist inside and/or outside cells where activities are performed. An example of CC is ribosome and mitochondrion.
- In this paper, GO annotations were used for the October 2020 release. These annotations amount to a total number of classes for 12157 for MF, 30851 for BP, and 4396 for CC as shown in Fig. 7. The number of protein functions required to be identified in this paper amounts to 4600 protein functions.

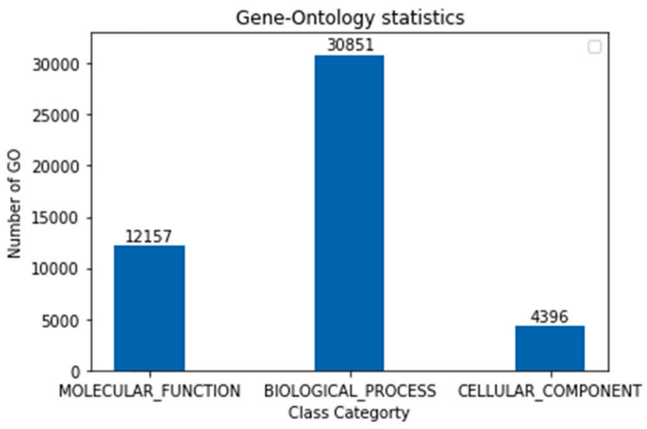


Fig. 7. Gene-Ontology annotations used in the dataset.

3.3. Preprocessing technique

Protein sequences are made up of sequences of 20 Amino-Acid characters as: Alanine (A), Arginine (R), Asparagine (N), Aspartic acid (D), Cysteine (C), Glutamine (Q), Glutamic acid (E), Glycine (G), Histidine (H), Isoleucine (I), Leucine (L), Lysine (K), Methionine (M), Phenylalanine (F), Proline (P), Serine (S), Threonine (T), Tryptophan (W), Tyrosine (Y), Valine (V). These sequences are similar to the example sequences shown in Fig. 8.

One hot-encoding creates a new dimension for each sequence (Rodríguez et al., 2018). All of these dimensions are orthogonal to each other in the vector space. The length of that vector is equal to the total number of sequences. As shown in Fig. 8**,** each dimension is represented as an integer binary vector of zeros except at the corresponding sequence index, where a value of 1 is assigned. The methodology behind using one-hot encoding is to prevent the model from overfitting behavior during the training process (Shorten and Khoshgoftaar, 2019).

One hot-encoding is used in the proposed model to represent the input layer in order to encode the input sequences. In this paper, the proposed model can accept lengths of input protein sequences up to 2000 amino acids each. This is performed to cover the majority of input amino acid sequences proposed in the scientific literature for real-life scenarios.

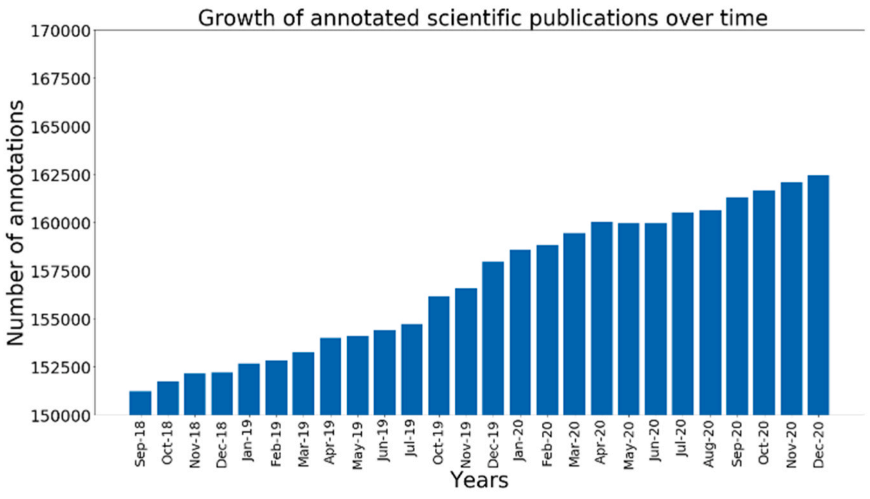
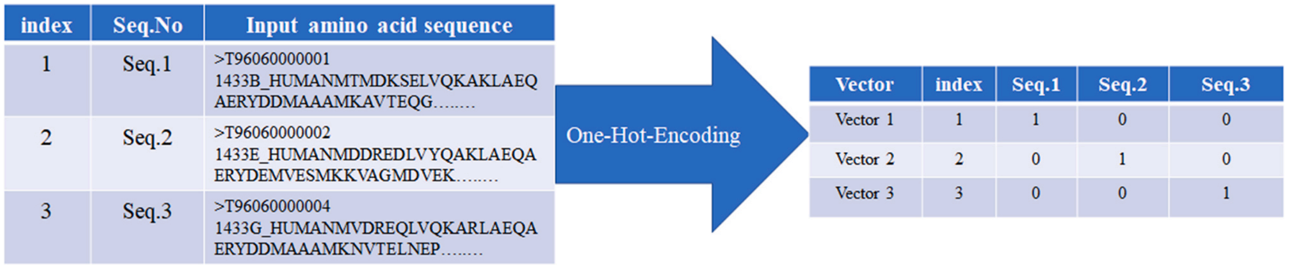


Fig. 6. Growth of Gene-Ontology Annotations over time.



index	Seq.No	Input amino acid sequence
1	Seq.1	>T96060000001 1433B_HUMANMTMDKSELVQKAKLAEQ AERYDDMAAAMKAVTEQG........
2	Seq.2	>T96060000002 1433E_HUMANMDDREDLVYQAKLAEQA ERYDEMVESMKKVAGMDVEK........
3	Seq.3	>T96060000004 1433G_HUMANMVDREQLVQKARLAEQA ERYDDMAAAMKNVTELNEP........

Vector	index	Seq.1	Seq.2	Seq.3
Vector 1	1	1	0	0
Vector 2	2	0	1	0
Vector 3	3	0	0	1

Fig. 8. Representation of One Hot-Encoding process for three input amino acid sequences.

3.4. *Neural networks building blocks*

3.4.1. *Pooling*

Pooling is a technique represented as a layer (Abdel-Hamid et al., 2013), for down sampling the convolutional layers' dimensions without losing the input features' details (O'Shea and Nash, 2015). Moreover, pooling is a spatial invariant technique that can detect the object's features in different positions. Two main types of pooling, max, and average. Max pooling is better for extracting the extreme features, while average pooling sometimes takes all features into accounts, resulting in non-crucial features (Giusti et al., 2013). In the proposed model Max-Pooling type (Krizhevsky et al., 2017) is used and implemented.

3.4.2. *Activation functions*

Activation functions add some non-linearity to the output neurons to allow the model to learn complex tasks (Sharma, 2017). According to the type of the function used, it activates/deactivates some neurons from the previous layer depending on the specific shape of the activation function (Lau and Lim, 2017).

In the proposed system, two different activation functions were used. The LSTM model uses a "tanh" activation function, as represented by Eq. 1, which illustrates the hyperbolic tangent function. The plotted graph of the function is depicted in Fig. 9-A.

$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \qquad (1)$$

Alternatively, the sigmoid function was chosen as an output layer of the overall system, as represented by Eq. 2. Fig. 9-B depicts the plotted graph of the sigmoid function employed.

$$f(x) = \frac{1}{1 + e^{-x}} \qquad (2)$$

3.4.3. *Flatten layer*

A flatten layer is used to convert the convolutional layer's output into a single long feature vector (1-D array). This process is performed to feed this vector to the output layer. According to studies (Chollet, 2017; Jonghoon et al., 2014), flatten layers can effectively speed up the learning process by double.

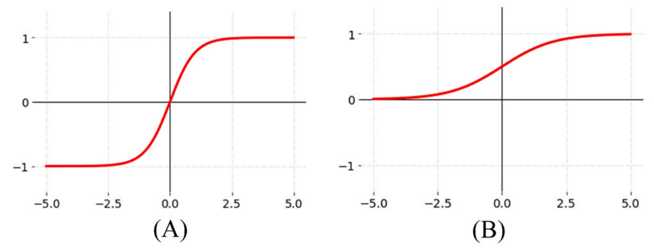


Fig. 9. Plotted graphs of the activation functions employed in Deep-CNN-LSTM-GO, (A) "tanh" activation function (B) "Sigmoid" activation function.

3.4.4. *Convolutional Neural Network (CNN)*

Convolutional Neural Network (CNN) is a supervised learning technique (Sainath et al., 2013). CNN can merge the feature extraction and feature classification processes into a single body. Compared to the conventional fully connected Multi-Layer Perceptron (MLP), CNN can deal with large inputs with high computational efficiency (Gu et al., 2018).

Since the input data is a One-dimensional (1D) sequence, it is convenient to use a 1D CNN configuration in the proposed model that fits the input data sequences with different size kernels (Kiranyaz et al., 2019). 1D CNN simply requires array operations, this means computational complexity is significantly lower than 2D CNNs. Therefore 1D CNN can be trained on any standard hardware setup and does not require a GPU.

3.4.5. *Long-Short Term Memory (LSTM)*

Long short-term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is a recurrent neural network architecture used in sequence modeling and NLP applications. LSTM is an enhanced version of the original Recurrent Neural Network (RNN) architecture. LSTM solves the inability to access the long term memory problem of RNN. Moreover, LSTM solves the vanishing gradient problem that the original RNN architecture suffers from (Hochreiter and Schmidhuber, 1997).

LSTM consists of three main parts: an input gate, an output gate, and a forget gate. It can be trained to learn what information to store in the Memory, how long to store it, and when to read it out (Sherstinsky, 2020). In the proposed model, multiple LSTM are used with different sizes, which will be described in greater details in the next section.

4. **proposed Deep_CNN_LSTM_GO model**

The proposed model is constructed from a combination of CNN and LSTM Neural Network as shown in Fig. 10. This hybrid model overcomes the sequence modeling problems and limitations, by taking the advantages of both architectures.

For Deep_CNN_LSTM_GO, the protein sequence is encoded into a matrix using one-hot encoding, which can be seen at the left of Fig. 10.

For each CNN block the model extracts the local features of input and at a same time maps these inputs with some non-linearity to the output. This process is repeated for each region, those regions are defined using the number of filters and kernel sizes. As a result of that, CNN is a Neural Network architecture that can learn the features hierarchically. In other words, as much as the number of convolution layers are increased, high-level, complex features are learned.

On the other hand, LSTM is an ingenious technique that can make the Neural Network decide what important information to remember for further use and what to filter out and forget in the sequence. Using this advantage, the neuron can learn from features and predict subsequent values. This combination enables the model to learn regional and temporal features at a time. In other words, the convolutional layers detecting "biological words" (regional pattern) and the LSTM tying them into a "biological sentence" with the advantage of the "remember" or "forget" characteristics, which is then fed to a flatten layer that generates

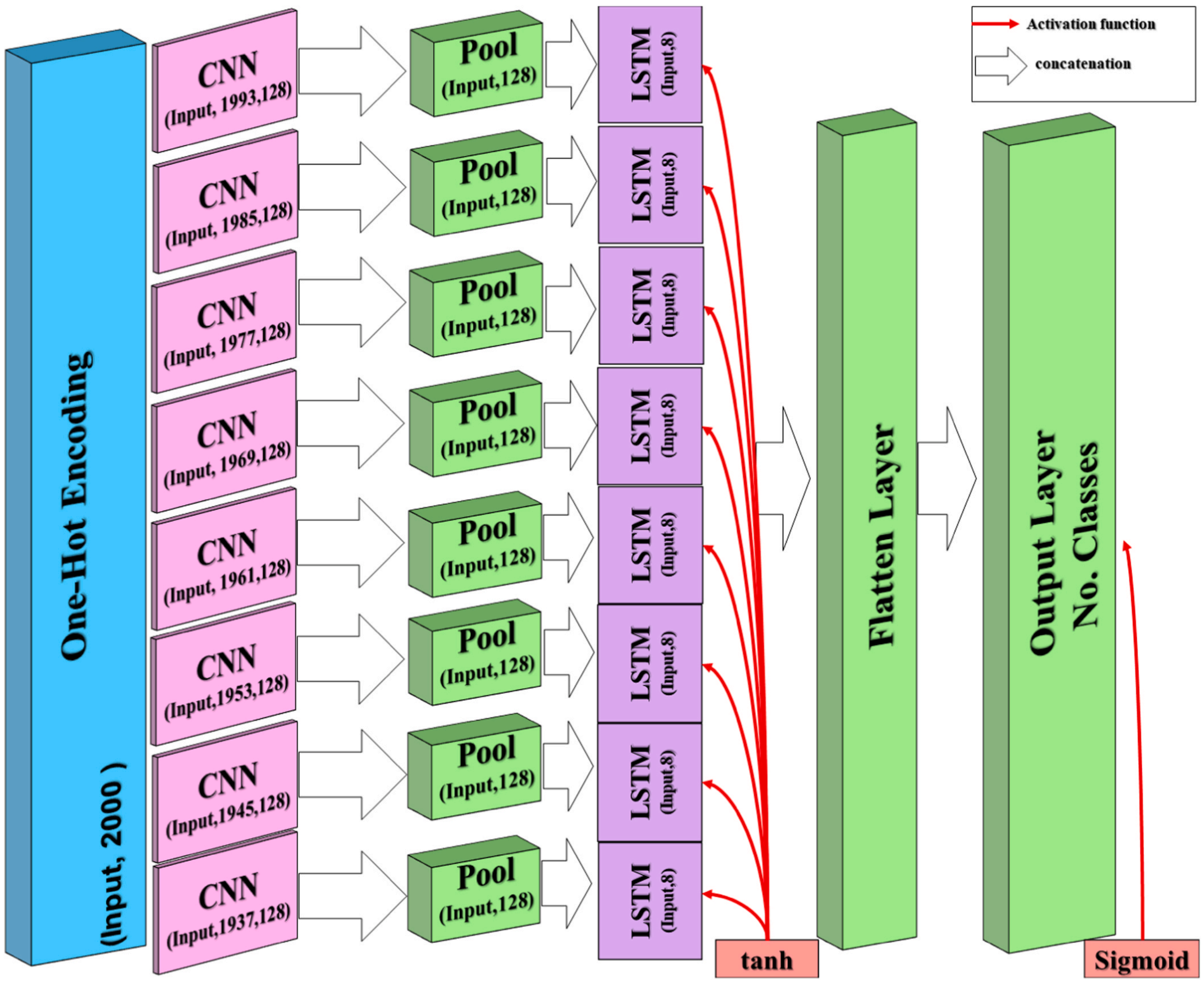


Fig. 10. : Proposed Deep_CNN-LSTM-GO model. For one hot encoding the 1st parameter is (input):- represent the input amino acid sequence, while the 2nd parameter (2000) is the max length of the input, For CNN the 1st parameter is the input represent the output from the previous layer, 2nd parameter is explains the number of filters (kernel) while the 3rd parameters is the filter (kernel) length or size.For Pooling the 1st parameter is the input represent the output from the previous layer, 2nd parameter is explains length or size of pooling layer. For LSTM the 1st parameter is the input represent the output from the previous layer, 2nd parameter is explains the number of output units.

the final output represented in GO. As shown Fig. 11 illustrates an example of what the proposed method can do, showing the input and output process.

Despite imposed restrictions applied to the proposed model in order to create a scalable model, such as the reduced number of layers, the training is performed using standard device CPU, and other parameters will be discussed in details in Section VI.

Deep_CNN_LSTM_GO can perform competitive result in some cases and outperform in others using different data sets, organisms, and evaluation metrics, as will be discussed in section V and VI.

As depicted in Fig. 10, The proposed Deep_CNN_LSTM_GO consists of six stages, described as follows:

- **First stage One Hot-Encoding** is the input stage that takes an input of the amino-acid sequences, stacking each amino acid sequence in separate space which located lndex= 1 and the others is zeros. It employs one-hot encoding with a maximum amino-acid sequence input length of 2000 different symbols.

- **Second stage 1D-CNN** is the convolutional neural network stage. It consists of eight parallel 1D-CNN architectures. These are constructed and built with the following number of filters:

$$\begin{bmatrix} \mathbf{2000-7} \\ \mathbf{1992-7} \\ \mathbf{1984-7} \\ \mathbf{1976-7} \\ \mathbf{1968-7} \\ \mathbf{1960-7} \\ \mathbf{1952-7} \\ \mathbf{1944-7} \end{bmatrix} = \begin{bmatrix} \mathbf{1993} \\ \mathbf{1985} \\ \mathbf{1977} \\ \mathbf{1969} \\ \mathbf{1961} \\ \mathbf{1953} \\ \mathbf{1945} \\ \mathbf{1937} \end{bmatrix}$$

  The maximum kernel size (filter length) of 128 per filter is used.

- **Third Stage Max-Pooling** is composed of eight Max-Pooling units attached to the outputs of the eight 1D-CNN blocks of the second stage. These units are constructed with a window size of 128.
- **Fourth stage LSTM** is allocated after each max-pooling unit. An LSTM stage is concatenated with a dimension of output space of size eight. It is activated with a "tanh" activation function.

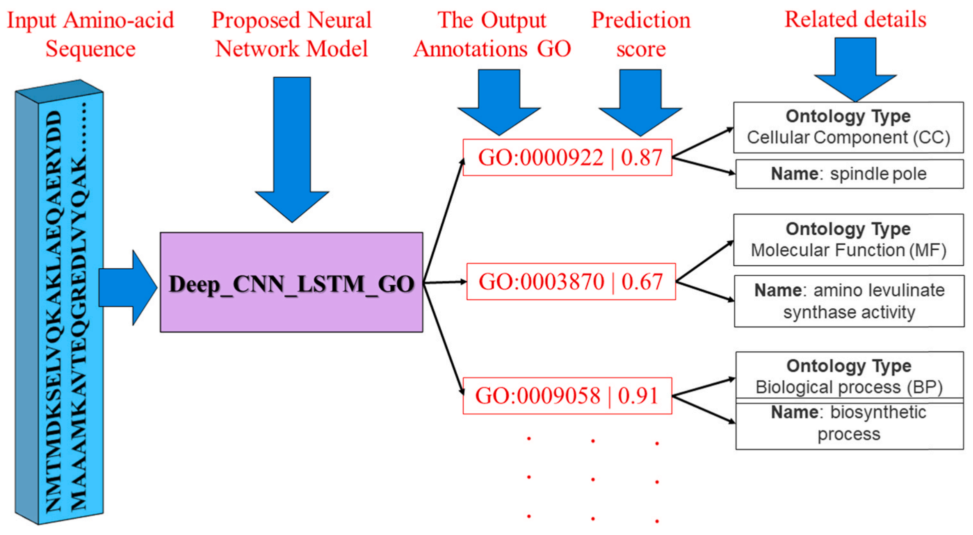

Input Amino-acid Sequence
Proposed Neural Network Model
The Output Annotations GO
Prediction score
Related details
NMTMDKSELVQKAKLAEQAERYDD MAAAMKAVTEQGREDLVYQAK.......
Deep_CNN_LSTM_GO
GO:0000922 | 0.87
Ontology Type Cellular Component (CC)
Name: spindle pole
GO:0003870 | 0.67
Ontology Type Molecular Function (MF)
Name: amino levulinate synthase activity
GO:0009058 | 0.91
Ontology Type Biological process (BP)
Name: biosynthetic process

Fig. 11. Illustrated example of what the proposed method can do, showing the input and output process.

- **Fifth Stage Flatten** All outputs from the LSTM stage are directed as inputs to each one of the eight Flatten blocks forming this stage.
- **Sixth stage output** takes its inputs from the outputs of the eight blocks of the Flatten stage. It contains neurons activated with the Sigmoid activation function to add non-linearities to the output. The number of the output neurons is equal to the number of required functions of 4600.

The model is optimized throughout the iterative process. During the training phase, Deep_CNN_LSTM_GO is needed to fine-tune the hyper-parameters to get the best results carefully. The training validation and test sets were split to be 90%, 9% and 1% respectively we used this distribution to be same as done in DeepGOPlus for further comparisons to fair.

Model error is computed using the binary cross-entropy cost/loss function (Goodfellow et al., 2016). Binary cross entropy performed well against the independent data and fits with the multi-class classification tasks such as protein function prediction tasks, where sequences represent the input data and the output data has one or more GO (class).

The weights of the proposed Neural Networks model are updated using the Adam-optimization algorithm (Sak et al.,). Adam-optimization algorithm is chosen because it combines the best properties of two state-of-the-art optimization algorithms (Kingma and Ba, 2014) and (Duchi, Jul 1 et al., 2011). It is recorded in the works of literature and many surveys (Duchi, Jul 1 et al., 2011) that most authors used Adam-optimization because it performs well against the non-convex optimization problems (Sak et al.). It also performs well with the sparse noisy gradients and it consumes less Memory compared to other optimization algorithms (Duchi et al., 2011).

Batch Gradient Descent (BGD) method (Ruder, 2016) is used to train the data. BGD takes the average of batch gradients of all training examples in just one step. Averaging gradients of all samples in training data helps the cost function smoothly diverge to the global minimum (Duchi et al., 2011; Ruder, 2016; Bottou et al., 2018). In addition, the early stop technique is used during the training process to prevent overfitting.

As discussed previously, 90% of the dataset were selected to train the neural networks model for the training phase. The weights are first initialized randomly to prevents the output neurons from vanishing or exploding. At the beginning of the training step, the performance of the neural networks is worst. Then it starts to be improve and it is optimized using ADAM optimization. The performance is evaluated by the system using the loss function, where this value is minimized using the cross-entropy in order to find the best accuracy. The training phase is started with 1024 Epoch, each Epoch has 1605 steps. After each completed epoch the validation loss is calculated. Then the optimization process is iteratively updated using BGD until the loss function is minimized.

The training phase is be terminated if one of two scenarios occurs. The first scenario is that the training and validation loss functions are not optimized after 6 epoch. Then using early stop will terminate the training phase with the latest optimized loss functions. The second scenario takes place when the number of Epochs is finished.

After the process is finished a final loss function is calculated using a 10% test set. This is the test loss value. The model concludes with a test loss value of 0.027383. The final training and validation loss functions are 0.0359, and 0.0460 respectively.

Deep_CNN_LSTM_GO model is trained and designed using the new version of TensorFlow that includes Keras framework (Abadi et al., 2016; https://www.tensorflow.org/) using python programming languages. The proposed model is trained on an Intel-Core i7–6700 machine, with 8 GB of RAM.

5. Evaluation metrics

For evaluating the proposed Deep_CNN_LSTM_GO, five different evaluation metrics were proposed and implemented. These are namely Precision, recall, Fmax, Smin, and AUPR. Each of these evaluation metrics is computed for each of the three protein functions (MF, BP, and CC).

To calculate and evaluate the model using those metrics, a Confusion Matrix (Stehman, 1997) is constructed first. Table 3 shows the Confusion Matrix parameters.

5.1. Precision

Precision is a tool that measures how precise or accurate the proposed model is out of the predicted positive which counts how many from the predictions are true. As shown in Eq. 3, the model's Precision is calculated from the numbers of True Positive (TP) and the number of

Table 3
Confusion Matrix.

		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

False Positive (FP).

$$\text{Precision} = \frac{\text{TP}}{\text{TPP}} \tag{3}$$

where, the Total Predicted Positive (TPP) is given by:

$$\text{TPP} = \text{TP} + \text{FP}$$

5.2. Recall

Recall measures the sensitivity of the model. Sensitivity refers to the actual predicted positive, as shown in Eq. 4 for the True Positive (TP) False Negative (FN).

$$\text{Recall} = \frac{\text{TP}}{\text{TAP}} \tag{4}$$

where, the Total Actual Positive (TAP) is given by:

$$\text{TAP} = \text{TP} + \text{FN}$$

5.3. Fmax

F-1 score is a balance between Precision and recall. It is the harmonic mean of Precision and recall. Fmax score is calculated using Eq. 5.

$$\text{Fmax} = \text{Max}\left\{\frac{\text{2xPrecisionxRecall}}{\text{Precision} + \text{Recall}}\right\} \tag{5}$$

5.4. S_{min}

$S_{\min}$ is another evaluation metric (Clark and Radivojac, 2013). It measures the semantic distance between real and predicted annotation based on the classes' Information Content (IC). This method is used in Deep_Go_Plus (Kulmanov and Hoehndorf, 2020), Go-Labeler (You et al., 2018a), and DeepText2Go (You et al., 2018b).

Each of the three protein functions (MF, BP, and CC) constitutes a class (C). For each of these classes, a subgraph (T) with its nodes (ν) is drawn.

The Information Content (IC) for a class (C) at node (ν) of subgraph (T) can be thought of as the number of bits of information one would receive about a protein if it were annotated with that particular subgraph (Clark and Radivojac, 2013).

IC at a particular node of a subgraph can be calculated in a straightforward manner using the inverse logarithm of the product of all prediction probabilities (Pre) for class (C) given by parents of this class (C) as shown in Eq. 6.

$$\text{IC}_C = -\log\left[\prod_{\nu \in T} Pre(C)parents)\right] \tag{6}$$

The Remaining Uncertainty (RU) is calculated at a certain gene ontology node ($i, where, i = 1..n$). RU is simply the total Information Content at node i, which is contained in True (T_i) but not in Predicted (P_i) annotation graphs.

The Average Remaining Uncertainty (ARU) is calculated by dividing the sum of all remaining uncertainties by the total number of gene ontology nodes (n) for a class (C) as shown in Eq. 7.

$$\text{ARU}_C = \frac{1}{n}\sum_{i=1}^{n}\sum_{T_i - P_i} IC_C \tag{7}$$

On the other hand, the Mis-Information (MI) introduced by the classifier corresponds to the total information content at a certain gene ontology node ($i, where, i = 1..n$) along incorrect paths in Prediction (P_i), but not in True (T_i) annotation graphs.

The Average Mis-Information (AMI) is calculated similarly to ARU and may be expressed by the following equation for a certain class (C) as shown in Eq. 8.

$$\text{AMI}_C = \frac{1}{n}\sum_{i=1}^{n}\sum_{P_i - T_i} IC_C \tag{8}$$

Finally, S_{min} is calculated for class (C) as in Eq. 9.

$$\text{S}_{\min C} = \text{Min}\left(\sqrt{(AMI_C)^2 + (ARU_C)^2}\right) \tag{9}$$

5.5. AUPR (area under Precision-Recall curve)

Precision-Recall curve shows the balance between the Precision and recall for different thresholds. A higher AUPR refers to that both Precision and recall are high. A high precision relates to a low false-positive rate. Alternatively, high recall relates to a low false-negative rate. In other words, high Precision and high recall mean that the proposed system returns many results, and the majority of them are correct.

6. Results and biological evaluation

In the evaluation results, five different comparisons are performed to evaluate the proposed Deep_CNN_LSTM_GO algorithm against different models proposed in the field. Three different datasets: UniProtKB-SwissProt's, CAFA3 and CAFA4 were used here. In the next five sub-sections Deep_CNN_LSTM_GO is evaluated using different These are presented in the following five sub-sections.

6.1. UniProtKB-Swissport comparison (MF, BP, CC)

The proposed Deep_CNN_LSTM_GO is tested for the selected test-set generated from UniProtKB-Swissprot. The selected samples in the test set are 100 different samples, from which none are included in the training set. The evaluation is performed against three different methods proposed in the field, GO-Labeler (You et al., 2018a), Deep_GO (Kulmanov and Khan, 2018) and Deep_Go_Plus (Kulmanov and Hoehndorf, 2020). As shown in Fig. 12, the proposed method is compared and tested against each one of the three sub-ontologies (MF, BP, CC) using Fmax, Smin, and AUPR performance metrics. The comparison results of most evaluations are summarised in Table 4 for clarity and convenience.

For Fmax, Deep_CNN_LSTM_GO performs 4th in MF and BP sub-ontology cases with 0.407, and 0.356 scores. Deep_CNN_LSTM_GO ranked 3rd to be outperformed by Deep_GO (Kulmanov and Khan, 2018) in the CC sub-ontology with a score of 0.675, and almost in tie with Deep_GO_Plus (Kulmanov and Hoehndorf, 2020) that placed the 1st with a negligible score difference of 0.024.

For Smin, the proposed model came 4th in MF and CC sub-ontologies with scores of 10.99 and 12.06, respectively. It ranked 2nd place to be outperformed by Deep_Go (Kulmanov and Khan, 2018), and Deep_-Go_Plus (Kulmanov and Hoehndorf, 2020) in the BP sub-ontology with a score of 24.41.

For the AUPR evaluation metric, Deep_CNN_LSTM_GO outperformed Deep_GO (Kulmanov and Khan, 2018) and Go-labeler (You et al., 2018a) in CC with a score of 0.721 and almost in a tie with Deep_Go_Plus (Kulmanov and Hoehndorf, 2020) with a score difference of just 0.005. It, however, ranked 3rd for the BP sub-ontology to be outperformed by Go-labeler (You et al., 2018a) with a score of 0.306 and very close to Deep_GO (Kulmanov and Khan, 2018) with a difference of 0.026. It ranked 4th for the MF sub-ontology with a score of 0.280.

6.2. CAFA3 comparison against FFPred and Go-FDR (human organism) (MF, BP, CC)

CAFA3 is used to evaluate the proposed method against FFPred (Cozzetto et al., 2016) and Go-FDR (Gong et al., 2016). As mentioned earlier, these techniques are in the top and most cited methods in the field The comparison is performed using the performance metrics: Fmax,

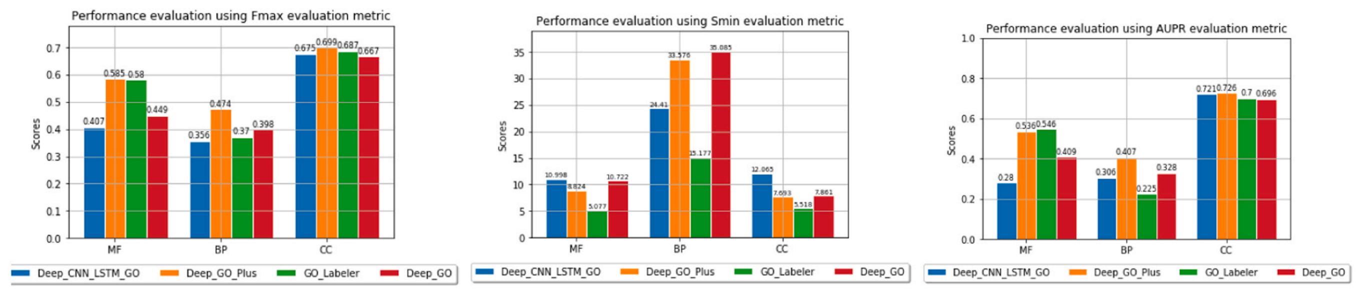


Fig. 12. UniProtKB Swissport performance comparison of the proposed method (Deep_CNN_LSTM_GO) with GO_Labeler, Deep_GO, and Deep_GO_Plus for the three cases of MF, BP and CC.

Table 4
Evaluation of the proposed Deep_CNN_LSTM_GO technique against different techniques proposed in the field for different datasets.

Dataset	Algorithm	Performance evaluation metric								
		Fmax			Smin			AUPR		
		MF	BP	CC	MF	BP	CC	MF	BP	CC
UniprotKB-SwissProt	GO-Labeler	0.580	0.370	0.687	5.077	15.177	5.518	0.546	0.225	0.700
	Deep GO	0.449	0.398	0.667	10.722	35.085	7.861	0.409	0.328	0.696
	Deep Go Plus	0.585	0.474	0.699	8.824	33.576	7.693	0.536	0.407	0.726
	Deep_CNN_LSTM_GO	**0.407**	**0.356**	**0.675**	**10.998**	**24.41**	**12.065**	**0.28**	**0.306**	**0.721**
	Organism name	**Fmax**			**Smin**			**AUPR**		
		MF	**BP**	**CC**	**MF**	**BP**	**CC**	**MF**	**BP**	**CC**
CAFA4	Human	0.113	0.215	0.443	9.015	38.025	9.175	0.0501	0.107	0.294
	Arabidopsis thaliana	0.32	0.191	0.507	6.714	32.612	11.562	0.039	0.091	0.412
	Danio rerio	0.1	0.189	0.444	16.03	31.101	5.909	0.0509	0.109	0.32
	Mus musculus Linnaeus	0.1	0.245	0.492	9.031	48.187	14.327	0.042	0.14	0.378
	Rattus norvegicus	0.108	0.26	0.368	16.565	49.348	15.844	0.0401	0.129	0.234
	Dictyostelium discoideum	0.081	0.23	0.584	5.871	44.309	12.866	0.042	0.136	0.468
	Escherichia coli K-12	0.082	0.118	0.094	11.832	26.557	6.111	0.03	0.049	0.032
	Schizosaccharomyces pombe 972 h	0.173	0.236	0.618	9.364	26.325	12.964	0.064	0.128	0.565
	Average	**0.1346**	**0.2105**	**0.44375**	**10.552**	**37.058**	**11.094**	**0.0447**	**0.1111**	**0.3378**
CAFA3	**Algorithm**	**Fmax**			**Precision**			**Recall**		
		MF	**BP**	**CC**	**MF**	**BP**	**CC**	**MF**	**BP**	**CC**
	FFPred3	0.38	0.26	0.44	0.35	0.301	0.41	0.4	0.23	0.43
	GoFDR	0.52	0.2	0.40	0.89	0.27	0.40	0.36	0.15	0.41
	Deep_CNN_LSTM_GO	**0.115**	**0.322**	**0.455**	**0.0701**	**0.401**	**0.46**	**0.3247**	**0.269**	**0.455**

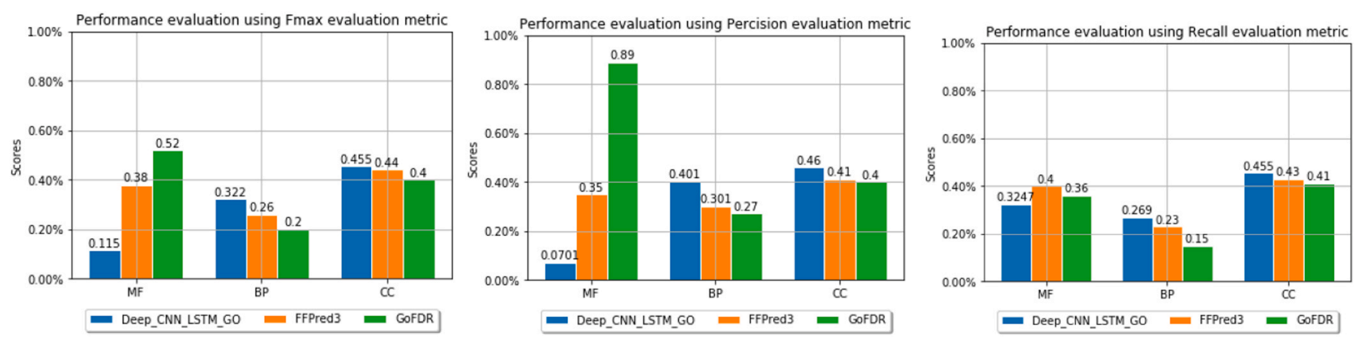


Fig. 13. CAFA3 performance comparison of the proposed method (Deep_CNN_LSTM_GO) with FFPred3 and GoFDR (human organism) for the three cases of MF, BP and CC.

Precision and Recall for each subontology (MF, BP, CC) on the human organism. Smin and AUPR not be choosen here bacuase FFpred (Cozzetto et al., 2016) and Go-FDR (Gong et al., 2016) didn't use them in the evaluation. The results are shown in Fig. 13 and the comparisons are presented in Table 4.

For Fmax, Deep_CNN_LSTM_GO outperformed FFpred (Cozzetto et al., 2016) and Go-FDR (Gong et al., 2016) in BP and CC sub-ontologies with scores of 0.322, and 0.45, respectively. In the MF sub-ontology case, it ranked 3rd with a score of 0.115.

In Precision and Recall, Deep_CNN_LSTM_GO outperformed FFpred (Cozzetto et al., 2016) and Go-FDR (Gong et al., 2016) for the BP and CC sub-ontologies with scores of 0.401, 0.46 and 0.693, 0.455 respectively. It came in 3rd place in the case of MF sub-ontology, with scores of 0.070, 0.325.

6.3. Performance of the proposed method for the four datasets (UniProtKB Swissport, CAFA4, CAFA3, CAFA2) combined (MF, BP, CC)

The proposed method is compared against itself for the four test datasets (UniProtKB Swissport CAFA2, CAFA3, and CAFA4). The performance metrics comparison included: Fmax, Smin, and AUPR for each one of the three sub ontologies (MF, BP, CC). The results are shown in Fig. 14, and the comparisons are presented in Table 4.

It is clear that the results for the Uniprot-Swissprot dataset

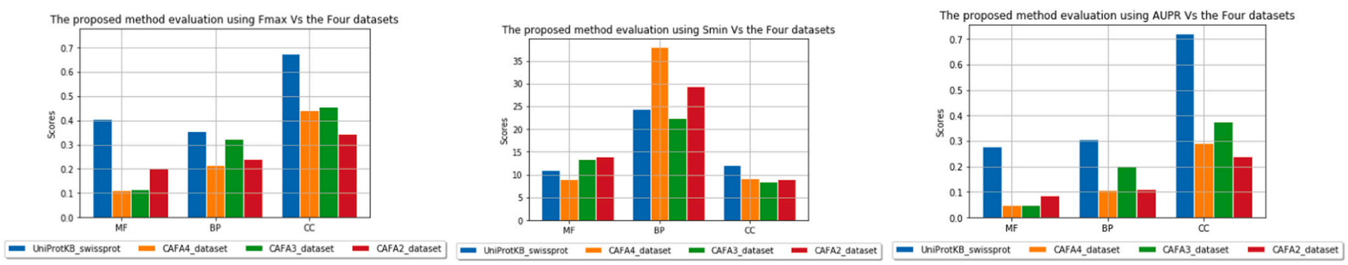


Fig. 14. Performance comparison of the proposed method (Deep_CNN_LSTM_GO) using the four datasets (UniProtKB Swissport, CAFA4, CAFA3, and CAFA2) for the three cases of MF, BP and CC.

outperformed the results for the CAFA series for Fmax and AUPR with scores of (0.407, 0.356, 0.675) and (0.28, 0.306, 0.721), respectively.

Moreover, it is clear from the graphs of Fig. 14 for the CAFA series, that the proposed model performs very close to the results of the three datasets on different organisms (human for CAFA4 and CAFA 3) and (bacteria for CAFA2). The performance metric involved is Fmax in CC sub-ontology with (0.443, 0.450, 0.484) values. This, in effect, shows the robustness of the proposed model accuracy in prediction. Note that the datasets have been proposed at different times as mentioned earlier.

6.4. CAFA4 comparison against eight different organisms (MF, BP, CC)

CAFA4 is used to evaluate the proposed method's performance for eight different organisms using Fmax, Smin, and AUPR for all the sub-ontology cases (MF, BP, CC). As shown in Table 4 the proposed model performed good results in different organisms,

For Fmax the proposed model performs 0.618 in CC which close to the results in UniprotKB-SwissProt in the CC 0.675, which leads that the proposed model can perform the same results to the other organisms or close to it. Another note to be taken from the results in CAFA4 that the scores for the 8 organisms are very close to each other which means that the proposed model can perform closely the same results for different organisms that have different amino acid sequence characteristics and features that the human.

6.5. Hyperparameters comparisons of the proposed method with other models (Deep_GO, Deep_Go_Plus)

To test the hyperparameters and the model design, the proposed method is compared against the three different methods proposed in the field: Deep_GO (Kulmanov and Khan, 2018), Deep_Go_Plus (Kulmanov and Hoehndorf, 2020). The hyperparameters used include the number of layers for each model, architectures used, trainable parameter sizes and activation functions. These parameters are presented in Table 5.

Table 5 shows that the proposed model is the only approach that can combine two different Neural Network architectures with the lowest number of trainable parameters and model complexity.

The proposed model is the only model in the comparison that can be trained on any standard CPU without requiring a dedicated GPU which will be explained in the following parameters that show the difference and the impact with reasonable numbers.

As shown in Table 6, a comparison shows the impact of the huge

Table 6
Comparision between the CPU and other versions of GPU's.

Comparison parameter	Intel Core i7-6700 CPU (HD Graphics 530)	Nvidia GTX 1060 3 GB RAM	Nvidia GeForce GTX TITAN X GPU	Nvidia GeForce GTX TITAN Z GPU
Speed Rank (https://gpu.userbenchmark.com/)	361st/654	74th/654	53rd/654	70th/654
CUDA Cores	no data	1280	3072	5760
Core clock speed	350 MHz	1506 MHz	1000 MHz	705 MHz
Memory Limits	1 GB	2.5 GB	12 GB	12 GB
Floating-point performance	403.2 gflops	4275 gflops	6691 gflops	2 × 5046 gflops
Memory clock speed	Up to 2133 MHz	8000 MHz	7.0 GB/s	7.0 GB/s

Table 5
Hyper-parameters comparisons of the proposed Deep_CNN_LSTM_GO technique against different deep learning methods proposed in the field (Deep_Go_Plus, Deep_GO).

Comparison parameter	Deep_CNN_LSTM_GO	Deep_Go_Plus	Deep_GO
Number of layers	16	46	20
No of trainable parameters (nearly number)	1 M	50 M	15 M
Trained classes size	12157 MF, 30851 BP, 4396 CC	10693 MF, 29264 BP,4034 CC	10693 MF, 29228 BP, 4033 CC
Number of output classes	4600	4774	1024
Model type used	Convolutional Neural Network+LSTM	Convolutional Neural Network	Convolutional Neural Network
Combined with other methods	None	Similarity based method	Hierarchical classification
Combined methodology	None	Weighted sum model	None
Optimizer	Adam-Optimization	Adam optimization	RMSProp optimizer
Activation functions used	TanhSigmoid	SigmoidReLU	Sigmoid
Over fitting prevention tech's	One hot encodingEarly stop	One hot encodingEarly stop	DropoutEarly stop
Pooling layer	Yes	Yes	Yes
Hardware used	Intel Core i7–6700 CPUHD Graphics 530 GPU	. 2 X Nvidia Titan X and P6000 GPUs with 12–24 Gb of RAM.	CPU 15 GB RAM+ Nvidia GeForce GTX TITAN Z GPU
Frame work used for the training	TensorFlow 2.1& Keras	TensorFlow 1.14& Keras	TensorFlowKeras
Drop out	No	No	Yes
Data types used	Amino-acid sequences	Amino-acid sequences	Amino-acid sequences +Protien-Protien Interaction Network (PPIN)

difference between the employed CPU and the NVIDIA GTX1060–3 GB-RAM GPU which is considered one of the simplest GPUs in the market. In addition, the comparison includes other upgraded GPU's versions.

The comparison in Table 6 shows that the employed CPU in this research is nearly 4 times slower than the GTX1060 3 GB in terms of speed parameter (https://gpu.userbenchmark.com/). While the Nvidia GeForce GTX TITAN Z GPU is 6 times slower, and 7 times slower than Nvidia GeForce GTX TITAN X GPU.

For the CUDA cores it's clear that the employed CPU has no CUDA cores which means no parallel processes will occur during the training, while the Nvidia GTX 1060 has 1280, Nvidia GeForce GTX TITAN X GPU has 3072, and Nvidia GeForce GTX TITAN Z GPU has 5760. The more CUDA cores in the systems, the more parallel processes occur, therefore the computation time decrease.

For the Memory limit, The employed CPU has 1 GB memory, while 2.5, 12, 12 Memory limits for Nvidia GTX 1060, Nvidia GeForce GTX TITAN X, Nvidia GeForce GTX TITAN Z GPU respectively. The more Memory, the more tensors can be installed and concatenated, and the trainable parameters will be increased. This leads to the ability to insert more layers with large sizes and kernels in the Neural Networks. Therefore the Neural Networks become very deep.

Core clock speed represents how fast a single core can perform a single task. Its noticed in Table 6 the employed CPU is nearly 4 times slower than the GTX 1060, and 3 times slower than GTX TITAN X, and two times slower than GTX TITAN Z.

Floating-point arithmetic is needed for very large or small real numbers that are calculated in multiple matrices. The more floating-point operations per second, the faster the model to finish the computations. As shown in Table 6 the used CPU can finish nearly 400 giga-flops per second which means ten times slower than GTX 1060 that can finish 4275 giga-flops per second. While 15 times slower than GTX TITAN X and 14 times than GTX TITAN Z.

The proposed model is scalable and expandable. In other words, the Amino-Acid input sequence length can be increased as well as the size of the GO to be predicted. Therefore, the number of layers, kernel, and filter sizes can be extended and increased, producing remarkable results with the lowest complexity compared to other methods.

7. Discussion

In the previous section, we discussed the five different evaluation methods done using the different datasets as well as different parameters.

First, the proposed model is **robust**, it was compared using UniProtKB-SwissProt's, and CAFA-series datasets, which are considered as the main benchmark datasets for protein function prediction evaluation. To achieve a solid comparison, we used multiple datasets which is not the case in most of the recent papers in bioinformatics for protein function prediction contributions as discussed in the related work section. In addition, this research uses three different evaluation metrics for each dataset during the comparison, in order to ensure and prove that the proposed model outperformed the others using different methods.

Second, the proposed model is **scalable**, as it combines two different neural network architectures, CNN and LSTM, to perform the same task, taking advantage of both architectures to overcome the limitations of using one of them alone. This enhanced the results as shown in the previous section. This integration gives the ability to the proposed model to be combined with other Neural Networks architectures for further enhancements. This could make the proposed model to be considered a state-of-the-art approach in the near future.

Third, the proposed hyper model is **dynamic**. Can be applied to other different applications in protein function prediction. The reason behind that, it integrates two different architectures, CNN and LSTM, which makes it easier to apply additional modifications to be used in different such as protein family classification, 3D structure prediction, protein-protein interaction networks and more, using different data types, images or sequences or both.

Fourth the proposed model is **compact**, having the lowest model complexity, trainable parameters and hardware support as mentioned and discussed earlier. Deep_CNN_LSTM_GO outperformed the other methods with this regard, which likely would give the proposed model the priority in the future to be adopted from the research community and institutions for future research.

8. Conclusion

The proposed Deep_CNN_LSTM_GO is an accurate model for protein function prediction. This paper presented the proposed method and compared it to different methods proposed in the field in terms of several performance evaluation criteria: Precision, recall, Fmax, Smin, AUPR, hardware requirements, model complexity, etc .Deep_CNN_LSTM_GO was tested using different organisms from different datasets (UniProt-SwissProt's, CAFA2, CAFA3, and CAFA4).

The novality of the proposed method combined the functionalities of CNN and LSTM neural network models in order to reduce the number of layers in the deep learning model. The amount of computations required was also significantly reduced by the use of 1D-CNN. This resulted in the fact that the proposed Deep_CNN_LSTM_GO was trained on any standard CPU without the need for a dedicated GPU. We predict that our contribution will lead to rapid advances in the bioinformatics field as it will allow researchers without expensive hardware resources to explore and advance this field.

Tabulated test results showed that the proposed Deep_CNN_LSTM_GO outperformed different methods proposed in the field in sub-ontologies and comparable results for the others using different test sets. The proposed model was able to annotate 4600 annotations for a single unknown protein in minutes.

Deep_CNN_LSTM_GO can be used with minor modifications, to be extended and trained for different data sources such as protein 3D structures or a combination of two different data sources to be applied on different tasks such as phenotype diseases prediction from amino-acid sequences. This is the subject of another publication.

References

Abadi M., Barham P., Chen J., Chen Z., Davis A., Dean J., Devin M., Ghemawat S., Irving G., Isard M., Kudlur M. Tensorflow: A system for large-scale machine learning. In12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16) 2016 (pp. 265–283).

Abdel-Hamid, O., Deng, L., Dong Yu, 2013. "Exploring convolutional neural network structures and optimization techniques for speech recognition". Interspeech.

Alejandro, A., Schäffer, Yuri, I., Wolf, Chris, P., Ponting, Eugene, V., Koonin, L., Aravind, Stephen, F.Altschul, 1999. IMPALA: matching a protein sequence against a collection of PSI-BLAST-constructed position-specific score matrices. Bioinformatics 15 (12), 1000–1011. https://doi.org/10.1093/bioinformatics/15.12.1000.

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J., 1990. "Basic local alignment search tool". J. Mol. Biol. 215, 403–410.

Altschul, Stephen F., Madden, Thomas L., Schäffer, Alejandro A., Zhang, Jinghui, Zhang, Zheng, Miller, Webb, Lipman, David J., 1997. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. 25 (17), 3389–3402. https://doi.org/10.1093/nar/25.17.3389.

Tensorflow 2015, Anonhttps://www.tensorflow.org/.

Anon https://gpu.userbenchmark.com/.

Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M., Sherlock, G., 2000. Gene ontology: tool for the unification of biology. Nat. Genet 25, 25–29. https://doi.org/10.1038/75556.

Berman, Helen M., Westbrook, John, Feng, Zukang, Gilliland, Gary, Bhat, T.N., Weissig, Helge, Shindyalov, Ilya N., Bourne, Philip E., 2000. The protein data bank. Nucleic Acids Res. 28 (1), 235–242. https://doi.org/10.1093/nar/28.1.235.

Bottou, L.éon, Frank, E.Curtis, Nocedal, Jorge, 2018. "Optimization methods for large-scale machine learning". SIAM Rev. 60, 223–311.

Chollet, François, 2017. Xception: Deep learning with depthwise separable convolutions, Proceedings of the IEEE conference on computer vision and pattern recognition, 2017.

Clark, W.T., Radivojac, P., 2013. Information-theoretic evaluation of predicted ontological annotations. Bioinformatics 29 (13), i53–i61.

Clark, Wyatt T., Radivojac, Predrag, 2011. "Analysis of protein function and its prediction from amino acid sequence". Protein. Struct. Funct. Bioinform. 79 (7), 2086–2096.

Cozzetto, D., Minneci, F., Currant, H., Jones, D.T., 2016. FFPred 3: feature-based function prediction for all Gene Ontology domains. Sci. Rep. 6, 31865. https://doi.org/10.1038/srep31865.

Duchi, J., Hazan, E., Singer, Y., 2011. Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. 12, 7.

Fa, R., Cozzetto, D., Wan, C., Jones, D.T., 2018. Predicting human protein function with multi-task deep neural networks. PLOS ONE 13 (6), 0198216. https://doi.org/10.1371/journal.pone.0198216.

Fischer, Markus, Pleiss, J.ürgen, 2003. "The lipase engineering database: a navigation and analysis tool for protein families". Nucleic Acids Res. 31 (1), 319–321.

A. Giusti, D.C. Cireşan, J. Masci, L.M. Gambardella, J. Schmidhuber, 2013. Fast image scanning with deep max-pooling convolutional neural networks, 2013 IEEE International Conference on Image Processing, Melbourne, VIC, pp. 4034–4038, doi: 10.1109/ICIP.2013.6738831.

Gong, Qingtian, Ning, Wei, Tian, Weidong, 2016. "GoFDR: a sequence alignment based method for predicting protein functions". Methods 93, 3–14.

Goodfellow, Ian, Bengio, Yoshua, Courville, Aaron, Goodfellow, 2016. Deep Learning. MIT Press. http://www.deeplearningbook.org.

GPCRdb in 2021: integrating GPCR sequence, structure and function Kooistra AJ, Mordalski S, Pándy-Szekeres G, Esguerra M, Mamyrbekov A, Munk C, Keserű GM, Gloriam DE Nucleic Acids Research, 2020, X:X.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., Chen, T., 2018. Recent advances in convolutional neural networks. Pattern Recognit. 77, 354–377. May 1.

Hochreiter, Sepp, Schmidhuber, J.ürgen, 1997. "Long short-term memory". Neural Comput. 9, 1735–1780.

Jiang, Y., Oron, T.R., Clark, W.T., Bankapur, A.R., D'Andrea, D., Lepore, R., Funk, C.S., Kahanda, I., Verspoor, K.M., Ben-Hur, A., Koo da, C.E., Penfold-Brown, D., Shasha, D., Youngs, N., Bonneau, R., Lin, A., Sahraeian, S.M., Martelli, P.L., Profiti, G., Casadio, R., Cao, R., Zhong, Z., Cheng, J., Altenhoff, A., Skunca, N., Dessimoz, C., Dogan, T., Hakala, K., Kaewphan, S., Mehryary, F., Salakoski, T., Ginter, F., Fang, H., Smithers, B., Oates, M., Gough, J., Törönen, P., Koskinen, P., Holm, L., Chen, C.T., Hsu, W.L., Bryson, K., Cozzetto, D., Minneci, F., Jones, D.T., Chapman, S., Bkc, D., Khan, I.K., Kihara, D., Ofer, D., Rappoport, N., Stern, A., Cibrian-Uhalte, E., Denny, P., Foulger, R.E., Hieta, R., Legge, D., Lovering, R.C., Magrane, M., Melidoni, A.N., Mutowo-Meullenet, P., Pichler, K., Shypitsyna, A., Li, B., Zakeri, P., ElShal, S., Tranchevent, L.C., Das, S., Dawson, N.L., Lee, D., Lees, J. G., Sillitoe, I., Bhat, P., Nepusz, T., Romero, A.E., Sasidharan, R., Yang, H., Paccanaro, A., Gillis, J., Sedeño-Cortés, A.E., Pavlidis, P., Feng, S., Cejuela, J.M., Goldberg, T., Hamp, T., Richter, L., Salamov, A., Gabaldon, T., Marcet-Houben, M., Supek, F., Gong, Q., Ning, W., Zhou, Y., Tian, W., Falda, M., Fontana, P., Lavezzo, E., Toppo, S., Ferrari, C., Giollo, M., Piovesan, D., Tosatto, S.C., Del Pozo, A., Fernández, J.M., Maietta, P., Valencia, A., Tress, M.L., Benso, A., Di Carlo, S., Politano, G., Savino, A., Rehman, H.U., Re, M., Mesiti, M., Valentini, G., Bargsten, J. W., van Dijk, A.D., Gemovic, B., Glisic, S., Perovic, V., Veljkovic, V., Veljkovic, N., Almeida-E-Silva, D.C., Vencio, R.Z., Sharan, M., Vogel, J., Kansakar, L., Zhang, S., Vucetic, S., Wang, Z., Sternberg, M.J., Wass, M.N., Huntley, R.P., Martin, M.J., O'Donovan, C., Robinson, P.N., Moreau, Y., Tramontano, A., Babbitt, P.C., Brenner, S.E., Linial, M., Orengo, C.A., Rost, B., Greene, C.S., Mooney, S.D., Friedberg, I., Radivojac, P., 2016. An expanded evaluation of protein function prediction methods shows an improvement in accuracy. Genome Biol. 17, 184. https://doi.org/10.1186/s13059-016-1037-6.

Jonghoon, Jin, Dundar, Aysegul, Culurciello, Eugenio, 2014. Flattened convolutional neural networks for feedforward acceleration, 1412, 5474.

José Juan Almagro Armenteros, Casper Kaae S.ønderby, Søren Kaae Sønderby, Henrik Nielsen, Winther, Ole, 2017. DeepLoc: prediction of protein subcellular localization using deep learning. Bioinformatics 33 (21), 3387–3395. https://doi.org/10.1093/bioinformatics/btx431.

Kawulok, J., Deorowicz, S., 2015. CoMeta: classification of metagenomes using k-mers. PloS One 10 (4), e0121453.

Keedwell, Edward, Narayanan, Ajit, 2005. Intelligent Bioinformatics: The Application of Artificial Intelligence Techniques to Bioinformatics Problems. John Wiley & Sons.

Kingma D.P., Ba J. Adam: A method for stochastic optimization. arXiv:1412.6980. 2014 Dec 22.

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., Inman, D.J., 2019. 1D convolutional neural networks and applications: a survey, 1905, 03554.

Krizhevsky, Alex, Sutskever, Ilya, Geoffrey, E.Hinton, 2017. ImageNet classification with deep convolutional neural networks. Commun. ACM 60 (6), 84–90. https://doi.org/10.1145/3065386.

Kulmanov, Maxat, Hoehndorf, Robert, 2020. Deep_Go_Plus: improved protein function prediction from sequence. Bioinformatics 36 (2), 422–429. https://doi.org/10.1093/bioinformatics/btz595.

Kulmanov, Maxat, Khan, Mohammed Asif, 2018. Robert Hoehndorf, Deep_GO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. Bioinformatics 34 (4), 660–668. https://doi.org/10.1093/bioinformatics/btx624.

M.M. Lau, K.H. Lim, 2017. Investigation of activation functions in deep belief network, 2017 2nd International Conference on Control and Robotics Engineering (ICCRE), Bangkok, pp. 201–206, doi: 10.1109/ICCRE.2017.7935070.

Li, Jinong, Zhang, Zhen, Rosenzweig, Jason, Wang, Young Y., Chan, Daniel W., 2002. Proteomics and bioinformatics approaches for identification of serum biomarkers to detect breast cancer. Clin. Chem. 48 (8), 1296–1304. https://doi.org/10.1093/clinchem/48.8.1296.

Lindsay, M., 2003. Target discovery. Nat. Rev. Drug Discov. 2, 831–838. https://doi.org/10.1038/nrd1202.

Minneci, F., Piovesan, D., Cozzetto, D., Jones, D.T., 2013. FFPred 2.0: improved homology-independent prediction of gene ontology terms for eukaryotic protein sequences. PLOS ONE 8 (5), 63754. https://doi.org/10.1371/journal.pone.0063754.

Mitchell, Alex, Chang, Hsin-Yu, Daugherty, Louise, Fraser, Matthew, Hunter, Sarah, Lopez, Rodrigo, McAnulla, Craig, McMenamin, Conor, Nuka, Gift, Pesseat, Sebastien, Sangrador-Vegas, Amaia, Scheremetjew, Maxim, Rato, Claudia, Yong, Siew-Yit, Bateman, Alex, Punta, Marco, Attwood, Teresa K., Sigrist, Christian J.A., Redaschi, Nicole, Rivoire, Catherine, Xenarios, Ioannis, Kahn, Daniel, Guyot, Dominique, Bork, Peer, Letunic, Ivica, Gough, Julian, Oates, Matt, Haft, Daniel, Huang, Hongzhan, Natale, Darren A., Wu, Cathy H., Orengo, Christine, Sillitoe, Ian, Mi, Huaiyu, Thomas, Paul D., Finn, Robert D., 2015. The InterPro protein families database: the classification resource after 15 years. Nucleic Acids Res. 43 (D1), D213–D221. https://doi.org/10.1093/nar/gku1243.

Murzin, A.G., Brenner, S.E., Hubbard, T., Chothia, C., 1995. "SCOP: a structural classification of proteins database for the investigation of sequences and structures". J. Mol. Biol. 247 (4), 536–540.

Nauman, M., Ur Rehman, H., Politano, G., Benso, A., 2019. Beyond homology transfer: deep learning for automated annotation of proteins. J. Grid Comput. 17, 225–237. https://doi.org/10.1007/s10723-018-9450-6.

Ofer, Dan, Linial, Michal, 2015. ProFET: Feature engineering captures high-level protein functions. Bioinformatics 31 (21), 3429–3436. https://doi.org/10.1093/bioinformatics/btv345.

O'Shea, Keiron, Nash, Ryan, 2015. "An introduction to convolutional neural networks", 1511, 08458.

Radivojac, P., Clark, W.T., Oron, T.R., Schnoes, A.M., Wittkop, T., Sokolov, A., Graim, K., Funk, C., Verspoor, K., Ben-Hur, A., Pandey, G., Yunes, J.M., Talwalkar, A.S., Repo, S., Souza, M.L., Piovesan, D., Casadio, R., Wang, Z., Cheng, J., Fang, H., Gough, J., Koskinen, P., Törönen, P., Nokso-Koivisto, J., Holm, L., Cozzetto, D., Buchan, D.W., Bryson, K., Jones, D.T., Limaye, B., Inamdar, H., Datta, A., Manjari, S. K., Joshi, R., Chitale, M., Kihara, D., Lisewski, A.M., Erdin, S., Venner, E., Lichtarge, O., Rentzsch, R., Yang, H., Romero, A.E., Bhat, P., Paccanaro, A., Hamp, T., Kaßner, R., Seemayer, S., Vicedo, E., Schaefer, C., Achten, D., Auer, F., Boehm, A., Braun, T., Hecht, M., Heron, M., Hönigschmid, P., Hopf, T.A., Kaufmann, S., Kiening, M., Krompass, D., Landerer, C., Mahlich, Y., Roos, M., Björne, J., Salakoski, T., Wong, A., Shatkay, H., Gatzmann, F., Sommer, I., Wass, M. N., Sternberg, M.J., Škunca, N., Supek, F., Bošnjak, M., Panov, P., Džeroski, S., Šmuc, T., Kourmpetis, Y.A., van Dijk, A.D., ter Braak, C.J., Zhou, Y., Gong, Q., Dong, X., Tian, W., Falda, M., Fontana, P., Lavezzo, E., Di Camillo, B., Toppo, S., Lan, L., Djuric, N., Guo, Y., Vucetic, S., Bairoch, A., Linial, M., Babbitt, P.C., Brenner, S.E., Orengo, C., Rost, B., Mooney, S.D., Friedberg, I., 2013. A large-scale evaluation of computational protein function prediction. Nat. Methods 10, 221–227. https://doi.org/10.1038/nmeth.2340.

Rodríguez, P., Bautista, M.A., Gonzàlez, J., Escalera, S., 2018. "Beyond one-hot encoding: Lower dimensional target embedding". Image Vis. Comput. 75, 21–31.

Ruder S. An overview of gradient descent optimization algorithms. arXiv:1609.04747. 2016 Sep 15.

Saeidnia, S., Manayi, A., Abdollahi, M., 2015. From in vitro experiments to in vivo and clinical studies; pros and cons. Curr. Drug Disco Technol. 12 (4), 218–224. https://doi.org/10.2174/1570163813666160114093140.

Saier Jr., Milton H., Tran, Can V., Barabote, Ravi D., 2006. TCDB: the transporter classification database for membrane transport protein analyses and information. Nucleic Acids Res. 34, D181–D186. https://doi.org/10.1093/nar/gkj001.

T.N. Sainath, A. Mohamed, B. Kingsbury, B. Ramabhadran, Deep convolutional neural networks for LVCSR, 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, 2013, pp. 8614–8618, doi: 10.1109/ICASSP.2013.6639347.

Sak H., Senior AW, Beaufays F., Long short-term memory recurrent neural network architectures for large scale acoustic modeling.

Sharma S., 2017. Activation functions in neural networks towards data science, 2017 Sep 6.

Sherstinsky, A., 2020. Fundamentals of recurrent neural network (rnn) and long short-term Memory (lstm) network. Phys. D Nonlinear Phenom. 404, 132306. Mar 1.

Shorten, C., Khoshgoftaar, T.M., 2019. A survey on image data augmentation for deep learning. J. Big Data 6, 60. https://doi.org/10.1186/s40537-019-0197-0.

Stehman, Stephen V., 1997. "Selecting and interpreting measures of thematic classification accuracy". Remote Sens. Environ. 62, 77–89.

Szklarczyk, D., Morris, J.H., Cook, H., Kuhn, M., Wyder, S., Simonovic, M., Santos, A., Doncheva, N.T., Roth, A., Bork, P., Jensen, L.J., von Mering, C., 2017. "The STRING database in 2017: quality-controlled protein–protein association networks, made broadly accessible". Nucleic Acids Res. 45, 362 gkw937.

Taju, Semmy Wellem, Nguyen, Trinh-Trung-Duong, Le, Nguyen-Quoc-Khanh, Kusuma, Rosdyana Mangir Irawan, Ou, Yu-Yen, 2018. DeepEfflux: a 2D convolutional neural network model for identifying families of efflux proteins in transporters. Bioinformatics 34 (18), 3111–3117. https://doi.org/10.1093/bioinformatics/bty302.

Tatusov, R.L., Galperin, M.Y., Natale, D.A., Koonin, E.V., 2000. "The COG database: a tool for genome-scale analysis of protein functions and evolution". Nucleic Acids Res. 28 (1), 33–36.

The UniProt Consortium, 2019. UniProt: a worldwide hub of protein knowledge. Nucleic Acids Res. 47 (D1), D506–D515. https://doi.org/10.1093/nar/gky1049.

Vroling, B., Thorne, D., McDermott, P., Joosten, H.J., Attwood, T.K., Pettifer, S., Vriend, G., 2012. NucleaRDB: information system for nuclear receptors (Database issue). Nucleic Acids Res. 40, D377–D380. https://doi.org/10.1093/nar/gkr960.

Wei, L., Ding, Y., Su, R., Tang, J., Zou, Q., 2018. "Prediction of human protein subcellular localization using deep learning". J. Parallel Distrib. Comput. 117, 212–217.

Wu, Cathy H., Yeh, Lai-Su.L., Huang, Hongzhan, Arminski, Leslie, Castro-Alvear, Jorge, Chen, Yongxing, Hu, Zhangzhi, Kourtesis, Panagiotis, Ledley, Robert S., Suzek, Baris E., Vinayaka, C.R., Zhang, Jian, Barker, Winona C., 2003. The protein information resource. Nucleic Acids Res. 31 (1), 345–347. https://doi.org/10.1093/nar/gkg040.

You, Ronghui, Zhang, Zihan, Xiong, Yi, Sun, Fengzhu, Mamitsuka, Hiroshi, Zhu, Shanfeng, 2018a. GOLabeler: improving sequence-based large-scale protein function prediction by learning to rank. Bioinformatics 34 (14), 2465–2473. https://doi.org/10.1093/bioinformatics/bty130.

You, Ronghui, Huang, Xiaodi, Zhu, Shanfeng, 2018b. "DeepText2Go: improving large-scale protein function prediction with deep semantic text representation". Methods 145, 82–90.

Yu, G., Rangwala, H., Domeniconi, C., Zhang, G., Zhang, Z., 2015. "Predicting protein function using multiple kernels". IEEE/ACM Trans. Comput. Biol. Bioinform. 12 (1), 219–233. https://doi.org/10.1109/TCBB.2014.2351821.

Yunes, Jeffrey M., Babbitt, Patricia C., 2019. Effusion: prediction of protein function from sequence similarity networks. Bioinformatics 35 (3), 442–451. https://doi.org/10.1093/bioinformatics/bty672.

Zhou, N., Jiang, Y., Bergquist, T.R., Lee, A.J., Kacsoh, B.Z., Crocker, A.W., Lewis, K.A., Georghiou, G., Nguyen, H.N., Hamid, M.N., Davis, L., Dogan, T., Atalay, V., Rifaioglu, A.S., Dalkıran, A., Cetin Atalay, R., Zhang, C., Hurto, R.L., Freddolino, P. L., Zhang, Y., Bhat, P., Supek, F., Fernández, J.M., Gemovic, B., Perovic, V.R., Davidović, R.S., Sumonja, N., Veljkovic, N., Asgari, E., Mofrad, M., Profiti, G., Savojardo, C., Martelli, P.L., Casadio, R., Boecker, F., Schoof, H., Kahanda, I., Thurlby, N., McHardy, A.C., Renaux, A., Saidi, R., Gough, J., Freitas, A.A., Antczak, M., Fabris, F., Wass, M.N., Hou, J., Cheng, J., Wang, Z., Romero, A.E., Paccanaro, A., Yang, H., Goldberg, T., Zhao, C., Holm, L., Törönen, P., Medlar, A.J., Zosa, E., Borukhov, I., Novikov, I., Wilkins, A., Lichtarge, O., Chi, P.H., Tseng, W.C., Linial, M., Rose, P.W., Dessimoz, C., Vidulin, V., Dzeroski, S., Sillitoe, I., Das, S., Lees, J.G., Jones, D.T., Wan, C., Cozzetto, D., Fa, R., Torres, M., Warwick Vesztrocy, A., Rodriguez, J.M., Tress, M.L., Frasca, M., Notaro, M., Grossi, G., Petrini, A., Re, M., Valentini, G., Mesiti, M., Roche, D.B., Reeb, J., Ritchie, D.W., Aridhi, S., Alborzi, S.Z., Devignes, M.D., Koo, D., Bonneau, R., Gligorijević, V., Barot, M., Fang, H., Toppo, S., Lavezzo, E., Falda, M., Berselli, M., Tosatto, S., Carraro, M., Piovesan, D., Ur Rehman, H., Mao, Q., Zhang, S., Vucetic, S., Black, G.S., Jo, D., Suh, E., Dayton, J.B., Larsen, D.J., Omdahl, A.R., McGuffin, L.J., Brackenridge, D.A., Babbitt, P.C., Yunes, J.M., Fontana, P., Zhang, F., Zhu, S., You, R., Zhang, Z., Dai, S., Yao, S., Tian, W., Cao, R., Chandler, C., Amezola, M., Johnson, D., Chang, J.M., Liao, W.H., Liu, Y.W., Pascarelli, S., Frank, Y., Hoehndorf, R., Kulmanov, M., Boudellioua, I., Politano, G., Di Carlo, S., Benso, A., Hakala, K., Ginter, F., Mehryary, F., Kaewphan, S., Björne, J., Moen, H., Tolvanen, M., Salakoski, T., 2019. The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. Genome Biol. 20, 244. https://doi.org/10.1186/s13059-019-1835-8.