UDC 519.68.02
DOI: 10.15587/1729-4061.2022.263270

IMPLEMENTATION OF MACHINE LEARNING MODELS TO DETERMINE THE APPROPRIATE MODEL FOR PROTEIN FUNCTION PREDICTION

Yekaterina Golenko
Corresponding author
Master of Science in Engineering*
E-mail: golenko.katerina@gmail.com
Aisulu Ismailova
PhD*
Anargul Shaushenova
Candidate of Technical Sciences*
Zhazira Mutalova
Master of Technical Sciences
Higher School of Information Technologies
Zhangir khan West Kazakhstan Agrarian Technical University
Zhangir khan str., 51, Uralsk, Republic of Kazakhstan, 090009
Damir Dossalyanov
PhD
Department of Public and Local Management
Narxoz University
Jandosova str., 55, Almaty, Republic of Kazakhstan, 050000
Aliya Ainagulova
Candidate of Technical Sciences*
Akgul Naizagarayeva
Master of Science in Engineering*
*Department of Information Systems
S. Seifullin Kazakh Agrotechnical University
Zhenis ave., 62, Astana, Republic of Kazakhstan, 010000

Predicting the function of proteins is a crucial part of genome annotation, which can help in solving a wide range of biological problems. Many methods are available to predict the functions of proteins. However, except for sequence, most features are difficult to obtain or are not available for many proteins, which limits their scope. In addition, the performance of sequence-based feature prediction methods is often lower than that of methods that involve multiple features, and protein feature prediction can be time-consuming. Recent advances in this field are associated with the development of machine learning, which shows great progress in solving the problem of predicting protein functions. Today, however, most protein sequences have the status of «uncharacterized» or «putative».

The need to assess the accuracy of identification of protein functions is an urgent task for machine learning approaches used to predict protein functions. In this study, the performance of two popular function prediction algorithms (ProtCNN and BiLSTM) was assessed from two perspectives and the procedures for building these models were described.

As a result of the study of Pfam families, ProtCNN achieves an accuracy rate of 0.988 % and bidirectional LSTM has an accuracy rate of 0.9506 %. The use of the Pfam dataset allowed increasing the classification accuracy due to the large training dataset. The quality of the prediction increases with a large amount of training data.

The study demonstrated that machine learning algorithms can be used as an effective tool for building protein function prediction models, in particular, the CNN network can be adapted as an accurate tool for annotating protein functions in the presence of large datasets

Keywords: protein function prediction, classification, neural networks, ProtCNN, bidirectional long short-term memory (BiLSTM)

Received date 02.08.2022
Accepted date 07.10.2022
Published date 30.10.2022

How to Cite: *Golenko, Y., Ismailova, A., Shaushenova, A., Mutalova, Z., Dossalyanov, D., Ainagulova, A., Naizagarayeva, A. (2022). Implementation of machine learning models to determine the appropriate model for protein function prediction. Eastern-European Journal of Enterprise Technologies, 5 (2 (119)), 42–49. doi: https://doi.org/10.15587/1729-4061.2022.263270*

1. Introduction

Proteins are long chains of amino acids that are formed from information received from DNA and then folded into three-dimensional shapes. The forms that protein molecules take are determined by the information embedded in DNA, and what form the DNA molecule itself folds into depends on the composition of the amino acids in the chain. In turn, the form in biology determines the function.

While techniques such as microarray analysis, RNA interference, and the yeast two-hybrid system can be used to experimentally demonstrate the function of a protein, advances in sequencing technologies have made the rate at which proteins can be experimentally characterized much slower than the rate at which new sequences become available [1]. Thus, annotation of new sequences is mostly done by computational prediction, as these types of annotations can often be done quickly and on many genes or proteins at the same time. The first such methods assumed function based on homologous proteins with known functions (homology-based function prediction). The development of context-based and structure-based methods has expanded the amount of information that can be predicted, and a combination of methods can now be used to derive a picture of complete cellular pathways

from sequence data [1]. The importance and prevalence of computer-aided prediction of gene function are highlighted by the analysis of the «evidence codes» used by the Gene Ontology (GO) database: 98 % of annotations were listed under an IEA code (derived from an electronic abstract), while only <0.1 % of the more than 179 million proteins in UniProt were based on experimental data [2].

A rapidly growing area of research is the application of machine learning to protein classification. Neural networks are the most widely used machine learning tools applied to solve classification problems. Protein function prediction also can be viewed as a classification problem.

Protein function prediction is a multi-label classification problem where we have a set of functions $F=(F_1, ..., F_m)$. Given a protein set, $P=(P_1, ..., P_n)$, where the first l proteins are labeled as $y_1,...,y_l$, each y_i is a vector with $y_{ij}=1$ in case the protein P_i is associated with the j-th function F_j, otherwise $y_{ij}=0$. The goal is to predict the labels $y_{1+1},...,y_n$ for the remaining unlabelled proteins $P_{1+1}, ..., P_n$.

Various approaches can be taken to solve classification problems with multiple labels, but the simplest method is to treat each GO term as an independent classification problem. The most suitable option for solving this problem is machine learning models with a better ratio of classification quality and calculation time. The huge advantage of machine learning is that model's performance and accuracy have a stable growth with a large amount of training data. Considering the rapid increase in the number of protein sequences in open databases, this circumstance makes the use of machine learning to solve the problem of protein function prediction a relevant problem for research.

A huge number of neural networks have been developed to solve object classification problems. This opens great opportunities for their use for the problem of protein function prediction, which is considered as a classification problem. However, based on statistics demonstrating that the majority of protein sequences in databases have not yet received the status of characterized, it can be argued that the capabilities of neural networks are not well studied in terms of the functional annotation of proteins. This leads to the expediency of studying these neural networks and testing them multiple times on different datasets, which will help determine their strengths and weaknesses for solving the above problem. Therefore, studies that are devoted to investigating machine learning models and assessing their accuracy to solve the problem of predicting protein functions are of scientific relevance.

2. Literature review and problem statement

Understanding protein function is important for the study of biological mechanisms of disease development and drug discovery. Many databases are updated daily to provide functional annotations from different perspectives such as: protein-protein interaction, biological network and many specific functional classes. However, the total number of detected protein sequences significantly exceeds the number of proteins characterized with a known function. In order to narrow the gap between the number of characterized and uncharacterized protein sequences, thousands of high-throughput genomic projects are being explored, but only 1 % of the sequences found have been confirmed by experimental annotation [3]. This created a great need for the development of theoretical methods for annotating protein functions. A wide range of methods have been developed and extensively used to discover protein functions. These include clustering of sequences, gene fusion, sequence similarity, evolution study, structural comparison, protein-protein interaction, functional classification via the sequence-derived and domain feature, omics profiling and integrated methods, which collectively consider multiple methods and data to promote the performance of function prediction [4]. Except for methods using sequence similarities, the disadvantage of these methods is that they use the features of proteins that are difficult to obtain or are not available at all for many proteins, which greatly limits the scope of such methods.

Among the numerous methods, BLAST [5] has gained the most popularity, demonstrating great potential for revealing protein functions. This is a program widely used in bioinformatics to search for homologs for a given protein sequence in various databases. The program receives a sequence in the FastA format as input, selects the database to be searched for, and the search algorithm. BLAST prediction is based on protein sequence similarity, while machine learning prediction is also based on sequences but without taking into account their similarity. This unique characteristic of machine learning makes it a good complement to BLAST and many other approaches to predict the functions of distantly relevant proteins and homologous proteins with different functions. However, BLAST has disadvantages such as slow execution and a relatively low level of data sensitivity.

In recent years, machine learning algorithms have been gaining popularity, and various online software tools based on machine learning have been developed as predictors without considering similarities in sequence or structure, including methods incorporating various lines of evidence as features for training classifiers and predicting GO terms.

Support Vector Machine (SVM) classifiers have been widely used in a large number of studies in the field of bioinformatics including areas of study of the structure and functions of proteins [6]. The main common disadvantage of SVM-based applications is the long training time for large datasets, moreover, it is difficult to understand and interpret the final model. For instance, FFPred [7] has been trained and tested on human annotated proteins in its first implementation, however, further showed generalization to other organisms (zebrafish, mouse, fly, worm and yeast). The new version of this tool, FFPred3, which is still based on SVM, has been extended to explore correlations between trait features extracted from sequences and structures within a vocabulary of 400 biological process terms (BP), 108 in the molecular function (MF) domain, and 89 in the cellular component (CC) domain [8]. The main disadvantage of [8] is that the performance of this tool has not been investigated when working with large datasets. Another example, a classifier called PoGO (Prediction of Gene Ontology terms) uses not only terms as characteristics, but also combines three sources (sequence similarity, biochemical properties, and protein tertiary structure) [9]. As in the previous study, there was no analysis of the effectiveness of the algorithm on a large dataset.

The k-nearest neighbors (k-NN) algorithm [10–12] has also found wide application in functional annotation systems. The disadvantage is that this algorithm shows low efficiency when working with large datasets, as well as with high-dimensional data. PANNZER2 [10] is a good annotation tool, which demonstrates high processing speed, but does not surpass similar classification tools in accuracy.

DeepText2GO [11] integrates the text-based method with the sequence-based method, which improves its accuracy but requires more training data. The work [12] does not demonstrate a new idea in this field, however, NetGO shows high results and outperforms most of the tools using similar algorithms. The main disadvantage of NetGO is high computational complexity.

Convolutional Neural Network (CNN) was originally developed to process 2D images, however, today it is successfully used for processing genomic sequences. A CNN consists of different types of layers: convolutional layers, subsampling layers and layers of a «normal» neural network – the perceptron. It should be noted that the models described in [13–17] were compared only with models built based on similar neural networks, or with classical tools like BLAST. Comparative analysis of different architectures is still relevant for machine learning models. In particular, in [13], poor performance of DeepGO demonstrates the weakness of deep learning based methods that only work with a small number of GO terms. SDN2GO [14], DeepAdd [15], DeepGOA [16] and TailGnn's [17] also show good but not outstanding results in annotating protein functions. Comparative analysis of these tools does not allow highlighting one tool as superior to others. They have a common disadvantage, which is computational complexity, moreover, to predict functions more accurately each of these tools requires a combination of different protein inputs, which is often not possible.

Recurrent neural networks (RNNs) are a type of neural networks specialized in sequence processing. Their feature is the transmission of signals from the output or hidden layer of the neural network to the input layer [18]. Like any other neural network, a recurrent neural network can consist of any number of layers. A neural network with a long short-term memory (LSTM network) is one of the varieties of recurrent neural networks [19]. The key component of an LSTM network is the so-called cell state. The state of the cell is involved in several linear transformations. The state of the cell is responsible for the learning process, error backpropagation, and updating the weights. Although LSTM networks work well for some problems, there is a list of disadvantages: LSTMs take a lot of time and memory to train, they are easy to overfit and sensitive to different random weight initializations.

The success of machine learning methods is also due to the fact that the amount of accumulated data allows training the developed models with sufficient accuracy.

However, most protein sequences are still labeled as «putative», «uncharacterized», «unknown function», or «hypothetical». Moreover, the identification accuracy of these approaches still needs further optimization and study [20, 21]. Thus, the need to assess the accuracy of protein function identification is still an urgent task for machine learning approaches used to predict protein functions.

3. The aim and objectives of the study

The aim of this work is to identify the most appropriate machine learning model and evaluate the performance (in terms of accuracy) of two popular machine learning algorithms (bidirectional LSTM and CNN) commonly used for protein function prediction. This will make it possible to apply and improve a suitable machine learning model with high accuracy on large amounts of data, which will allow increasing the number of characterized proteins.

To achieve this aim, the following objectives are accomplished:

– to implement two machine learning models and train them using a random partition of the public Pfam dataset, which consists of three sets of data: *train* – 80 % (model training), *val* – 10 % (model validation) and *test* – 10 % (model performance evaluation);

– to evaluate the performance of the models using two metrics: accuracy and multi-class loss log.

4. Materials and methods

4. 1. Dataset description

To compare different models with protein domain misalignment annotation, the public database Pfam was used as the main data source. Pfam is a database of protein domain families. Each family in it is represented by a multiple alignment of protein sequence fragments and a hidden Markov model (HMM). 77.2 % of ~137M sequences in UniprotKB have at least one Pfam family annotation [22]. As of November 2021, Pfam contained 19 632 entries (families) united into 657 clans [23].

This database contains five features:

– sequence: The amino acid sequence for the given domain. This sequence represents a domain, not a full protein;

– family_accession: Accession number;

– sequence_name: The name of the sequence;

– aligned_sequence: Contains one sequence from a multiple sequence alignment;

– family_id: One-word family name.

Table 1 shows some examples of data used.

Table 1

Examples of data

No.	Field name	Data
1	sequence:	VLERKISTRQTREELIKKGVLIPD
	family_accession	PF02755.15
	sequence_name	I3IWL9_ORENI/33-56
	aligned_sequence	VLERKISTRQTREELIKKGVLIPD
	family_id	RPEL
2	sequence	NPCTIDSCGPKGCVHIAMSCDDN
	family_accession	PF00526.18
	sequence_name	F0ZFD3_DICPU/581-603
	aligned_sequence	NPCTIDSC.GPK....G.CVHIAM.SCDDN
	family_id	Dicty_CTDC
3	sequence:	KLNSLGGLVALNLGSIDNASASGTLV
	family_accession	PF07581.12
	sequence_name:	Q7WYX3_PSEAI/240-265
	aligned_sequence	KLNSLGGLVALNL.....GSIDNASASG.TLV
	family_id	Glug

We used the same dataset as [24]. The benchmark includes a random split of 17,929 Pfam families into a test train, where 80 % of the sequences are used for training, 10 % for model tuning, and 10 % as test sequences. Fig. 1 illustrates the total number of sequences in the dataset.

Thus, to build and train the model, three datasets were used: *train* dataset, *val* dataset, and *test* dataset, consisting of 1,086,741 sequences, 126,171 sequences, and 126,171 sequences, respectively.

```
[ ]  # Given data size
     print('Train size: ', len(df_train))
     print('Val size: ', len(df_val))
     print('Test size: ', len(df_test))

     Train size:  1086741
     Val size:  126171
     Test size:  126171
```

Fig. 1. The total number of sequences in each dataset

4. 2. Protein sequence coding

To code the protein sequences, we represented letter codes of amino acid sequences by the corresponding integer values in ascending order, which are further used for integer coding. We used the one-hot encoding method to code the protein sequences. This type of encoding creates a new binary feature for each possible category and assigns a value of 1 to the feature of each sample that corresponds to its original category.

4. 3. Machine learning models

A common outline of machine learning solutions, the details of which may vary depending on the method, is shown in Fig. 2 [25].

ProtCNN uses residual blocks inspired by the ResNet architecture, which also includes extended convolutions to provide a larger receptive field without increasing the number of model parameters. As an input signal, the one-hot encoded unaligned amino acid sequence with zero padding is passed to the network. In the general case, the formation of the output feature map of the hidden layer l of the CNN architecture can be described as follows:

$$h_j^l = f\left(\sum_i x_i^{l-1} * k_j^l + b_j^l\right), \qquad (1)$$

where f is the activation function; b_j is the shift coefficient for the feature map; k_j is the convolution kernel number j; x_i^{l-1} – map of features of the previous layer; * – convolution operation. Fig. 3 shows the architecture of the CNN model.

BiLSTM is one of the types of recurrent neural networks, which processes sequence data in both forward and backward directions with two separate hidden layers. BiLSTM is based on input, forget and output gates. The following formulae (2) are used to calculate the predicted values [26]:

$$\begin{aligned}
&\text{input gate}\left(i_t\right) = \sigma_g\left(W_i X_t + R_i h_{t-1} + b_i\right),\\
&\text{forget gate}\left(f_t\right) = \sigma_g\left(W_f X_t + R_f h_{t-1} + b_f\right),\\
&\text{cell candidate}\left(c_t\right) = \sigma_g\left(W_c X_t + R_c h_{t-1} + b_c\right),\\
&\text{output gate}\left(o_t\right) = \sigma_g\left(W_o X_t + R_o h_{t-1} + b_o\right),
\end{aligned} \qquad (2)$$

where σ_g is the gate activation function and W_i, W_f, W_c, and W_o are input weight matrices, while R_i, R_f, R_c, and R_o are the weight matrices connecting the previous cell output state to the three gates and the input cell state. X_t is the input, and h_{t-1} is the output at the previous time ($t-1$). b_i, b_f, b_o and b_c are bias vectors.

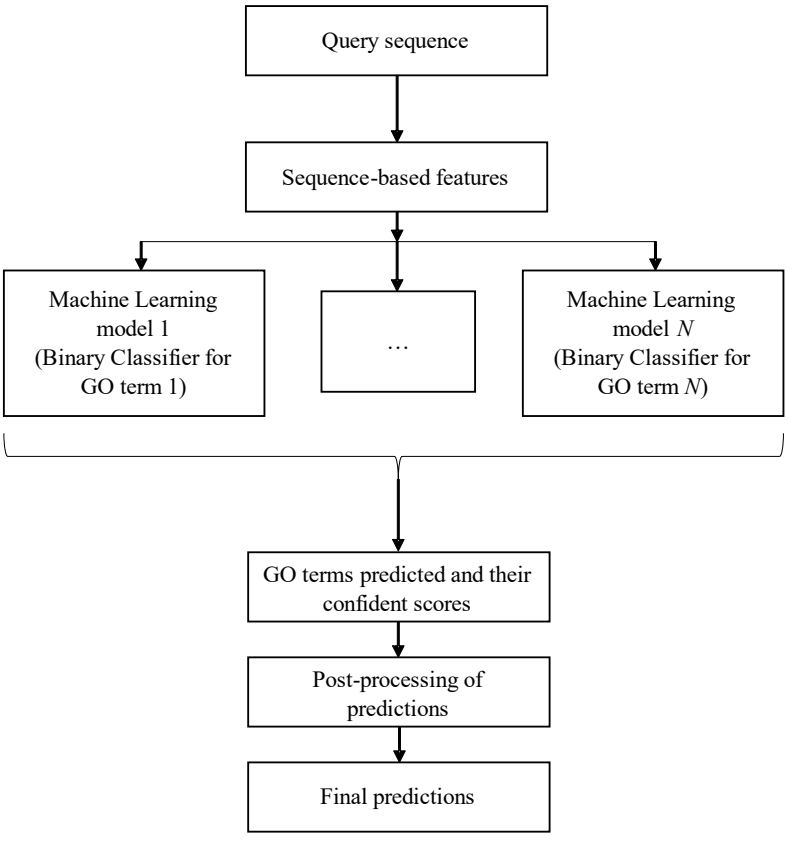


Fig. 2. General algorithm for machine learning solutions to predict GO terms of proteins

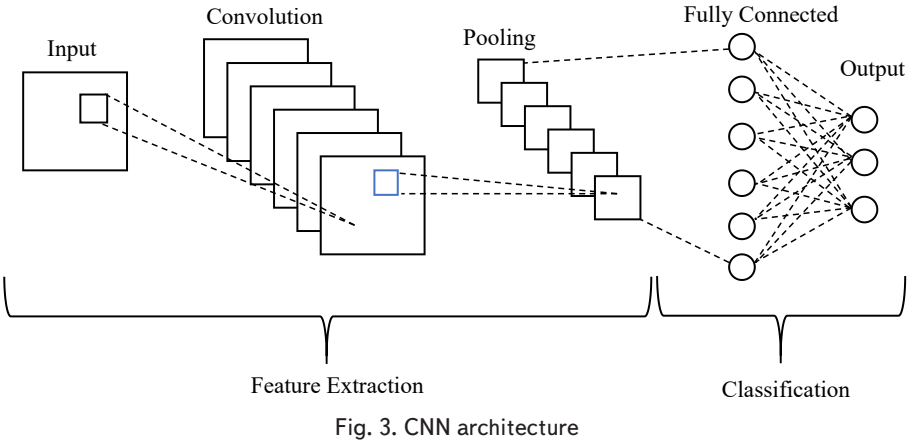


Fig. 3. CNN architecture

At each time iteration t, the cell output state, C_t, and the layer output, h_t, can be calculated as follows [27]:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t, \quad (3)$$

$$h_t = o_t * \tanh(C_t). \quad (4)$$

The architecture of the bidirectional LSTM model is presented in Fig. 4.

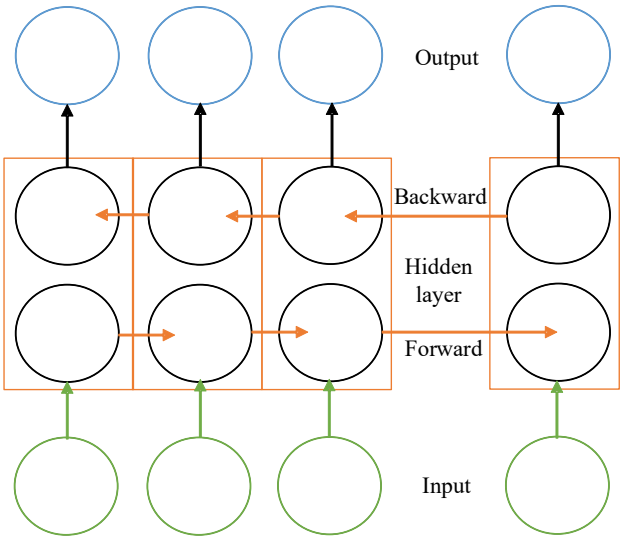


Fig. 4. BiLSTM architecture

Using bidirectional LSTM runs inputs in two ways, allowing to preserve contextual information from past and future at any point of time.

4. 4. Assessing the identification accuracies of the studied models

To evaluate the performance of the different models for predicting protein properties, we used metrics of the classification accuracy, defined respectively by (5):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \quad (5)$$

where TP, TN, FP, and FN represent true positive, true negative, false positive, and false negative.

In testing, accuracy reflects the classification accuracy of the learned classifier on the testing dataset, which has no overlap with the training dataset. It is generally accepted that the test accuracy serves as a good indicator of prediction performance [28].

The models were implemented in Python using the Num.py, TensorFlow and Keras libraries and run on the Colab cloud platform.

5. Results of studying machine learning models for protein function prediction

5. 1. Implementation of BiLSTM and ProtCNN neural networks

During the implementation of the models, the following were used:

– Adam algorithm as an optimization algorithm [29];

– Accuracy indicator as an objective function;

– Binary-crossentropy function returning the classification error as a logistic loss function *Loss*:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} y_i * \log(\hat{y}_i) + (1 - \hat{y}_i) * \log(1 - \hat{y}_i), \quad (6)$$

where y_i is the true class label; $\hat{y}_i$ is the classifier's response (calculated class label) to the i-th object; N is the number of classes. Fig. 5, 6 show a part of the code of BiLSTM and ProtCNN, respectively.

The model architecture consists of an embedding layer for learning the vector representation for each code followed by BiLSTM. Dropout is added for regularization to prevent model overfitting. The output layer gives probability values for all the unique classes, and based on the biggest predicted probability, the model will classify amino acid sequences to one of its protein family accession.

The ProtCNN architecture starts with an initial convolution operation applied to the input data with a kernel size of 1 to extract the main properties. Two residual blocks are used to capture complex patterns in the data, which help to train with more epochs and better model performance. After two residual blocks, max pooling is applied to reduce the spatial size of the representation. Dropout is added for regularization to prevent model overfitting.

5. 2. Evaluation of the performance of the models using two metrics: accuracy and multi-class loss log

Table 2 shows the results of testing the BiLSTM network obtained during the first 10 training epochs.

Fig. 7 shows graphical representations of the training results of the LSTM network model for 25 epochs in terms of Accuracy and Loss, respectively.

```
x_input = Input(shape=(100,))
emb = Embedding(21, 128, input_length=max_length)(x_input)
bi_rnn = Bidirectional(CuDNNLSTM(64, kernel_regularizer=l2(0.01),
                                 recurrent_regularizer=l2(0.01), bias_regularizer=l2(0.01)))(emb)
x = Dropout(0.3)(bi_rnn)

# softmax classifier
x_output = Dense(1000, activation='softmax')(x)

model1 = Model(inputs=x_input, outputs=x_output)
model1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

model1.summary()
```

Fig. 5. A fragment of the program code for the BiLSTM network model implementation

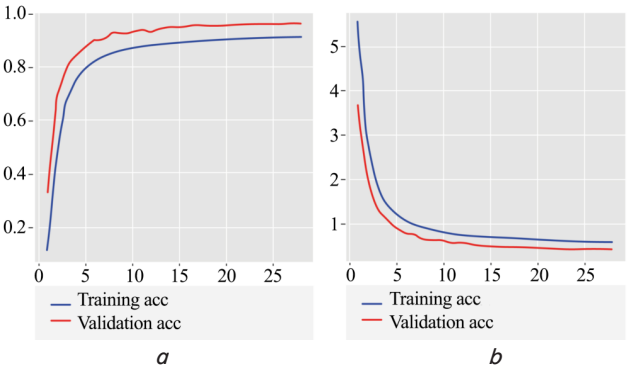
```
# model

x_input = Input(shape=(100, 21))

#initial conv
conv = Conv1D(128, 1, padding='same')(x_input)

# per-residue representation
res1 = residual_block(conv, 128, 2)
res2 = residual_block(res1, 128, 3)

x = MaxPooling1D(3)(res2)
x = Dropout(0.5)(x)

# softmax classifier
x = Flatten()(x)
x_output = Dense(1000, activation='softmax', kernel_regularizer=l2(0.0001))(x)

model2 = Model(inputs=x_input, outputs=x_output)
model2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

model2.summary()
```

Fig. 6. A fragment of the program code for ProtCNN implementation

Table 2

Learning results of the BiLSTM network model

Epoch	Loss	Accuracy	Val_loss	Val_accuracy
1	5.5302	0.1114	3.6586	0.3238
2	2.8557	0.4747	2.0003	0.6737
3	1.8840	0.6672	1.3561	0.7960
4	1.4464	0.7497	1.0806	0.8376
5	1.2164	0.7915	0.9047	0.8712
6	1.0732	0.8180	0.7718	0.8977
7	0.9671	0.8377	0.7293	0.9020
8	0.8970	0.8503	0.6265	0.9233
9	0.8420	0.8606	0.6085	0.9219
10	0.8020	0.8674	0.5973	0.9246



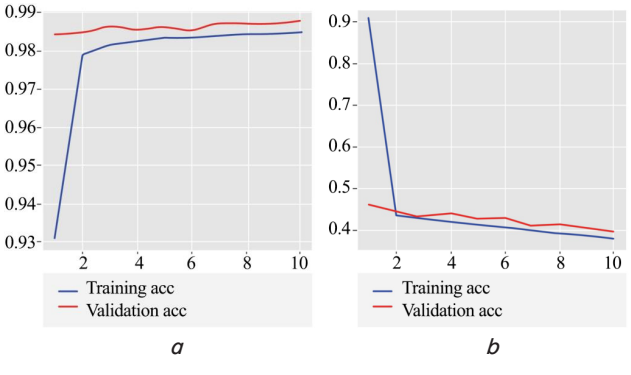
a *b*

Fig. 7. The results of the BiLSTM network model training according to the Accuracy indicator: *a* – accuracy value, *b* – loss function value

Table 3

Learning results of the ProtCNN model

Epoch	Loss	Accuracy	Val_loss	Val_accuracy
1	0.9027	0.9306	0.4603	0.9843
2	0.4362	0.9790	0.4435	0.9849
3	0.4296	0.9814	0.4326	0.9864
4	0.4216	0.9823	0.4394	0.9855
5	0.4131	0.9832	0.4263	0.9861
6	0.4051	0.9835	0.4281	0.9852
7	0.3967	0.9840	0.4088	0.9872
8	0.3907	0.9844	0.4124	0.9870
9	0.3865	0.9845	0.4034	0.9871
10	0.3787	0.9849	0.3954	0.9879



a *b*

Fig. 8. The results of ProtCNN model training according to the Accuracy indicator: *a* – accuracy value, *b* – loss function value

Table 3 illustrates the results of testing the ProtCNN network obtained during the first 10 training epochs.

Fig. 8 shows graphical representations of the training results of the ProtCNN model for 10 epochs in terms of Accuracy and Loss, respectively.

The statistical differences in the accuracy of the ProtCNN and BiLSTM models are illustrated in Table 4.

The comparison of the accuracy of ProtCNN and BiLSTM shows that a convolutional neural network predicts more accurately than a bidirectional long short-term memory network.

Table 4

Statistical differences in the accuracy of ProtCNN and BiLSTM

No.	Model	Train Accuracy	Val Accuracy	Test Accuracy
1	Bidirectional LSTM	0.9571	0.9511	0.9506
2	ProtCNN	0.9967	0.9877	0.9880

6. Discussion of the results of the comparative analysis of two machine learning models

In this paper, we compare the performance of ProtCNN and the bidirectional LSTM network in the Pfam domain annotation task. Table 2 and Table 3 illustrate training results at each epoch and show the increase of accuracy after each step, which allows us to talk about good predictive results of both models. For the machine learning methods and model architectures used herein, the ProtCNN model has been found to generally outperform bidirectional LSTM across different protein types. These results represent a significant advance over previous deep learning efforts in terms of the number of families and the number of training sequences per family. Table 4 shows that on randomly split data, ProtCNN achieves an accuracy rate of 0.988 % and bidirectional LSTM has an accuracy rate of only 0.9506 %. The performance difference is significant over a wide range of similarities between the test sequences and the training set.

The obtained results are determined by two factors. The first one is the volume of datasets on which the models were trained. Given ~1.1 million training examples across 17,929 output families of a wide variety of sizes, the studied models are highly accurate. Second, the use of extended convolutions in ProtCNN helped the model learn more complex features better than BiLSTM, as extensions allow for larger receptive fields. Skipped connections helped the model retain important spatial information from previous layers.

When choosing between ProtCNN and bidirectional LSTM, there is a clear trade-off between speed and accuracy. However, the ability of the approaches to improve the accuracy of a single model without additional computational overhead suggests that this trade-off is not necessary. Applying more sophisticated machine learning techniques can lead to further performance improvements.

As a limitation, this work didn't cover other popular metrics such as sensitivity (*SE*), specificity (*SP*) and Matthews correlation coefficient (*MCC*).

In future work, we will experiment with more different neural networks using four metrics for assessing the identification accuracies, as well as using the new version of the Pfam database.

7. Conclusions

1. In this study, two machine learning algorithms (BiLSTM and ProtCNN) were implemented and trained. The following instruments were applied during model implementation: *Adam* algorithm as an optimization algorithm, *Accuracy* indicator as an objective function and *Binary-crossentropy* function returning the classification error as a logistic loss function *Loss*. The advantage of the selected tools is high efficiency with sufficient ease of implementation, although the learning process can take quite a long time. The prediction models were trained using the training set and evaluated on the test set. Using the full Pfam dataset allowed both models to be trained to a sufficiently high level of accuracy without leading to overtraining.

2. The performance evaluation of the implemented models demonstrated that the ProtCNN model was found to outperform the bidirectional LSTM model for different types of proteins. The best accuracy of ProtCNN is 0.9967 for the *train* set, 0.9877 for the *val* set and 0.9880 for the *test* set, while BiLSTM showed only 0.9571 for the *train* set, 0.9511 for the *val* set and 0.9506 for the *test* set.

Conflict of interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship or otherwise, that could affect the research and its results presented in this paper.

References

1. Gabaldon, T., Huynen, M. A. (2004). Prediction of protein function and pathways in the genome era. Cellular and Molecular Life Sciences (CMLS), 61 (7-8), 930–944. doi: https://doi.org/10.1007/s00018-003-3387-y
2. du Plessis, L., Skunca, N., Dessimoz, C. (2011). The what, where, how and why of gene ontology--a primer for bioinformaticians. Briefings in Bioinformatics, 12 (6), 723–735. doi: https://doi.org/10.1093/bib/bbr002
3. Barrell, D., Dimmer, E., Huntley, R. P., Binns, D., O'Donovan, C., Apweiler, R. (2009). The GOA database in 2009--an integrated Gene Ontology Annotation resource. Nucleic Acids Research, 37, D396–D403. doi: https://doi.org/10.1093/nar/gkn803
4. Piovesan, D., Giollo, M., Leonardi, E., Ferrari, C., Tosatto, S. C. E. (2015). INGA: protein function prediction combining interaction networks, domain assignments and sequence similarity. Nucleic Acids Research, 43 (W1), W134–W140. doi: https://doi.org/10.1093/nar/gkv523
5. Boratyn, G. M., Camacho, C., Cooper, P. S., Coulouris, G., Fong, A., Ma, N. et. al. (2013). BLAST: a more efficient report with usability improvements. Nucleic Acids Research, 41 (W1), W29–W33. doi: https://doi.org/10.1093/nar/gkt282
6. Stephenson, N., Shane, E., Chase, J., Rowland, J., Ries, D., Justice, N. et. al. (2019). Survey of Machine Learning Techniques in Drug Discovery. Current Drug Metabolism, 20 (3), 185–193. doi: https://doi.org/10.2174/1389200219666180820112457
7. Lobley, A. E., Nugent, T., Orengo, C. A., Jones, D. T. (2008). FFPred: an integrated feature-based function prediction server for vertebrate proteomes. Nucleic Acids Research, 36, W297–W302. doi: https://doi.org/10.1093/nar/gkn193
8. Cozzetto, D., Minneci, F., Currant, H., Jones, D. T. (2016). FFPred 3: feature-based function prediction for all Gene Ontology domains. Scientific Reports, 6 (1). doi: https://doi.org/10.1038/srep31865

9. Jung, J., Yi, G., Sukno, S. A., Thon, M. R. (2010). PoGO: Prediction of Gene Ontology terms for fungal proteins. BMC Bioinformatics, 11 (1). doi: https://doi.org/10.1186/1471-2105-11-215
10. Törönen, P., Medlar, A., Holm, L. (2018). PANNZER2: a rapid functional annotation web server. Nucleic Acids Research, 46 (W1), W84–W88. doi: https://doi.org/10.1093/nar/gky350
11. You, R., Huang, X., Zhu, S. (2018). DeepText2GO: Improving large-scale protein function prediction with deep semantic text representation. Methods, 145, 82–90. doi: https://doi.org/10.1016/j.ymeth.2018.05.026
12. You, R., Yao, S., Xiong, Y., Huang, X., Sun, F., Mamitsuka, H., Zhu, S. (2019). NetGO: improving large-scale protein function prediction with massive network information. Nucleic Acids Research, 47 (W1), W379–W387. doi: https://doi.org/10.1093/nar/gkz388
13. Kulmanov, M., Khan, M. A., Hoehndorf, R. (2017). DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. Bioinformatics, 34 (4), 660–668. doi: https://doi.org/10.1093/bioinformatics/btx624
14. Cai, Y., Wang, J., Deng, L. (2020). SDN2GO: An Integrated Deep Learning Model for Protein Function Prediction. Frontiers in Bioengineering and Biotechnology, 8. doi: https://doi.org/10.3389/fbioe.2020.00391
15. Du, Z., He, Y., Li, J., Uversky, V. N. (2020). DeepAdd: Protein function prediction from k-mer embedding and additional features. Computational Biology and Chemistry, 89, 107379. doi: https://doi.org/10.1016/j.compbiolchem.2020.107379
16. Zhang, F., Song, H., Zeng, M., Wu, F.-X., Li, Y., Pan, Y., Li, M. (2021). A Deep Learning Framework for Gene Ontology Annotations With Sequence- and Network-Based Information. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 18 (6), 2208–2217. doi: https://doi.org/10.1109/tcbb.2020.2968882
17. Spalević, S., Veličković, P., Kovačević, J., Nikolić, M. (2020). Hierarchical Protein Function Prediction with Tail-GNNs. arXiv. doi: https://doi.org/10.48550/arXiv.2007.12804
18. LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. Nature, 521 (7553), 436–444. doi: https://doi.org/10.1038/nature14539
19. Cao, R., Freitas, C., Chan, L., Sun, M., Jiang, H., Chen, Z. (2017). ProLanGO: Protein Function Prediction Using Neural Machine Translation Based on a Recurrent Neural Network. Molecules, 22 (10), 1732. doi: https://doi.org/10.3390/molecules22101732
20. Jiang, Y., Oron, T. R., Clark, W. T., Bankapur, A. R., D'Andrea, D., Lepore, R. et. al. (2016). An expanded evaluation of protein function prediction methods shows an improvement in accuracy. Genome Biology, 17 (1). doi: https://doi.org/10.1186/s13059-016-1037-6
21. Pearson, W. R. (2015). Protein Function Prediction: Problems and Pitfalls. Current Protocols in Bioinformatics, 51 (1). doi: https://doi.org/10.1002/0471250953.bi0412s51
22. UniProt: the universal protein knowledgebase (2016). Nucleic Acids Research, 45 (D1), D158–D169. doi: https://doi.org/10.1093/nar/gkw1099
23. Pfam 35.0 is released. Xfam Blog. Available at: https://xfam.wordpress.com/2021/11/19/pfam-35-0-is-released/
24. Bileschi, M. L., Belanger, D., Bryant, D., Sanderson, T., Carter, B., Sculley, D. et. al. (2019). Using Deep Learning to Annotate the Protein Universe. bioRxiv. doi: https://doi.org/10.1101/626507
25. Vu, T. T. D., Jung, J. (2021). Protein function prediction with gene ontology: from traditional to deep learning models. PeerJ, 9, e12019. doi: https://doi.org/10.7717/peerj.12019
26. Abduljabbar, R. L., Dia, H., Tsai, P.-W. (2021). Unidirectional and Bidirectional LSTM Models for Short-Term Traffic Prediction. Journal of Advanced Transportation, 2021, 1–16. doi: https://doi.org/10.1155/2021/5589075
27. Kurtukova, A. V., Romanov, A. S. (2019). Modeling the neural network architecture to identify the author of the source code. Proceedings of Tomsk State University of Control Systems and Radioelectronics, 22 (3), 37–42. doi: https://doi.org/10.21293/1818-0442-2019-22-3-37-42
28. Deen, A., Gayanchandani, M. (2019). Protein Function Prediction using SVM Kernel Approach. International Journal of Scientific & Engineering Research, 10 (7), 1995–2000. Available at: https://www.ijser.org/researchpaper/Protein-Function-Prediction-using-SVM-Kernel-Approach.pdf
29. Kingma, D. P., Ba, J. (2014). Adam: A Method for Stochastic Optimization. 3rd International Conference for Learning Representations. San Diego. doi: https://doi.org/10.48550/arXiv.1412.6980