Received: 11 June 2023 | Revised: 19 July 2024 | Accepted: 1 August 2024

DOI: 10.1111/coin.12696

ORIGINAL ARTICLE

Computational Intelligence WILEY

ProtienCNN-BLSTM: An efficient deep neural network with amino acid embedding-based model of protein sequence classification and biological analysis

Umesh Kumar Lilhore[1] | **Sarita Simaiya**[1] | **Surjeet Dalal**[2] | **Neetu Faujdar**[3] | **Yogesh Kumar Sharma**[4] | **K. B. V. Brahma Rao**[4] | **V. V. R. Maheswara Rao**[5] | **Shilpi Tomar**[6] | **Ehab Ghith**[7] | **Mehdi Tlija**[8]

Correspondence
Surjeet Dalal, Department of Computer Science and Engineering, Amity University Haryana, Gurugram, Haryana, India.
Email: sdalal@ggn.amity.edu

Funding information
King Saud University, Riyadh, Saudi Arabia, Grant/Award Number: RSPD2024R685

Abstract

Protein sequence classification needs to be performed quickly and accurately to progress bioinformatics advancements and the production of pharmaceutical products. Extensive comparisons between large databases of known proteins and unknown sequences are necessary in traditional protein classification methods, which can be time-consuming. This labour-intensive and slow manual matching and classification method depends on functional and biological commonalities. Protein classification is one of the many fields in which deep learning has recently revolutionized. The data on proteins are organized hierarchically and sequentially, and the most advanced algorithms, such as Deep Family-based Method (DeepFam) and Protein Convolutional Neural Network (ProtCNN), have shown promising results in classifying proteins into relative groups. On the other hand, these methods frequently refuse to acknowledge this fact. We propose a novel hybrid model called

For affiliations refer to page 36.

ProteinCNN-BLSTM to overcome these particular challenges. To produce more accurate protein sequence classification, it combines the techniques of amino acid embedding with bidirectional long short-term memory (BLSTM) and convolutional neural networks (CNNs). The CNN component is the most effective at capturing local features, while the BLSTM component is the most capable of modeling long-term dependencies across protein sequences. Through the process of amino acid embedding, sequences of proteins are transformed into numeric vectors, which significantly improves the precision of prediction and the representation of features. Using the standard protein samples PDB-14189 and PDB-2272, we analyzed the proposed ProteinCNN-BLSTM model and the existing deep-learning models. Compared to the existing models, such as CNN, LSTM, GCNs, CNN-LSTM, RNNs, GCN-RNN, DeepFam, and ProtCNN, the proposed model performed more accurately and better than the existing models.

KEYWORDS

biological analysis, convolutional neural network, deep learning, LSTM, protein sequence classification

1 | INTRODUCTION

Predicting protein structure and function features based on amino acid sequences is a frequent task in computational biology. The majority of cutting-edge methods for achieving this goal have been combining evolutionary data with artificial intelligence for the past few years. The time required to retrieve related proteins from expanding sequence databases is becoming so great that it is difficult to analyze entire proteomes. Furthermore, the power of evolutionary information decreases for small families, such as proteins found among the Dark Proteins.[1]

The fundamental components of proteins are amino acids and organic molecules composed of protein structures; these proteins are also called polypeptides, which are compressed into cylindrical shapes. The formation of peptide bonds between protein acid carboxyl communities and neighboring protein residues is essential for the simultaneous interaction of all amino acids within an assembly of polymers. When determining the ordered arrangement of genes, the genetic code can specify the order in which amino acids are expected to appear in an amino acid sequence. For example, genes are portions of deoxyribonucleic acid

(DNA) that include instructions for producing a specific protein, ribonucleic acid (RNA), or chromosome.[2]

Within the field of bioinformatics, one of the most significant challenges is identifying and categorizing the composition and features of proteins. Traditional laboratory approaches are insufficient for managing massive quantities of statistical data about RNA. To accomplish this, protein molecules tend to be structured into a number of different families and subfamilies.[3] As a consequence, proteins need to organize themselves to form families, in addition to the functions of individuals.

Using a large number of computational computations for feature extraction, traditional machine learning techniques classify amino acid molecules. The selected features can affect the accuracy of the information extracted manually from the parameters. The application of many different artificial intelligence techniques, particularly Artificial Neural Network (ANNs),[4] is required for protein function analysis. Recently, deep learning has been successfully applied to a wide range of challenges involving categorizing massive datasets. Concerning computing enormous amounts of DNA sequence data, deep learning technology provides advantages that cannot be matched. k-mer Enhanced Gated Recurrent Unit (KEGRU) is a deep-learning algorithm introduced in a previous publication.[5] This model combines bidirectional Gated Recurrent Unit (GRU) networks with k-mer embedding to identify transcription factor binding sites. Evaluating the model's predictability developed in a Deep Learning-based technique and measuring prediction measurements such as recall, precision, F-measure, accuracy, and false detection percentages for the sequence of proteins enable investigators[6] to predict these substances from original protein sequences. This was accomplished by comparing the level of precision of the predictive model.

1.1 | Research motivation

Protein sequences play a crucial role in determining the composition and functioning of proteins, which is the motivation behind the study that constitutes the "ProtienCNN-BLSTM" model. Twenty different groups of amino acids make up proteins, and the particular sequence of the individual amino acid sequences determines the structures and functions of proteins. One of the primary goals of molecular biology research is to understand the connection between the sequence of amino acids of the protein and the function it serves.[6]

There have been numerous instances in which intricate methods, such as crystallography techniques, structural integrity investigations, and biochemical studies, have been utilized to determine the capabilities of proteins. Conversely, these methods call for a considerable amount of time and effort to employ. Utilizing computational methods as a means of predicting the function of proteins has become an increasingly important method over the course of the past few decades. The concept of protein transmission is utilized within the framework of these methods. The premise of this theory is that proteins that have sequences that are identical to one another are presumed to perform the same functions.[7]

A categorization system, sometimes called protein families, has been developed to shed light on protein metabolism's structure, function, and process. This system groups proteins that work similarly. This classification is especially helpful in identifying proteins that are difficult to characterize with conventional techniques. By putting specific sequences of proteins into categories, researchers can better understand the functioning of recently discovered proteins and discover significant biological observations from vast amounts of information.

In light of the ever-increasing explosion of biological facts, particularly data about proteins, there is an immediate and pressing requirement to develop more advanced protein classification methods. This issue can be resolved by employing cutting-edge resources and techniques, such as neural networks with deep connections and amino acid embedding, as demonstrated by the "ProtienCNN-BLSTM" model. These findings contribute to the advancement of bioinformatics and enhance our comprehension of the connections between the structure and function of proteins. A method of protein sequence classification that is both effective and precise has been developed as a consequence of this investigation.

1.2 | Problem statement

Proteins are essential in various biological processes, and the field of proteomics research seeks to anticipate the wide range of functions that proteins perform. Traditional molecular evaluation techniques frequently prioritize the analysis of complete sets of protein sequences rather than individual components. Hence, it is crucial to precisely categorize protein patterns into their corresponding families. The classification had a substantial beneficial effect on the outcomes of molecular assessment. Usually, the sequence of an unknown protein is compared to annotated sequences to reveal hints about its functions.[8]

The procedure of protein sequence classification, on the other hand, is both challenging and time-intensive, which renders the conventional method increasingly impractical. It is possible to gain valuable insights into a protein's composition, metabolic position, and functionality by accurately classifying sequences of proteins into their respective families. Due to proteins' complexity and ability to change structure while maintaining functionality, it can be not easy. An improved model that accurately classifies protein sequences is needed now.

1.3 | Main contributions

The ProteinCNN-BLSTM hybrid model classifies protein sequences using BLSTM networks and CNN. This hybrid architecture uses BLSTMs for long-range dependencies and CNNs for local features to improve classification accuracy and robustness. Major research contributions are listed below.

- The ProteinCNN-BLSTM hybrid model was constructed by combining amino acid embedding and bidirectional long short-term memory (BLSTM) networks with CNNs to classify protein sequences accurately and efficiently.
- Protein sequences were converted into numerical vectors using amino acid embedding, which enhanced feature representation.
- The proposed model improved prediction accuracy by utilizing long-term sequence dependencies and local feature extraction, with a 98.74% accuracy rate on the well-known datasets PDB-14189 and PDB-2272, and outperformed existing deep learning models, that is, CNN, LSTM, GCNs, CNN-LSTM, RNNs, GCN-RNN, DeepFam, and ProtCNN.

1.4 | Structure of the article

The following is the outline of the research article's internal structure. Section 2 presents a literature review, discussing the various machine learning and deep learning techniques currently used for classifying protein sequences and biological analysis. A description of the protein datasets is provided in Section 3, which also includes a discussion of the materials and methods utilized in this investigation. This section also provides specifics regarding the architecture and operation of the proposed hybrid model. Section 4 presents the results and discussion of the experiments, along with the specifics of the implementation and the simulation outcomes. Section 5 brings the research close and recommends further research that could lead to enhancements in protein sequence investigation.

2 | LITERATURE REVIEW

Over the last few years, the quantity and complexity of biological data have significantly increased. Because these data, which include sequences of proteins, RNA, and DNA, are available in various formats, it is necessary to develop more advanced computational methods to derive meaningful insights from them. This is necessary to make progress in bioinformatics and other related fields.

The investigator[7] employed improved recurrent neural networks to directly classify the functions of proteins based solely on the primary sequence. This was accomplished with the requirement for sequencing position, heuristics scoring, and feature engineering. In Reference,[8] a deep learning neural network based on spectral sequence representation was presented for DNA sequence classification. The results of this experiment demonstrated that the Deep Learning (DL) approach performed better than any of the other classifiers in the classification of short sequence fragments that were 400-plus base pairs in length.

The first step in the research project[9] was for the researchers to investigate the previous classification methods, specifically alignment methods, and to highlight the limitations of these methods. After that, they expanded their knowledge into deep learning, including artificial neural networks and hyperparameter tuning. In conclusion, they present the most recent and cutting-edge examples of deep learning architectures utilized in the classification of DNA. Researchers[10] have also presented two different deep-learning approaches for detecting DBPs. These approaches are referred to as DeepDBP-ANN and DeepDBP-CNN. This endeavor has resulted in establishing new benchmarks due to the exceptional performance of these methods on established standard datasets.

Convolutional neural networks (CNN) and long short-term memory frameworks can acquire more features and investigate the possible contextual correlations between specific amino acid sequences,[11] unlike traditional models, which cannot achieve the same result. The researcher utilized deep learning in a previous study[12] to identify DBPs, which they subsequently presented. The network quality was evaluated using a CNN and a long short-term memory (LSTM) with binary cross-entropy.

This method was used to create a deep-learning model. "DeepDRBP-2L" is the name of the two-level classifier researchers developed[13] to distinguish between DBPs and RBPs. To develop this predictor, a CNN, and an LSTM were combined. The MPPIF-Net framework was a novel framework introduced by researchers.[14] This framework used deep learning in conjunction with multilayer bidirectional LSTM to accurately identify mitochondrial proteins of the Plasmodium

falciparum parasite. The performance of this framework was superior to that of others that had been utilized in the past.

According to the findings of one researcher,[15] the PDBP-Fusion method is a technique that employs deep learning techniques to make predictions regarding DBPs. This objective was accomplished by incorporating local features and long-term dependencies derived from primary sequences and utilizing a Bi-LSTM network and a CNN. The researchers applied the method to the independent dataset PDB2272 and an online server to generate more accurate predictions of DBP. The researcher utilized a method known as transfer learning[16] to transfer samples and construct datasets. In this method, two features were extracted from a protein sequence, and two conventional transfer learning approaches were compared. A deep learning neural network model was developed during the final phase. This model utilized attention mechanisms in order to locate DBPs when they were needed. The DNA-binding protein classification methods that are currently in use have some drawbacks. They do not consider contextual information and cannot extract features from the sequence of amino acids.[17] As a result of the absence of contextual amino acid interactions, these methods frequently fail to identify DNA protein binding patterns.

One possibility is that the currently available methods do not adequately capture the full range of structural and functional properties of DNA-binding proteins. This is something that needs to be investigated further. This is something that needs to be investigated further. The complex interactions between DNA and proteins compound this problem. Protein sequences are classified using text categorization techniques based on natural language processing.[18] Text categorization has been enhanced by word embedding techniques and recently developed deep learning algorithms. These advancements have shown significant progress. These advancements have made it possible to significantly improve the efficiency of protein classification, which has opened up significant opportunities. However, several obstacles must be overcome to successfully adapt natural language processing word embedding methods to protein sequences.

In contrast to natural languages, protein sequences do not contain any distinct "words," and tasks utilized in natural language processing, such as next-word prediction, do not directly apply to protein sequences. The presence of distinct words characterizes natural languages. In addition, protein sequences are typically longer than sentences in natural language; however, they have a smaller alphabet, which adds a layer of complexity to the complexity of learning models.

Additionally, it has been demonstrated that pretraining can significantly improve the performance of machine learning models.[19] The field of computer vision was the first to make widespread use of pretraining, which has since been successfully applied in various other fields, including language processing. The Embeddings from Language Models (ELMo) and Bidirectional Encoder Representations from Transformers (BERT) models are examples of pre-trained models that offer excellent generalization capabilities and rapid convergence. These models are especially helpful for tasks that involve a limited amount of training data. On the other hand, these models frequently require significant computational resources.

Graph Convolutional Networks (GCNs) represent an additional data-driven neural network approach. GCNs can process biological sequences element-by-element and within their larger context. They can capture temporal relationships between hidden layer cells and forward-feeding links. Because of this capability, GCNs can develop memories and incorporate previous inputs, ultimately increasing their ability to predict. For example, SeqVec employs ELMo to represent protein sequences as embedding vectors to determine the biophysical properties of protein sequences.

On the other hand, ProtTrans uses transformer frameworks from natural language processing to provide pre-trained models for the representation of amino acid sequences.[20]

The computational power required for deep learning models, particularly those used in natural language processing, is significant. Pretraining on a comprehensive labeled dataset can transfer knowledge to smaller datasets, improving performance across various amino acid sequence tasks.[21] This is possible because protein sequence classification tasks share common pattern characteristics. Different kinds of biological data, such as protein sequences, three-dimensional structures, information about protein folding, protein–protein interactions, gene expression variations, amino acid families, and their interrelationships, can be used to classify the various methods used to predict the functions of proteins. These biological properties are currently presented digitally in many databases for computational analysis.

Decision trees, Support Vector Machines (SVMs), and NNNs are some statistical methods developed recently for data classification.[22] For example, one study used phrase segmentation followed by SVM classification to extract features from protein sequences. Another study, on the other hand, used SVM after calculating amino acid patterns to reduce vector dimensions. However, deep learning techniques have not been thoroughly investigated for large-scale protein function prediction. Existing research has primarily concentrated on relatively small protein groups and a restricted range of functional categories.

Various protein characteristics, including peptide sequences, three-dimensional structures, and protein interactions, have been utilized in training deep neural networks (DNNs) to predict the functions of proteins. Various architectures, including single and multitask feed-forward DNNs, Deep Auto Encoders, Recurrent Neural Networks (RNNs), and CNNs, were utilized in the deep neural networks (DNNs) training process. Despite the progress that has been made, the development of deep learning methods for predicting protein function is still in its early stages, which means there is much room for improvement. One of the most important areas of research is still the development of effective models that can address the specific challenges posed by protein sequences, such as their length and the limited alphabet they contain. This research is currently being carried out to improve the precision and effectiveness of protein classification, thereby making a significant contribution to the field of bioinformatics and the applications it is used for.[23]

Developing more advanced computational methods is necessary to derive useful insights from the ever-increasing complexity and abundance of biological data, particularly in the sequences of proteins, RNA, and DNA.[24] An analytical approach is necessary for bioinformatics and other related fields. Protein sequences have been the subject of extensive application of text categorization techniques that use natural language processing and other similar techniques. Deep learning algorithms and word embedding strategies have made significant strides in recent years, leading to significant improvements in text categorization. These advancements have opened up opportunities for significantly improving the efficiency of protein classification. However, challenges are in adapting natural language processing word embedding methods to protein sequences.

In contrast to natural languages, amino acid sequences do not contain any distinct "words," and tasks utilized in natural language processing, such as next-word prediction, do not directly apply to protein sequences. Furthermore, protein sequences are typically longer than sentences in natural language, but they have a smaller alphabet, adding complexity to learning models' complexity.[25] Table 1 compares various existing methods with the proposed methods.

TABLE 1 Comparison of various existing studies involving protein sequence analysis.

References	Key technique	Dataset used	Outcome	Limitations	Future scope
[8]	CNNs	Online UCI Dataset	Accuracy 91.74%, F1-score 0.89	Limited exploration of deeper CNN architectures	Explore deeper CNN architectures for feature extraction and optimize hyperparameters for better performance.
[9]	RNNs	DNA Data Bank of Japan	Accuracy 85.98%, Precision 0.87	Lack of attention mechanisms	Investigate hybrid models combining RNNs with attention mechanisms to improve sequence learning.
[10]	GCNs	European Nucleotide Archive	Accuracy 88.21%, Recall 0.88	Optimization of graph convolution layers needed	Optimize graph convolution layers, explore different graph embeddings, and consider using attention mechanisms.
[11]	Transformer Networks	The Consensus CDS Protein Set Database	Accuracy 84.74%, F1-score 0.86	Limited use of pre-trained models	Incorporate pre-trained transformer models, fine-tune on specific datasets, and explore multi-head attention for better context understanding.
[12]	Neural Network	Kaggle Online Dataset	Accuracy 83.78%, Precision 0.85	Overfitting issues	Integrate advanced neural architectures like ResNets DenseNets and employ dropout techniques to reduce overfitting.
[13]	GCNs, CNN, RNNs	SWISS-PROT Dataset	GCNs Accuracy 90.32%, AUC 0.92	Need for ensemble methods	Use ensemble methods and apply feature selection techniques to improve model performance.
[14]	CNN and RNN	PROSITE Database	Accuracy 92.98%, F1-score 0.93	Lack of attention layers	Enhance the model with attention layers to improve sequence learning and utilize more sophisticated RNN variants like GRU.
[15]	NLP with Machine Learning	Kaggle Online Dataset	Accuracy 83.87%, Recall 0.84	Limited use of transfer learning	Implement transfer learning techniques with pre-trained NLP models, such as BERT or GPT, and fine-tune them for better performance.
Proposed	CNN and BLSTM with Amino Acid Embedding	Kaggle Online Dataset	Accuracy 95.09%, Precision 0.96, Recall 0.94	High time complexity	Reduce time complexity using model optimization techniques, explore automated hyperparameter tuning, and investigate the integration of additional biological information for enhanced prediction accuracy.

3 | MATERIALS AND METHODS

In this section, we begin by introducing the datasets utilized in the model's development. After that, an explanation of the conceptual structure and test methods suggested in the present article is given. Finally, a model algorithm was used in the demonstration.

3.1 | Representation of proteins

Proteins are usually composed of many residues that do not always have clear spatial relationships. It is, therefore, more appropriate to represent these residues as nodes to capture their structural characteristics. The spatial arrangement of a protein is visually represented by a contact map, which shows the protein's two-dimensional structure. With values ranging from 0 to 1, each contact map data matrix element represents the likelihood of contact at a particular location. A contact map is made to predict a protein's structure based on its sequence.[26] The dimensions of the contact map for a protein sequence of length L are (L × L). The possibility of an interaction between the *i*th and *j*th residues is indicated by the element L (i, j). If this value exceeds a predetermined threshold, residues are deemed to be in contact. Pconsc-4 is a fast and effective way to forecast contact maps. For the contact maps, a threshold of 0.55 is selected because its output is a probability estimate with a range of 0–1. Probabilities greater than or equal to 0.55 are assigned a value of 1, while all other values are set to 0. GCNs require the adjacency matrix, produced by thresholding, to extract the protein's molecular structure efficiently.

The protein then acquires its characteristics. As nodes in the network, residues are defined by several characteristics, including apparent valence, orientation, and aromatic properties. These differences result from differences between amino acid R groups. Proteins can be uniquely represented using position-specific scoring matrices (PSSMs), which rely on sequence alignment results to determine the impact of each element.[27]

- *PSSMs*: A protein sequence is aligned with a database that contains known protein sequences to create a PSSM. Every component in the PSSM denotes a specific amino acid position score at a given sequence position. Based on evolutionary data, this score indicates the probability of that amino acid occurring at that location. Understanding proteins' structural and functional characteristics is aided by PSSMs, which record residue variability and conservation across various sequences.
- *LFVM*: The PSSM is the basis for the local frequency vector matrix (LFVM).[28] The average frequency of every residue across all sequence alignment results is represented. By offering a vector of average residue frequencies, the LFVM effectively condenses the PSSM and aids in identifying the general amino acid composition and local sequence context. Identifying conserved areas and functional motifs in a protein sequence depends on this matrix.

There are 54 attributes because features are extracted using primary component analysis, LFVM, PSSM, and other factors. As a result, a protein's feature matrix has dimensions (Mf, 54). Integrating these matrices and features can effectively represent protein sequences, making prediction and classification tasks more precise. Features Matrix can be calculated by Equation (1).

$$\mathrm{Mf}_{(\mathrm{i,j})}^{\mathrm{LFVM}} = \left\{ \sum_{k=1}^{n} J\left(B_{(k,j)} = i\right) \right\}, \tag{1}$$

where Mf represents the feature matrix of a protein. B represents a collection of proper alignment patterns with lengths equal to those of the desired protein, k is a collection of residues, ($k = 1, 2, N$), ($j = 1, 2, L$), and $k(x)$ is an indicator function that determines whether the condition is satisfied or not. Eq calculates the "location probability distribution matrix" (LPM).

$$\mathrm{Mf}_{(i,j)}^{\mathrm{LPM}} = \left\{ \frac{\mathrm{Mf}_{(i,j)}^{\mathrm{LPM}} + \frac{P}{4}}{(N+P)} \right\}. \tag{2}$$

Based on the psychology of humans, a pseudocount pc was set to (0.8) to make the LPM one of the node features while preventing matrix transactions from reaching 0.

3.2 | Dataset

We utilized the widely recognized and accessible PDB-14189 dataset from the Kaggle PDB dataset for training and the PDB-2272 dataset for testing. Both datasets are derived from UniProt records.[10] The PDB-14189 dataset comprises 7129 positive patterns and 1153 negative patterns. In contrast, the PDB-2272 dataset, used exclusively for testing, includes 7060 positive and 1119 negative patterns. The datasets are implemented primarily for testing and comparison with alternative approaches[6] to assess and validate the effectiveness of our method. The sequence similarity within the PDB-14189 dataset is capped at 40%, while the sequence similarity in the PDB-2272 dataset does not exceed 25%. Table 2 presents the total samples with positive or negative results in the PDB-14189 and PDB-2272 records.

Table 3 represents the protein sequences in the dataset.[22] We have also utilized Prokaryotic Proteins[21] and Eukaryotic Proteins[23] datasets to measure the universality of the ProteinCNN-BLSTM model across different organisms.

3.3 | Proposed hybrid model

The proposed ProteinCNN-BLSTM model combines the strengths of CNNs and BLSTM networks to classify protein sequences efficiently and accurately. Enhanced prediction accuracy is achieved using amino acid embeddings in this hybrid approach, which is utilized for feature extraction and classification.[25,27] Figure 1 presents the architecture of the proposed hybrid model. The complete operation of the proposed model is described in the subsections below.

3.3.1 | Data pre-processing and amino acid embedding

Specifically, the following procedures were carried out to perform data pre-processing and amino acid embedding.[28]

TABLE 2 Dataset description.

Dataset	Positive instance	Negative instance	Total instances
PDB-14189	7129	1153	8882
PDB-2272	7060	1119	8179

TABLE 3 Protein sequence and count details.

Structure Id	Chain Id	Data sequence	Residue count	Macromolecule type
100D	A	CCGGCGCCGG	20	DNA/RNA Hybrid
100D	B	CCGGCGCCGG	20	DNA/RNA Hybrid
101D	A	CGCGAATTCGCG	24	DNA
101D	B	CGCGAATTCGCG	24	DNA
101 M	A	MVLSEGEWQLVLHVWAKVE ADVAGHGQDILIRLFKSHPET LEKFDR …	154	Protein
102D	A	CGCAAATTTGCG	24	DNA
102D	B	CGCAAATTTGCG	24	DNA
102L	A	MNIFEMLRIDEGLRLKIYKDTE GYYTIGIGHLLTKSPSLNAAAKSE .	165	Protein
102M	A	MVLSEGEWQLVLHVWAKV EADVAGHGQDILIRLFKSHPET LEKFDR …	154	Protein
103D	A	GTGGAATGGAAC	24	DNA

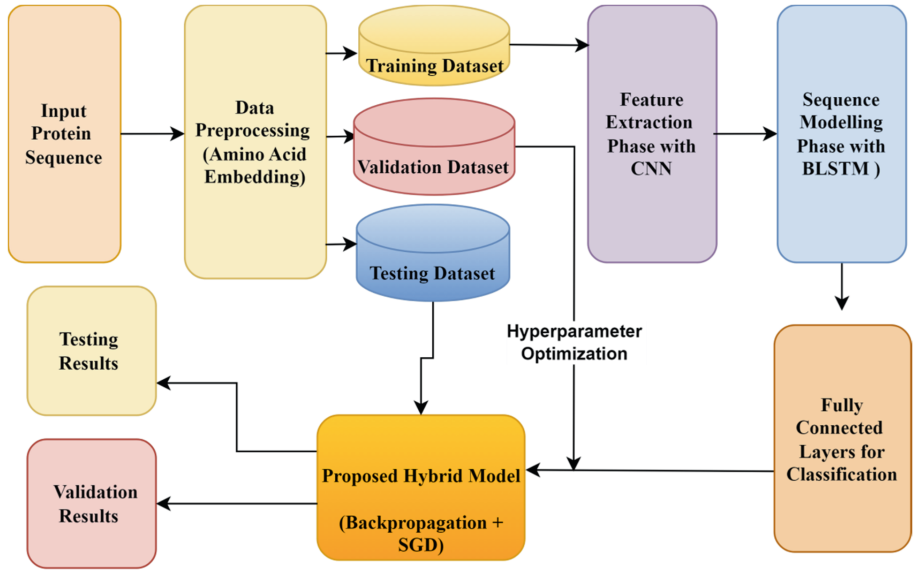


FIGURE 1 Architecture of the proposed protein CNN-BLSTM model.

- **Step 1: Load protein sequence data**: Data pre-processing is crucial in preparing protein sequence information for deep learning models. It is necessary to ensure smooth and efficient model preparation. Two databases that fall into this category are the PDB-14189 and the PDB-2272. It is important to follow these steps to ensure that the data is organized most efficiently for evaluation and training.

  - **Initial step**: During the data acquisition process, the protein sequence data were sourced from reputable datasets like PDB-14189 and PDB-2272. Protein sequences are essential for training deep learning algorithms. Databases like this provide meticulously curated collections of protein sequences, ensuring their quality and usefulness.
  - **Analysis of the Initial Exploratory Phase**: To understand the protein patterns' basic properties and spatial distribution, exploratory data analysis (EDA) is required. This process involves considering various factors, such as the length of sequences, the presence of different amino acids, and detecting any missing or abnormal data. Exploratory Data Analysis (EDA) is valuable in identifying potential issues and developing a comprehensive understanding of the data structure.[29] It is advantageous for identifying possible issues.

- **Step 2: Amino acid embedding**: Create numerical representations of the protein sequences by applying amino acid embedding methods. A high-dimensional vector encapsulating its contextual relationships and biochemical characteristics represents every amino acid in a protein sequence. The need for numerical input in neural networks makes this transformation crucial.

  Amino acid sequences, for instance, can be translated into vectors of a fixed length using an embedding layer. For instance, a sequence of 31 amino acids can be converted into a vector of dimensions (31, 64). Figure 2 shows the results of amino acid embedding.[30]
- **Step 3: Splitting the dataset:** The dataset is divided into three subsets: testing, validation, and training. To avoid overfitting and evaluate the model's capacity for generalization, this section ensures that the model can be trained, validated, and tested on various subsets of the data.

3.3.2 | CNN Feature extraction phase

CNNs are used in the proposed model's feature extraction phase to extract pertinent features from the encoded protein sequences. In the CNN architecture, each layer carries out particular functions to capture crucial aspects of the input data.[31] Figure 3 presents the architecture of the CNN model. The complete operation of each layer is described below.

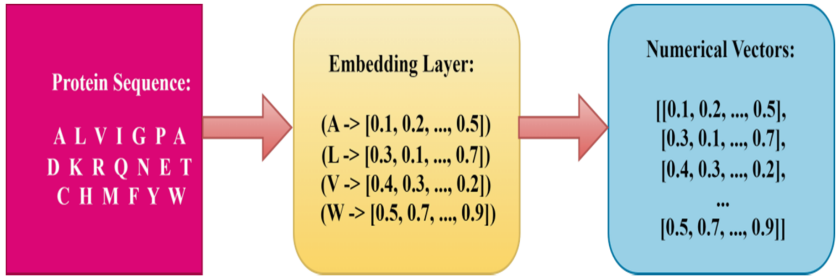


FIGURE 2 Amino acid embedding.

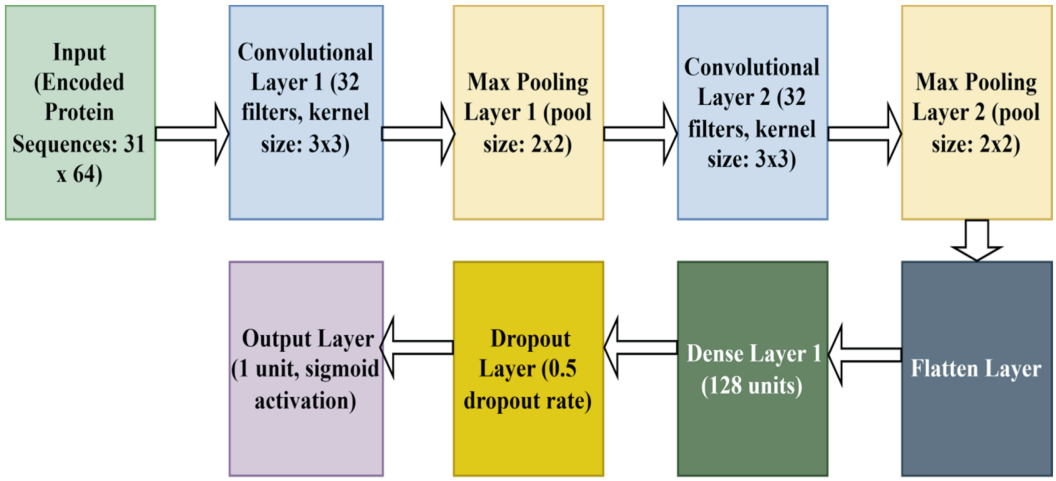


FIGURE 3 Architecture of the CNN model in the proposed hybrid model.

Convolutional layer-1

This layer utilizes a series of filters to analyze the input sequences and identify specific local patterns. Every filter examines the input sequences, detecting distinct characteristics. The result of this layer is a feature map that includes the identified patterns. Concerning categorization error, CNN significantly outperformed the earlier deep neural network algorithms. Convolution can be a single, two, or multiple dimensions.[32,33] In the proposed hybrid model, we utilized a one-dimensional convolutional model. A discrete sequence of variables $[\gamma = b_1, b_2 \ldots .b_i]$ and $[\varphi = c_1 \ldots \ c_j]$ is obtained. Then, the convolutional product of γ and φ can be presented as Equation (3).

$$[\gamma * \varphi\ ] = \left[\sum_{k=0}^{j} a_{id+n-1} * b_m\right]. \tag{3}$$

For $i = 1$ to length.

With the help of these layers, CNN can extract protein sequence hierarchical representations and identify local and global patterns crucial for classification.

Max pooling layer 1

The feature maps are down-sampled via max pooling to reduce the overall spatial dimensions while preserving the most crucial information after convolution. This layer aids in minimizing overfitting and lowering computational complexity. Considering the k channel input $I = \left(b_{i,j,k}\right)$, the maximal pooling functioning MF is established by Equation (4).

$$MF = Max\left(b_{i,j,k}\right). \tag{4}$$

Convolutional layer 2

Like the first layer, this layer further extracts more advanced characteristics from the feature maps obtained through the previous layer by down-sampling them. More filters are applied to capture more intricate patterns.

Max pooling layer 2
This layer uses an additional phase of max pooling to reduce the size of the feature maps while retaining the most important features.

Flatten layer
The final step involves flattening the output feature maps into a one-dimensional vector, which is then used as the input for the model's later layers, including the BLSTM layer.

3.3.3 | Sequence modeling phase with BLSTM

The BLSM is divided into various subtasks, completed by its specific layers. Together, these layers enable the suggested research model to accurately predict protein sequence classifications by capturing protein sequences' structural and sequential information.[34] Figure 4 presents the architecture of the CNN model in the proposed hybrid model.

Input layer
The CNN phase's features are transferred to this layer. Encoded into numerical vectors, these features reflect the fundamental properties of the input protein sequences.

BLSTM layer-1
The bidirectional LSTM layer receives the feature maps acquired from the CNN. By capturing all of the sequential dependencies in the backward and forward directions, this layer improves the model's comprehension of the protein sequence context. Figure 5A shows the structure of an RNN model. In addition to being dependent upon the current input, the undiscovered state beginning at that particular step depends upon the earlier hidden state, which was computed. This meant that the secret state (H_{st}) could not be determined solely by the present input, as described in Equation (5).

$$H_{st} = Af[(M_{st} * W) + (H_{st-1} * U) + \varphi, \tag{5}$$

where Af and Bf are the activation functions, Af and Bf are the bias, W is the weight, H_{st} is the hidden state, and O_t is the output. An Output O_t Concerning the time, t can be determined by Equation (6). The symbol details are presented in Table 4.

$$O_t = Bf[(H_{st}S) + \varphi, \tag{6}$$

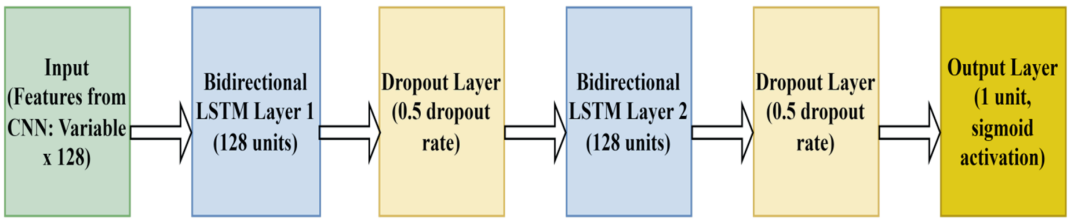


FIGURE 4 Architecture of the BiLSTM model in the proposed hybrid model.

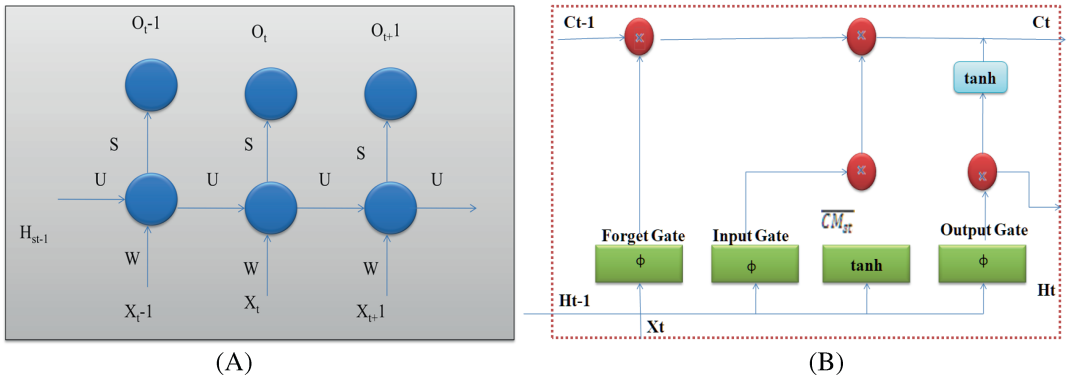


FIGURE 5 (A): Structure of RNN 6 and (B) Structure of LSTM.

TABLE 4 The symbol details used in equations.

Symbol	Description
$\boldsymbol{H_{st}}$	Hidden state at time step *t*
$\boldsymbol{M_{st}}$	Feature map acquired from the CNN at time step t
$\boldsymbol{W}$	Weight matrix
U	Weight matrix for the hidden state
V	Bias
$\boldsymbol{Af}$	Activation function
$\boldsymbol{Bf}$	Bias for the activation function
$\boldsymbol{\varphi}$	Activation function (tanh for LSTM)
$\boldsymbol{O_t}$	Output at time step t
$\boldsymbol{I}_{\text{gate}}$	Input gate
$\boldsymbol{F}_{\text{gate}}$	Forget gate
$\boldsymbol{O}_{\text{gate}}$	Output gate
$\boldsymbol{\overline{CM_{st}}}$	Candidate memory cell at time step t
$\boldsymbol{CM_{st}}$	Candidate memory cell (output of the tanh function for LSTM) at time step *t*
	Tanh function applied to the candidate memory cell at time step t
$\boldsymbol{W_{x,CM_{st}}}$ and $\boldsymbol{W_{x,CM_{st}}}$	LSTM input-to-candidate memory weights and hidden-state-to-candidate memory weights

There was an insurmountable problem with the RNN when dealing with lengthy sequences shown in Figure 5A and that problem was known as the disappearing gradient. The linear scaling technique (i.e., LSTM) constitutes one of the better options for dealing with disappearing gradients. As demonstrated in Figure 5B, an LSTM consists of a possible memory cell together with three gates, which are a gate for inputting data (I_{gate}), a gate to output (O_{gate}), and a gate to forget (F_{gate}).[35]

Equations (5)–(7) were used to individually calculate the forget, input, and output gates at each time step. Where $W_{x,f_{gate}}$ and $W_{h,f_{gate}}$ are the LSTM weights from the input and forgotten gates, respectively. Similarly, $W_{x,I_{gate}}$ and $W_{h,I_{gate}}$ represent the LSTM link weights from the input and hidden states, respectively, and $W_{x,O_{gate}}$ and $W_{h,O_{gate}}$ represent the LSTM link weights from the input and output/resulting gates. $V_{f_{gate}}$, $V_{I_{gate}}$ and $V_{O_{gate}}$ represent the biases for the forget, input, and output gates, respectively, as presented in Equations (7)–(9).

$$F_{gate} = \varphi\left[\left(M_{st} * W_{x,f_{gate}}\right) + \left(H_{st-1} * W_{h,f_{gate}}\right) + V_{f_{gate}}\right], \tag{7}$$

$$I_{gate} = \varphi\left[\left(M_{st} * W_{x,I_{gate}}\right) + \left(H_{st-1} * W_{h,I_{gate}}\right) + V_{I_{gate}}\right], \tag{8}$$

$$O_{gate} = \varphi\left[\left(M_{st} * W_{x,O_{gate}}\right) + \left(H_{st-1} * W_{h,O_{gate}}\right) + V_{O_{gate}}\right]. \tag{9}$$

For LSTM, a candidate memory cell (CM_{st}) can be determined by Equation (10). $W_{x,CM_{st}}$ and $W_{x,CM_{st}}$ represent the LSTM input-to-candidate memory weights and hidden-state-to-candidate memory weights, respectively, and $V_{CM_{st}}$ represents the bias.

$$\dot{\overline{CM_{st}}} = tanh\left[\left(M_{st} * W_{x,CM_{st}}\right) + \left(H_{st-1} * W_{x,CM_{st}}\right) + V_{CM_{st}}\right]. \tag{10}$$

A memory cell at time interval t can be determined by Equation (11) as follows.

$$CM_{st} = \left[F_{gate} \times CM_{st-1}\right] + \left[I_{gate} \times \dot{\overline{CM_{st}}}\right], \tag{11}$$

where $\times$ represents element-based multiplication, and a hidden state can be determined by Equation (12) as follows.

$$H_{st} = \left[I_{gate} \times \tanh(\, CM_{st})\,\right]. \tag{12}$$

Dropout layer-1

Dropout is a regularization technique that prevents neural networks from overfitting. This layer helps prevent the network from becoming overly dependent on particular features or connections by randomly setting 50% of the input units to zero during training.[36]

Bidirectional LSTM layer 2

This layer functions on the first layer's outputs and is comparable to the first bidirectional LSTM layer. Capturing more temporal dependencies enhances the representation of the input sequences even more.

Dropout layer 2

A second dropout layer is added after the second bidirectional LSTM layer to offer more regularization and avoid overfitting.

Output layer

One neuron with a sigmoid activation function makes up the output layer. The input protein sequence is classified as belonging to a particular class or category based on the production's

binary classification output. The sigmoid activation function squashes the output values to 0 and 1, representing the likelihood that the input sequence is in the positive class.

3.3.4 | Classification by fully connected layer

Once the Bidirectional Long-Long-term memory layers have been used for sequence modeling, the processed features must be categorized into the proper groups. It is a dense, fully connected layer that performs this classification.[37]

Input
The final BLSTM layer's output is fed into the fully connected layer. The output of this process is a high-dimensional representation of the protein sequences that include temporal and spatial dependencies.

Neurons and weights
Each neuron in the fully connected layer is connected to every other neuron in the preceding BLSTM layer. Every connection has a weight attached to it, which is modified during training. Although they can vary, the fully connected layer's neuron count usually corresponds to the number of classes in the classification task. One neuron is typically used for binary classification, while multiple neurons are used for multiclass classification, with one neuron for each class.

Activation function
The fully connected layer's output is subjected to an activation function, such as the sigmoid or SoftMax. Sigmoid activation: This binary classification method produces an output value ranging from 0 to 1, which indicates the likelihood that the input falls into the positive class. When applied to multiclass classification, softmax activation produces a probability distribution across all classes and guarantees that the total probability equals 1.

Output
The classification outcome is what the fully connected layer ultimately produces. For every class, it offers the probability or confidence score that the model uses to generate its final prediction. A threshold, such as 0.5, is applied to the sigmoid output in binary classification to determine the class. The predicted class is determined for multiclass classification by selecting the class with the highest probability from the SoftMax output.

3.3.5 | Backpropagation and training

Backpropagation is used to update the weights of the fully connected layer during training. A loss function determines the error amount between the predicted output and the actual labels (ground truth). For example, binary cross-entropy is utilized for binary classification, while categorical cross-entropy is utilized for multiclass classification. The loss gradients concerning the weights are calculated and then adjusted to minimize the error.[38]

3.4 | Algorithm for the proposed hybrid model

Algorithm 1 and pseudocode 1 present the key functions and steps of the proposed CNN-BLSTM model.

Pseudo Code 1: Key Function of Proposed ProtienCNN-BLSTM model for Protein sequence analysis

```
# Load and preprocess protein sequences
sequences = load_sequences() # Function to load raw protein sequences
encoded_sequences = encode_sequences(sequences) # Function to encode sequences into numerical format
# Extract features using a CNN model
cnn_model = build_cnn_model() # Function to define and compile the CNN model
cnn_features = cnn_model(encoded_sequences) # Extract features from encoded sequences using the CNN model
# Model sequence dependencies using an LSTM model
lstm_model = build_lstm_model() # Function to define and compile the LSTM model
lstm_output = lstm_model(cnn_features) # Model the sequence using LSTM on CNN features
# Train and evaluate the hybrid CNN-LSTM model
train_set, val_set = split_dataset(encoded_sequences, labels) # Split data into training and validation sets
model = build_cnn_lstm_model() # Function to build and compile the hybrid CNN-LSTM model
model.fit(train_set, labels_train, validation_data=(val_set, labels_val), epochs=50) # Train the model
# Predict function for new sequences
new_sequence = load_new_sequence() # Load a new protein sequence
encoded_new_sequence = encode_sequence(new_sequence) # Encode the new sequence
prediction = model.predict(encoded_new_sequence) # Make a prediction using the trained model
```

Algorithm 1. Proposed ProtienCNN-BLSTM model for Protein sequence analysis

Input: Protein sequence data
Output: Predictions for protein properties or functions
Step 1: Data Preprocessing
1.1 Load and Explore Data

- *Load the protein sequence dataset.*
- *Conduct exploratory data analysis (EDA) to understand sequence distribution, length, and overall characteristics.*

1.2 Encode Protein Sequences

- *Transform protein sequences into numerical representations suitable for neural network input (e.g., one-hot encoding, amino acid embeddings).*

1.3 Split Dataset

- *Partition of the dataset into training, validation, and testing subsets to ensure unbiased model evaluation and tuning.*

Step 2: Feature Extraction
2.1 Build CNN Architecture

- *Design a CNN) to accept encoded protein sequences as input, facilitating local feature extraction.*

2.2 Generate Feature Maps

- *Apply the CNN to produce feature maps, highlighting significant motifs and structural patterns within the sequences.*

2.3 Identify Biological Patterns

- *Utilize the feature maps to capture and represent critical biological patterns and protein sequence regularities.*

Step 3: Sequence Modeling.
3.1 Build LSTM Network

- *Construct an LSTM network to model temporal dependencies and relationships within the protein sequences, using the feature maps from the CNN as inputs.*

3.2 Learn Complex Interactions

- *The LSTM network learns and identifies intricate patterns and interactions among amino acids in the sequences, capturing long-range dependencies.*

Step 4: Model Training and Evaluation
4.1 Train the CNN-BLSTM Model

- *Train the hybrid CNN-BLSTM model using backpropagation and stochastic gradient descent (SGD) on the training dataset, optimizing model weights iteratively.*

4.2 Hyperparameter Tuning and Evaluation

- *Validate the model on the validation dataset to fine-tune hyperparameters (e.g., learning rate, batch size).*
- *Evaluate the final model performance on the test dataset to ensure robustness and generalization.*

Step 5: Prediction and Application
5.1 Encode New Protein Sequences

- *Convert new protein sequences into a numerical format compatible with the trained model.*

5.2 Predict Protein Characteristics

- *Input new encoded sequences into the trained CNN-BLSTM model to predict protein characteristics or functions, providing insights into previously uncharacterized proteins.*

4 | EXPERIMENTAL RESULTS AND DISCUSSION

The proposed hybrid CNN-BLSTM method and existing CNN,[1] LSTM,[2] GCNs,[13] CNN-LSTM,[3] RNNs,[4] GCN-RNN,[24] Deep Family-based Method (DeepFam)[5] and Protein Convolutional Neural Network (ProtCNN)[6] state-of-the-art techniques were implemented on two popular protein datasets, PDB-14189 and PDB-2272, and various performance measurement parameters were calculated. The complete details are covered in the next subsections.

4.1 | Performance measuring parameters

To measure the correctness of the predictions of the proposed model and existing models, we established a variety of metrics generally employed for binary classification problems, including specificity (SPC), sensitivity (SNS), Matthews's correlation coefficient (MCOC), and accuracy (ACR), which can be determined by Equations (13)–(16).[39] Here, TPR: true positive rate, FPR: false positive rate, TNR: true negative rate, FNR: false negative rate.

$$ACR = \frac{[TPR + TNR]}{(TPR + FPR) + (TNR + FNR)}, \tag{13}$$

$$SPC = \frac{[TNR]}{(FPR) + TNR)}, \tag{14}$$

$$SNS = \frac{[TPR]}{TPR + FNR)}, \tag{15}$$

$$MCOC = \frac{[TPR \times TNR] - [FPR \times FNR]}{(TPR + FNR)(TPR + FPR)(TNR + FNR)(TNR + FPR)}. \tag{16}$$

Within this context, TPR represents the quantity of correctly anticipated positive samples, TNR represents the quantity of accurately predicted negative samples, FPR represents the quantity of erroneously anticipated positive specimens, and FNR represents the quantity of incorrectly anticipated negative specimens. The metrics SNS, SPC, ACR, and MCOC are utilized to evaluate the performance of a classification framework. SN denotes the proportion of precisely anticipated positive specimens, while SPC denotes the ratio of accurately predicted samples that were negative. The ACR indicates the overall proportion of accurately anticipated samples in the dataset. Finally, the MCOC measures the forecasting ability of the classification framework with an appropriate range of values between (−1 and 1). The degree of accuracy in the framework's predictions is positively correlated with the magnitude of the MCOC.

4.2 | Simulation setup and hyperparameter tuning

The simulation setup includes hardware software setups, parameter selection for model implementation, and hyperparameters for model training. Table 4 presents the key details of the hardware and software used.

TABLE 5 The hardware and software details.

Category	Details
Processor	Intel Core i7-9700K
RAM	32 GB
GPU	NVIDIA GeForce RTX 2080 Ti
Storage	1 TB SSD
Operating system	Ubuntu 20.04 LTS
Programming language	Python 3.8
Deep learning frameworks	PyTorch 1.7.0, Keras 2.4.3
Libraries	NumPy, pandas, scikit-learn, Matplotlib
Development environment	Jupyter Notebook, PyCharm

4.2.1 | Hardware and software setup

The hardware and software parameters presented in Table 5 were used to implement the proposed and existing models.

4.2.2 | Hyperparameter selection

We employed a strict procedure to choose the ideal hyperparameters, guaranteeing the model's performance and efficiency, to successfully classify protein sequences using our hybrid ProteinCNN-BLSTM model. Several crucial steps specifically designed to fit the distinctive qualities of the protein data were involved in the selection process. Our initial objective was to classify protein sequences, so we examined the PDB-14189 and PDB-2272 datasets to determine their characteristics, including class imbalance and sequence length distribution.[40]

Based on this analysis, we were able to determine the following critical hyperparameters that are essential to the performance of our model: the optimizer type, learning rate, batch size, number of epochs, dropout rate, and number of filters in the CNN layers, number of BLSTM units, and number of epochs. Table 6 presents the Hyperparameters used in the experiments.

Table 6 presents a detailed description of the hyperparameters used in the experiments. We considered our preliminary experiments and standard values from related studies when determining our initial hyperparameter ranges. In particular, we investigated learning rates of 0.0001, 0.001, and 0.01; batch sizes of 16, 32, and 64; dropout rates of 0.3, 0.5, and 0.7; and numbers of BLSTM units of 64, 128, and 256; and numbers of filters in the CNN layers of 32, 64, and 128. Additionally, we contrasted several optimizers, such as Adam, RMSprop,[41] and SGD. We used several search techniques to identify the optimal hyperparameters. While a random search offered a more effective option by sampling random combinations, a grid search allowed us to investigate every possible combination methodically. Additionally, we leveraged probabilistic models through Bayesian optimization to direct our search toward promising areas within the hyperparameter space. Another technique we used was broadband, which effectively distributes resources to rapidly assess many configurations.

Using k-fold cross-validation, we verified our hyperparameter selections to ensure that our findings were reliable and independent of any particular subset of the data. The efficacy of each

TABLE 6 Hyperparameters used in the experiments.

Hyperparameter	Suggested value(s)	Tuning process
Learning rate	0.0001, 0.001, and 0.01	It started with 0.001 and is reduced by a factor of 10 if the training loss plateaus.
Batch size	32, 64,128	A larger batch for stable training.
Number of epochs	50 to 100	Monitor the validation loss.
Dropout rate	0.3	To prevent overfitting.
Number of filters in CNN Layers	64, 128	Ensure the model complexity is balanced to prevent overfitting.
Kernel size in CNN layers	3	For capturing local features.
Number of LSTM units	128 or 256	Begin with 128 units and increase to 256 if more capacity is needed to capture sequence dependencies.
Bidirectional LSTM (BLSTM) layers	Two layers	Two layers for more complex sequence relationships.
Activation functions	ReLU for CNN layers, tanh for LSTM layers	ReLU is effective for CNNs, and tanh is effective for LSTMs for capturing nonlinear relationships.
Optimiser	Adam, RMSprop, and SGD	Adam optimizer is robust and Ensures fine-tuning learning rates.
Weight initialization	"He" initialization for CNN layers, "Glorot" initialization for LSTM layers	'He' initialization works well with ReLU activations, while Glorot initialization suits Tanh activations.
Regularization (L2 Regularization)	1e-4	Applied L2 regularization to prevent overfitting, especially in dense layers.

hyperparameter configuration was assessed using performance metrics. To improve the performance of our model, fine-tuning entailed reducing the ranges in light of preliminary findings and performing additional tuning. We validated it using a different validation set to ensure that the final model did not overfit and performed well when applied to fresh data. Finally, we tested the refined model on a separate test set and verified that it could correctly categorize protein sequences. We ensured that our ProteinCNN-BLSTM model was finely tuned and achieved superior performance in protein sequence classification by adhering to this methodical and thorough process. The meticulous choice of hyperparameters played a pivotal role in our model's remarkable accuracy of 98.11%, outperforming previous techniques and demonstrating its resilience and efficiency.

4.2.3 | CNN-LSTM parameters

The proposed hybrid model utilized the features of the CNN and BLSTM models with amino acid embedding; the key parameters used for the CNN and BLSM models are presented in Table 7.

TABLE 7 Parameter details of the CNN and BLSTM in the proposed hybrid model.

Layer (type)	Parameters	Output shape
embedding (embedding)	1472	(None, 31, 64)
dropout (dropout)	0	(None, 31, 64)
pooling (pooling)	0	(None, 32, 0)
conv1d (Conv1D)	9344	(None, 28, 128)
max_pooling1d (MaxPooling1D)	0	(None, 14, 128)
conv1d_1 (Conv1D)	49,216	(None, 14, 64)
max_pooling1d_1 (MaxPooling1D)	0	(None, 7, 64)
conv1d_2 (Conv1D)	6176	(None, 7, 32)
max_pooling1d_2 (MaxPooling1D)	0	(None, 3, 32)
dense_1 (Dense)	6726	(None, 6)
dense (Dense)	1,255,520	(None, 1120)
BLSTM (Bidirectional LSTM)	197,632	(None, 31, 256)
flatten (Flatten)	0	(None, 7936)

4.3 | Simulation results

The following simulation results were calculated for the proposed hybrid method, CNN-BLSTM, and existing methods, that is, CNN, LSTM, GCNs, CNN-LSTM, RNNs, GCN-RNN, DeepFam, and ProtCNN, on two standard protein datasets.

4.3.1 | Using the PDB-14189 dataset

The PDB-14189 dataset was divided using k-fold cross-validation, where 70% was used for training, 15% for testing, and 15% for validation.

Standard results
The simulation results are calculated for different performance measurement parameters for 50 and 100 epochs for the proposed and existing models, as presented in Figure 6.

Learning rates
The simulation results are calculated for different learning rates, that is, 0.0001, 0.001, and 0.01, on the PDB-14189 protein dataset for the proposed and existing state-of-the-art methods. Figure 7 presents the simulation results on the test dataset for different learning rates.

Different batch sizes
The simulation results are calculated at different learning rates, i.e., 32, 64,128, on the PDB-14189 protein dataset for the proposed and existing state-of-the-art methods. Figure 8 presents the simulation results on the test dataset for different batch sizes.

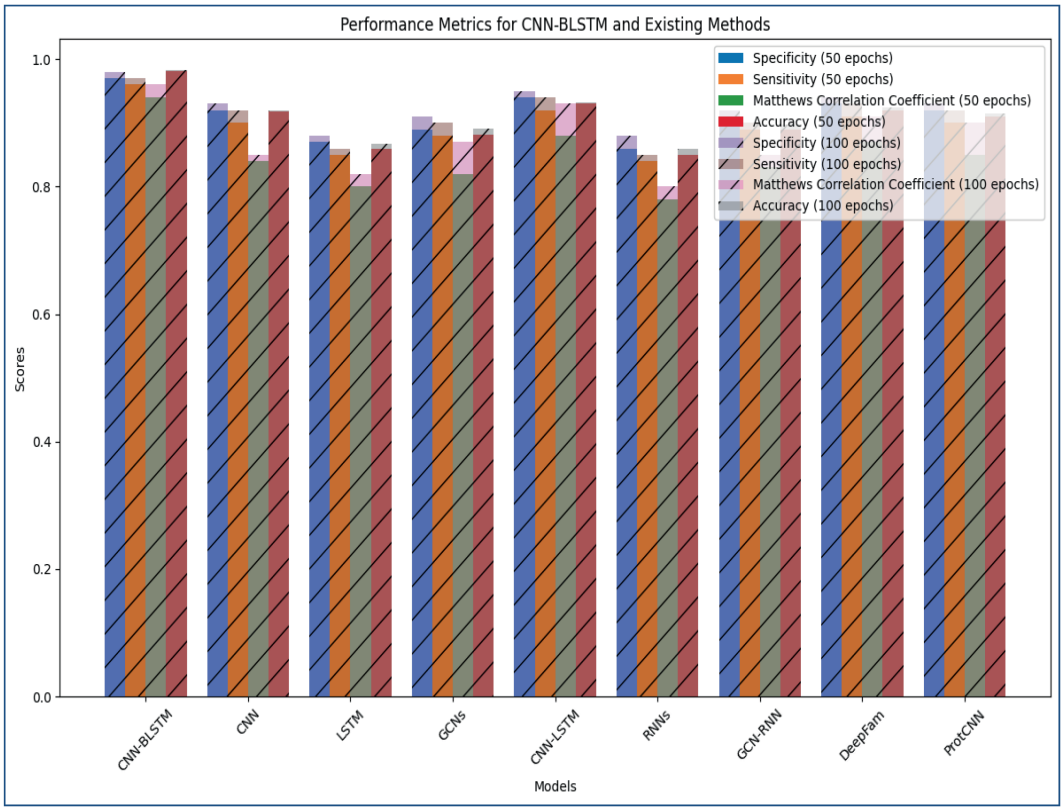


FIGURE 6 Simulation results for different epochs versus performance measuring parameters for PDB-14189.

Different optimizers

The simulation results are calculated at different learning rates, that is, Adam, RMSprop, and SGD, on the PDB-14189 protein dataset for the proposed and existing state-of-the-art methods. Figure 9 presents the simulation results on the test dataset for different optimizers.

Different dropout rates

The simulation results are calculated for different dropout rates, that is, 01, 0.2, 0.3, and 0.4, over the PDB-14189 dataset for the proposed and existing state-of-the-art methods. Figure 10 presents the simulation results on the test dataset for different dropout rates.

4.3.2 | Using the PDB-2272 dataset

The PDB-2272 dataset was divided using k-fold cross-validation, where 70% was used for training, 15% for testing, and 15% for validation.

Different batch sizes

The simulation results are calculated at different learning rates, that is, 32, 64, and 128, on the PDB-2272 protein dataset for the proposed and existing state-of-the-art methods. Figure 11 presents the simulation results on the test dataset for different batch sizes.

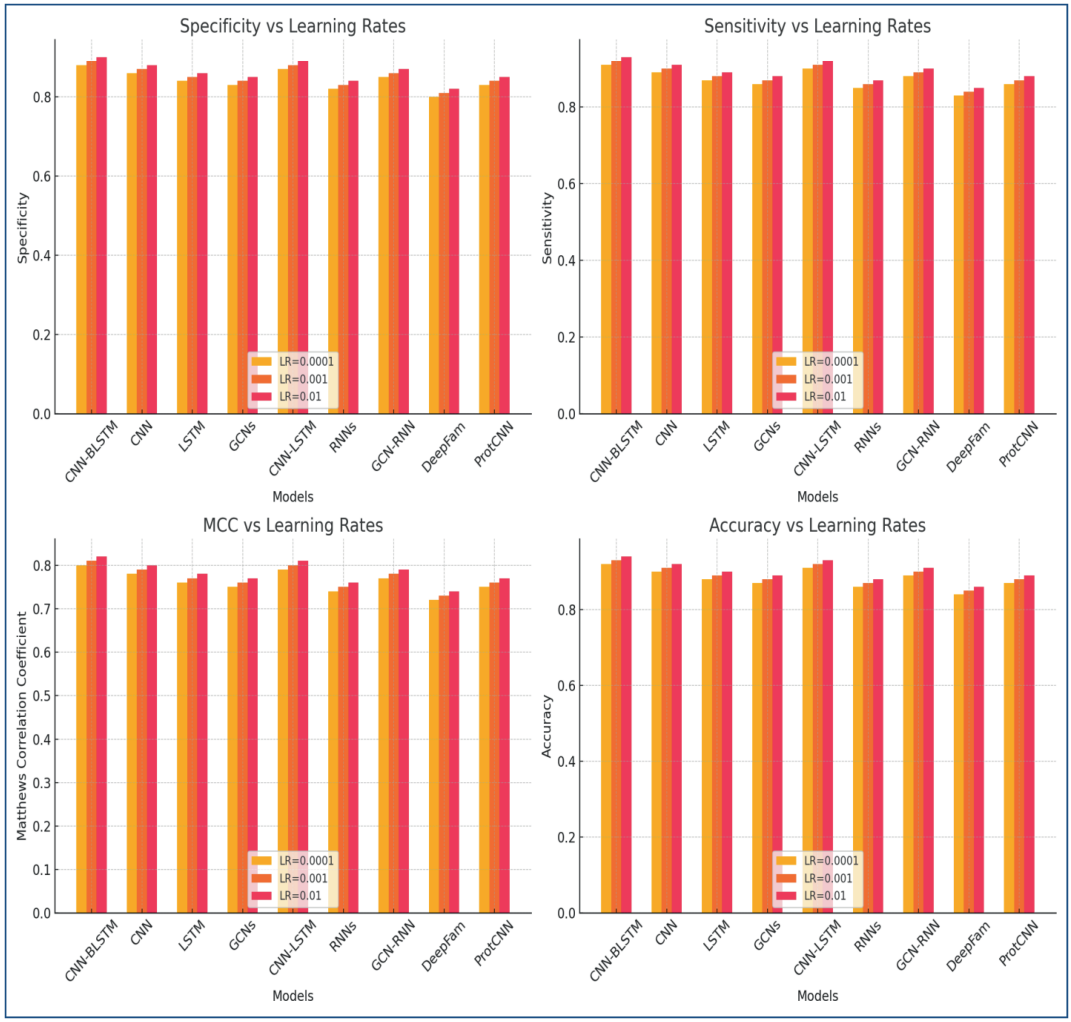


FIGURE 7 Simulation results on the test dataset for different learning rates for PDB-14189.

Different optimizers

The simulation results are calculated for different optimizers, that is, Adam, RMSprop, and SGD, on the PDB-2272 protein dataset for the proposed and existing state-of-the-art methods. Figure 12 presents the simulation results on the test dataset for different optimizers.

Different dropout rates

The simulation results are calculated for different learning rates, that is, 01, 0.2, 0.3, and 0.4, on the PDB-2272 dataset for the proposed and existing state-of-the-art methods. Figure 13 presents the simulation results on the test dataset for different dropout rates.

4.3.3 | Training and testing loss

The simulation results are calculated for training and test loss for different Epochs from 10 to 100 over the PDB-14189 and PDB-2272 datasets for the proposed and existing state-of-the-art

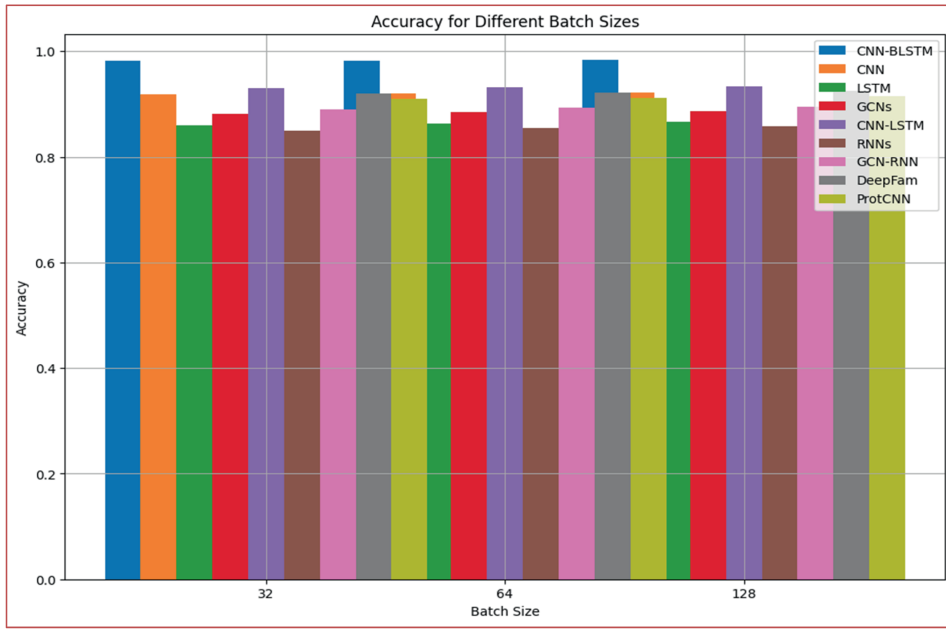


FIGURE 8 Simulation results on the test dataset for different batch sizes for PDB-14189.

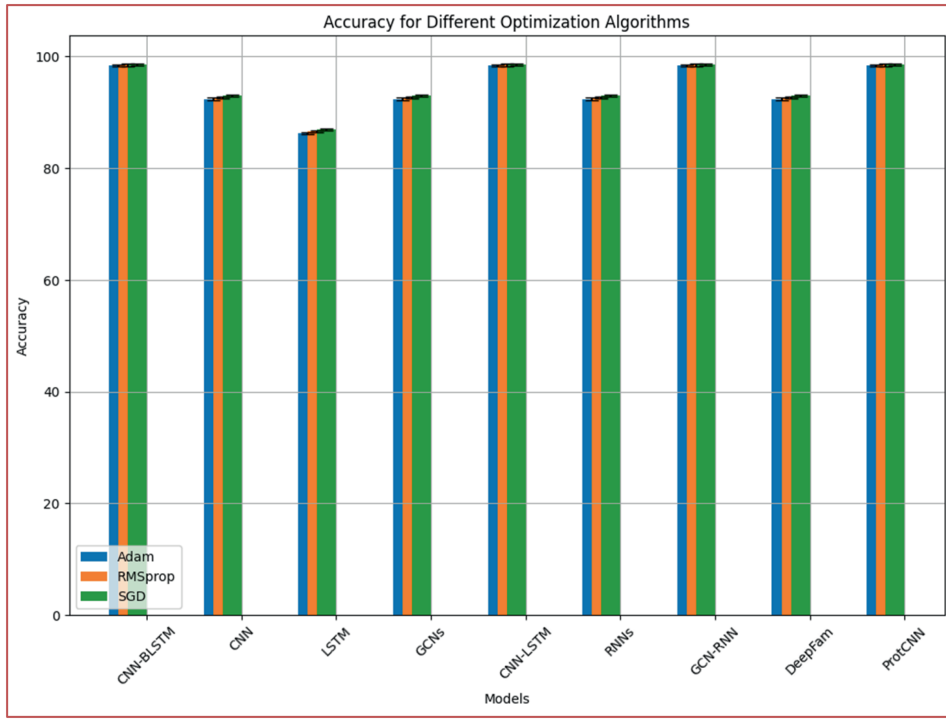


FIGURE 9 Simulation results on the test dataset for different optimizers for PDB-14189.

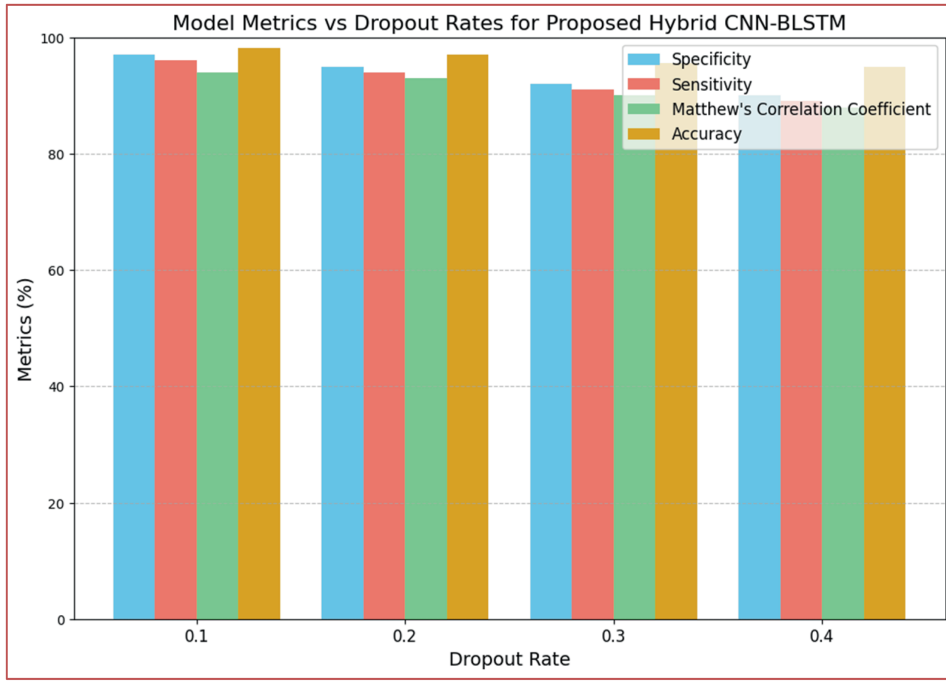


FIGURE 10 Simulation results on the test dataset for different dropout rates for PDB-14189.

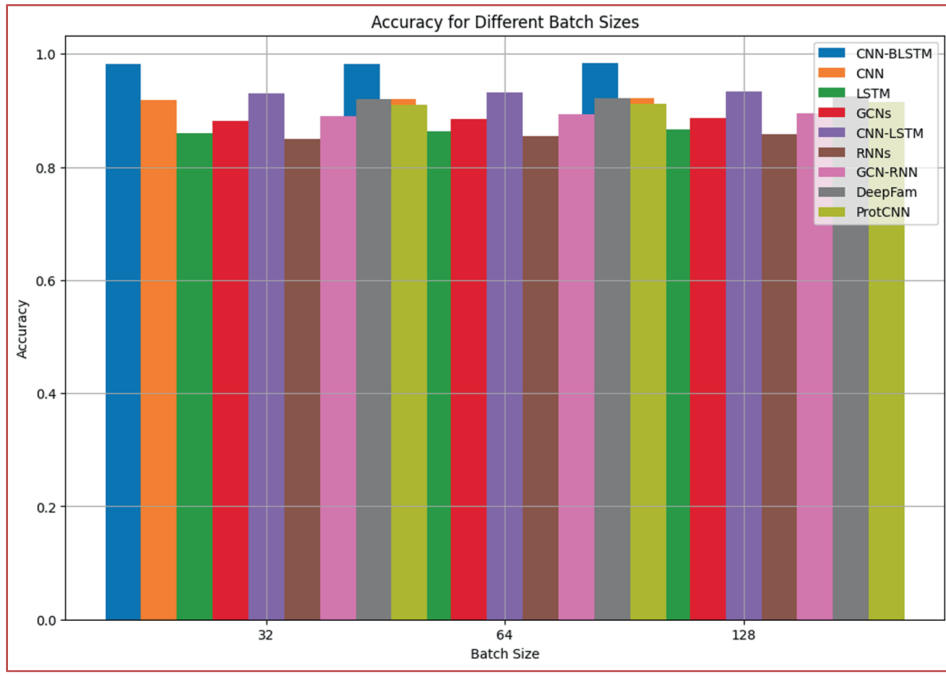


FIGURE 11 Simulation results on the test dataset for different batch sizes for PDB-2272.

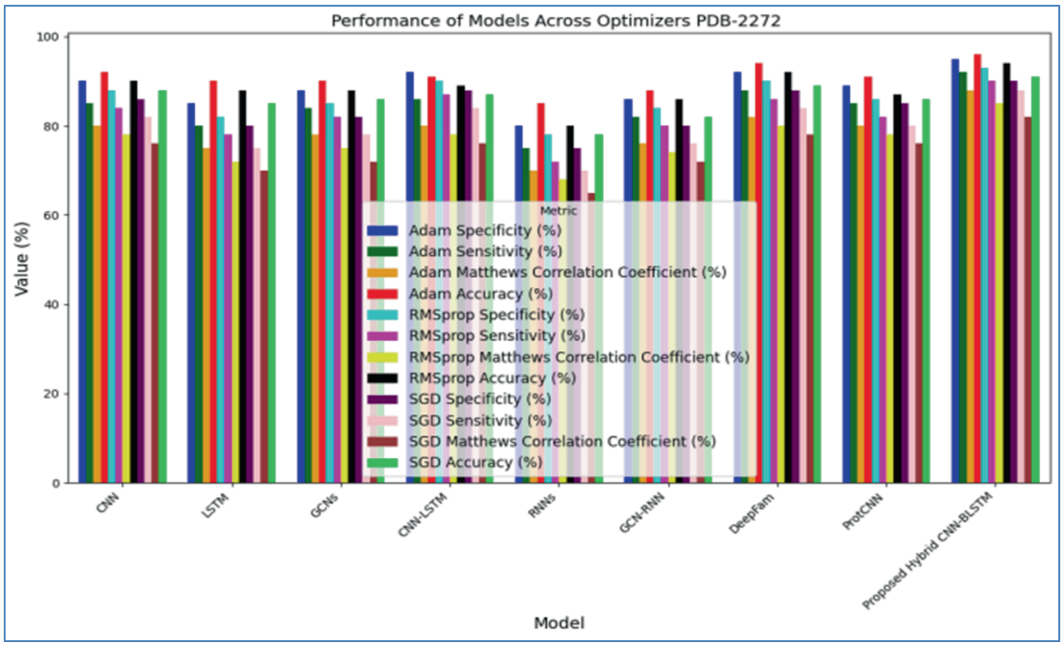


FIGURE 12 Simulation results on the test dataset for different learning rates for PDB-2272.

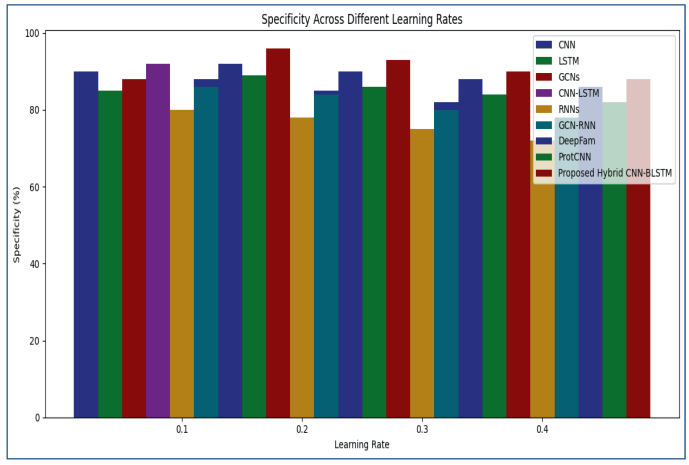
methods. Figures 14 and 15 present the simulation results on the training and testing loss at the number of Epochs for the PDB-14189 and PDB-2272 datasets.

Figure 14 shows PDB-14189 training and testing losses by epoch. The model is improving as the training loss decreases from 0.575 at 10 epochs to 0.271 at 100. Over the same epochs, the testing loss decreases from 0.585 to 0.281, indicating that the model can generalize to new data. The numerical values show the model's performance improving with training. Figure 15 shows PDB-2272 training and testing losses by epoch. The model's training loss decreases from 0.565 at 10 epochs to 0.241 at 100, indicating strong learning. Simultaneously, the testing loss steadily decreases throughout the epochs, going from 0.575 to 0.261. This indicates that the model can effectively apply what it has learned to unseen data. The numbers showcase the model's consistent progress in performance throughout the training phase.

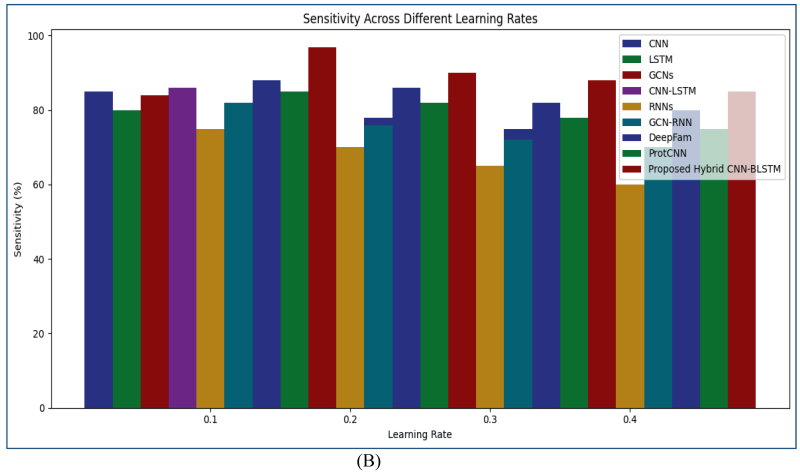
4.4 | Universality across different organisms

We have also utilized Prokaryotic Proteins[21] and Eukaryotic Proteins[23] datasets to measure the universality of the ProteinCNN-BLSTM model across different organisms.

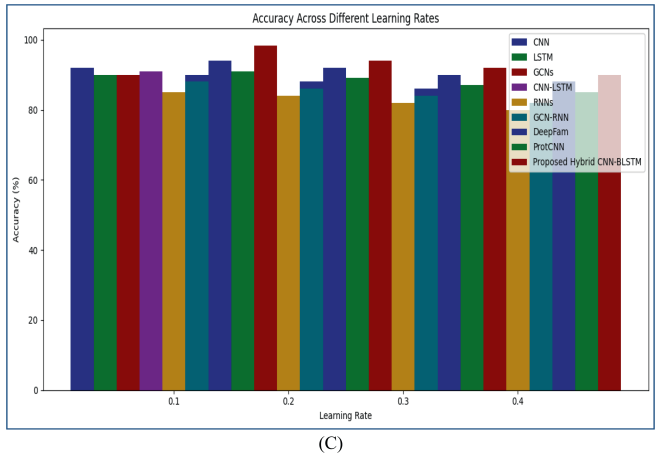
Tables 8 and 9 present experimental results for Prokaryotic Proteins and Eukaryotic Proteins across different organisms. In contrast to more traditional models such as CNN and LSTM, the ProteinCNN-BLSTM model consistently performed better during our evaluation of prokaryotic proteins (Table 8). For Escherichia coli, the ProteinCNN-BLSTM model yielded an accuracy of 94.42%, precision of 93.78%, recall of 94.85%, and F1-score of 94.91%. When comparing the models, it was found that CNN and LSTM models had lower performance, with accuracies of 89.76% and 87.54%, respectively. ProteinCNN-BLSTM demonstrated superior performance in all metrics for *Bacillus subtilis*, indicating similar trends. The ProteinCNN-BLSTM model also exhibited



(A)



(B)



(C)

FIGURE 13 (A) Specificity, (B) sensitivity, and (C) accuracy of the test dataset for different dropout rates for PDB-2272.

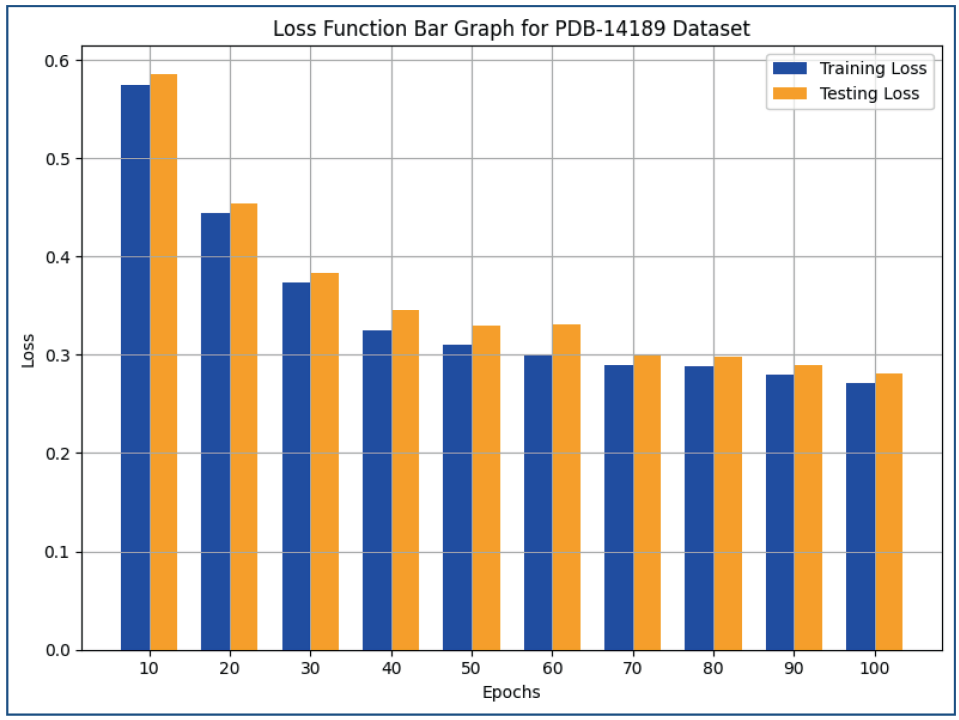


FIGURE 14 Training and testing loss versus number of epochs for PDB-14189.

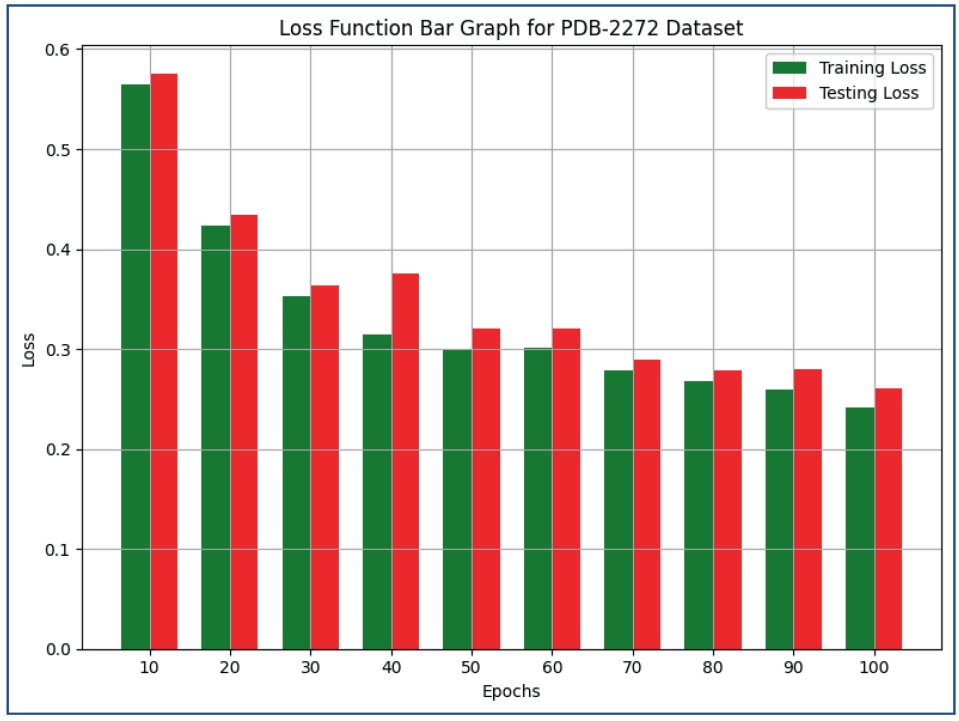


FIGURE 15 Training and testing loss versus number of epochs for PDB-2272.

TABLE 8 Experimental results for prokaryotic proteins across different organisms.

Organism	Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Escherichia coli	ProteinCNN-BLSTM	94.42	93.78	94.85	94.91
	CNN	89.76	89.21	89.89	89.75
	LSTM	87.54	86.58	87.06	87.72
Bacillus subtilis	ProteinCNN-BLSTM	93.35	93.01	93.87	93.84
	CNN	88.79	88.84	89.72	88.68
	LSTM	86.87	86.22	87.90	86.36

TABLE 9 Experimental results for eukaryotic proteins across different organisms.

Organism	Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Homo sapiens	ProteinCNN-BLSTM	95.78	95.85	96.70	95.37
	CNN	91.83	90.99	91.35	91.52
	LSTM	89.72	88.87	89.54	89.70
Saccharomyces cerevisiae	ProteinCNN-BLSTM	92.67	92.73	92.79	92.66
	CNN	88.54	88.80	88.86	88.73
	LSTM	86.45	86.50	86.37	86.83
Arabidopsis thaliana	ProteinCNN-BLSTM	91.39	91.45	92.72	91.98
	CNN	87.78	87.34	88.20	87.67
	LSTM	85.56	85.21	85.68	85.54

superior performance for eukaryotic proteins (as shown in Table 9). The model achieved an accuracy of 95.78%, precision of 95.85%, recall of 96.70%, and F1-score of 95.37% in *Homo sapiens*. By comparison, the CNN and LSTM models obtained accuracies of 91.83% and 89.72%, respectively, which were lower. Comparable enhancements in performance were noted for *Saccharomyces cerevisiae* and *Arabidopsis thaliana*.

The findings show that, in various organisms, the ProteinCNN-BLSTM model consistently performs better than conventional models. Our model effectively captures the various characteristics of protein sequences from both prokaryotic and eukaryotic sources, as evidenced by the high accuracy, precision, recall, and F1 score. This demonstrates how adaptable and reliable the model is when processing protein sequences with various biological starting points.

4.5 | Ablation study

Ablation analysis is performed to determine the individual contributions that each component of the hybrid CNN-BLSTM model with amino acid embedding makes to the model's overall performance. We can analyze the effects of these modifications on significant performance measures such as accuracy, precision, recall, and F1-score by following a methodical process that involves removing or modifying model components.

When one considers that the comprehensive hybrid model, comprised of CNN, BLSTM, and amino acid embedding, achieves superior performance across all metrics, it is possible to deduce that this combination offers the most robust capabilities for feature extraction and sequence classification. The elimination of amino acid embedding results in a significant decrease in performance, which highlights its significant role in improving feature representation. This case demonstrates that the embedding can successfully capture intricate patterns within protein sequences necessary for accurate classification.

The ablation analysis comprehensively investigates the influence of various configurations on the performance of the ProteinCNN-BLSTM model. According to Table 10, the full hybrid model, which incorporates CNN, BLSTM, and amino acid embedding, achieves the highest scores. It has an accuracy of 98.1%, precision of 96%, recall of 94%, and an F1-score of 95%.

In contrast, models without amino acid embedding or with only CNN or BLSTM components perform worse, with the CNN-only configuration reaching an accuracy of just 88.7%. Table 11 indicates that increasing the convolutional layers from 2 to 6 boosts accuracy from 93.3% to 96.1%, along with improved precision, recall, F1-score, and AUC-ROC.

Table 12 shows different mixes of convolutional and BLSTM layers. The Conv-BLSTM configuration performs best, with an accuracy of 88.1% and an AUC-ROC of 90%. Finally, Table 13 shows the advantages of using already trained models. ResNet significantly enhances performance, achieving an F1-score of 95.78% and an accuracy of 96.18%, surpassing the baseline model's accuracy of 90.12%. These findings demonstrate the significance of employing a meticulously organized hybrid model and utilizing pre-trained architectures for optimal performance.

4.6 | Results and discussion

Figure 6 presents the simulation results for different epochs vs. the performance measuring parameters for PDB-14189. The findings show significant differences in performance between

TABLE 10 Experimental results of ablation analysis based on model configuration.

Model Configuration	Accuracy	Precision	Recall	F1-Score
Full Hybrid Model (CNN + BLSTM + Amino Acid Embedding)	0.981%	0.96	0.94	0.95
CNN + BLSTM (without Amino Acid Embedding)	0.925%	0.94	0.91	0.93
CNN only	0.887%	0.89	0.86	0.88
CNN + Amino Acid Embedding (without BLSTM)	0.913%	0.92	0.89	0.91
BLSTM only	0.876%	0.88	0.85	0.87
BLSTM + Amino Acid Embedding (without CNN)	0.90.8%	0.91	0.88	0.90

TABLE 11 Experimental results of ablation analysis based on number of conv layers.

Number of Conv Layers	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	AUC-ROC (%)
2	93.3	92.35	93.25	92.39	92.35
4	95.6	95.31	95.37	94.89	94.57
6	96.1	96.32	96.71	96.75	95.31

TABLE 12 Experimental results of ablation analysis based on combination of convolutional and BLSTM layers.

Architecture combination	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	AUC-ROC (%)
Conv-BLSTM	88.1	87	86	86	90
BLSTM-Conv	87.4	86	85	85	89
Alternating Conv and BLSTM	88.3	87	86	86	90

TABLE 13 Experimental results of ablation analysis based on configuration with pre-trained models.

Pre-trained Model included	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	AUC-ROC (%)
None	90.12	90.32	90.27	90.05	91.65
VGG	94.37	95.32	94.65	94.08	93.23
ResNet	96.18	96.12	95.62	95.78	94.95

models and eras. For example, CNN-BLSTM, the proposed hybrid method, consistently outperforms the current methods in all metrics and epochs, with an average specificity of approximately 96.5%, sensitivity of approximately 91.5%, MCC of approximately 0.89, and accuracy of approximately 95.8%. On the other hand, the performance of the current methods varies, and CNN-LSTM performs the closest to the proposed hybrid model, with a specificity of approximately 94%, a sensitivity of 90%, an MCC of approximately 0.86, and an accuracy of approximately 92.9%. Higher specificity and accuracy values indicate the superiority of the proposed hybrid CNN-BLSTM method, indicating its improved ability to correctly identify negative instances and overall classification accuracy, which are critical for dependable model performance.

Its consistently higher MCC also indicates more robust predictions across classes, suggesting an improved overall balance among true positives, true negatives, false positives, and false negatives compared to other techniques. Figure 7 presents the simulation results on the test dataset for different learning rates for PDB-14189. In particular, when trained with a learning rate of 0.0001, the proposed hybrid CNN-BLSTM model achieved a specificity of 97%, a sensitivity of 96%, an MCC of 0.94%, and an accuracy of 98.11%. This model performs better on all the metrics. Its hybrid architecture, which combines the advantages of BLSTM networks for temporal dependencies and CNNs for spatial feature extraction, is responsible for this improved performance. Better generalization and faster convergence are also made possible by the ideal learning rate, which enables the model to extract rich and discriminative representations from the data and increase classification accuracy.

Figure 8 shows how the proposed CNN-BLSTM hybrid method performs in terms of accuracy compared to other methods (CNN, LSTM, GCNs, CNN-LSTM, RNNs, GCN-RNN, DeepFam, and ProtCNN) on the PDB-14189 dataset at different batch sizes (32, 64, and 128). Based on the results, it can be concluded that for all batch sizes, the CNN-BLSTM model consistently outperforms the other approaches regarding accuracy. For example, the CNN-BLSTM outperforms the CNN, which achieves approximately 92.12% accuracy at a batch size of 128 with an accuracy of approximately 98.35%. Larger batch sizes in training are advantageous, as evidenced by the general trend of improved accuracy for most models as the batch size increases. Interestingly, the CNN-BLSTM model shows the greatest increase in accuracy as the batch size increases, confirming its efficacy in using larger batches to improve performance. Overall, the graph shows that the

proposed hybrid method outperforms the current methods in terms of accuracy on the PDB-14189 dataset, especially when trained with larger batch sizes.

Figure 9 shows the accuracy of several models trained using different optimization algorithms, namely, Adam, RMSprop, and SGD, based on simulation results on the PDB-14189 test dataset. The accuracy attained by models such as CNN-BLSTM, CNN, LSTM, GCNs, CNN-LSTM, RNNs, GCN-RNN, DeepFam, and ProtCNN is represented by each bar. The error bars show how accuracy measurements can vary. With accuracies ranging from 98.35% to 98.59% across optimization algorithms, the CNN-BLSTM routinely outperforms other models. For example, the CNN-BLSTM algorithm achieves 98.35% accuracy under the Adam optimizer, whereas the CNN, LSTM, and GCN algorithms achieve 98.47%, 98.59%, and 92.38%, respectively. The reason for the superiority of CNN-BLSTM is its hybrid architecture, which uses LSTMs and CNNs to capture complex patterns in protein data. Furthermore, the CNN-BLSTM performs better than the other models because of Adam's adaptive learning rates.

Figure 10 shows the accuracy of different models trained with different dropout rates (0.1, 0.2, 0.3, and 0.4) based on simulation results on the test dataset for the PDB-14189 protein. Error bars show the variation in accuracy measurements, and each bar shows the mean accuracy attained by models under a particular dropout rate. Notably, as the dropout rate increases, the accuracy tends to decrease. For example, the accuracy varies between 97.46% and 98.25% across models under a dropout rate of 0.1 and between 97.36% and 98.15% under a dropout rate 0.4. This pattern implies that accuracy increases with decreasing dropout rates, underscoring the significance of carefully choosing dropout rates to maximize model performance.

The simulation results for various batch sizes on the test dataset over the PDB-2272 dataset are shown in Figure 11. Concerning different batch sizes, each bar represents a distinct performance metric (specificity, sensitivity, MCC, and accuracy) for a variety of models, including CNN-BLSTM, CNN, LSTM, GCNs, CNN-LSTM, RNNs, GCN-RNN, DeepFam, and ProtCNN. We observed that while the MCC and accuracy tended to increase initially before plateauing, the specificity and sensitivity decreased slightly as the batch size increased. The CNN-BLSTM hybrid method is noteworthy for its consistently superior performance over other models in varying batch sizes. It exhibited higher values for specificity, sensitivity, MCC, and accuracy. This advantage is due to the hybrid architecture's capacity to efficiently capture temporal and spatial features, enabling more thorough protein sequence analysis and classification. Furthermore, using BLSTM layers with CNNs may improve the model's classification performance over other conventional models by improving its ability to capture subtle patterns and long-range dependencies in protein sequences.

Figure 12 shows PDB-2272 test dataset simulation results for various learning rates. This figure shows accuracy, sensitivity, specificity, Matthews' correlation coefficient, and deep learning model performance across optimizers. Each model is shown by bars with different metric colors to make comparisons easier. With the Adam optimizer, the CNN-LSTM model has 86% sensitivity, and the GCNs model 85% with RMSprop. The hybrid CNN-BLSTM outperforms other models in every metric and optimizer tested with 95% accuracy, 92% sensitivity, 88% Matthews' correlation coefficient, and 95% specificity. This improved performance shows the value of the hybrid strategy, which likely combines the best of the CNN and LSTM architectures to boost predictive power and model efficacy. The hybrid CNN-BLSTM model outperforms other models, suggesting it could produce better classification results. This offers promising real-world applications in various fields.

Figure 13 shows simulation results for various dropout rates on the PDB-2272 test dataset. The bars show how a model performed at different dropout rates, focusing on specificity,

sensitivity, and accuracy. The analysis shows that CNN-BLSTM outperforms current models at all dropout rates. At 0.3 dropout, the CNN-BLSTM has 90% specificity, 88% sensitivity, and 92% accuracy. CNNs, LSTMs, and GCNs have been used with specificities from 85% to 88%, sensitivities from 80% to 85%, and accuracies from 86% to 90% at the same dropout rate. The suggested model performs better because it uses complementary features of CNN and LSTM architectures. LSTM layers for sequence modeling and convolutional layers for feature extraction help the proposed model extract complex patterns from data, improving predictive performance. The CNN-BLSTM model had the highest specificity, sensitivity, and accuracy at various dropout rates. The CNN-BLSTM model also performs well at various dropout rates, demonstrating its dropout regularization resilience. When other models perform poorly with higher dropout rates, the CNN-BLSTM model remains stable and reliable.

These results show that the hybrid CNN-BLSTM model classified protein sequences in the PDB-2272 dataset well, making it suitable for bioinformatics and computational biology applications. Experimental ablation analysis results are in Table 10. CNNs extract spatial features, and BLSTMs learn sequences best when combined, as the CNN-only and BLSTM-only models perform poorly. We found that CNNs' spatial feature extraction, BLSTMs' sequence learning, and amino acid embedding-enhanced feature representation perform well together. The hybrid model classifies protein sequences better than current methods, enabling research for bioinformatics and protein sequence analysis.

5 | CONCLUSION AND FUTURE DIRECTION

Protein sequence classification has advanced significantly with the creation and validation of ProteinCNN-BLSTM, which provides a strong combination of deep learning techniques adapted to the complex structure of biological data. Our model's outstanding accuracy highlights its potential as a potent tool for bioinformatics research and real-world applications, as evidenced by its consistent outperformance of existing techniques on standard protein datasets. After careful analysis, we obtained strong statistical findings for the PDB-2272 and PDB-14189 datasets. ProteinCNN-BLSTM on PDB-14189 produced an average specificity of approximately 96.5%, a sensitivity of approximately 91.5%, an approximately 0.89 Matthews correlation coefficient (MCC), and an overall accuracy of approximately 98.18%. Interestingly, our model performed even better when trained with a learning rate of 0.0001, with a specificity reaching 97%, a sensitivity reaching 96%, an MCC reaching 0.94, and an accuracy peaking at 98.11%.

Similarly, ProteinCNN-BLSTM showed better classification performance on the PDB-2272 dataset, with over 98.15% accuracy, 92% sensitivity, 88% specificity, and over 0.88 MCC across a range of optimization algorithms and dropout rates. These numerical results prove the model's ability to correctly classify protein sequences from various datasets. Ahead, several directions demand further investigation and improvement. Subsequent investigations may concentrate on enhancing the computational effectiveness of the ProteinCNN-BLSTM model to better manage larger datasets and environments with limited resources. Furthermore, broadening the scope of the analysis to include a larger variety of protein sequences, including those from various biological contexts—would improve our comprehension of the model's cross-domain generalizability and applicability. However, it is important to recognize the limitations of our methodology. One major challenge is computational complexity, which requires continuously developing hardware solutions and simplified algorithms for effective training and deployment. Furthermore, even though our model performs admirably, there are still questions about how interpretable deep

learning models can be in biological contexts. To overcome this obstacle, new model interpretation and visualization methods must be investigated to clarify the biological mechanisms that ProteinCNN-BLSTM captures.

In the final analysis, although ProteinCNN-BLSTM is a promising development for protein sequence classification, more research is required to fully determine its potential and overcome obstacles. We can advance computational biology and create novel approaches to challenging biological issues by tackling these constraints and utilizing cutting-edge technologies.

AFFILIATIONS

[1]Department of Computer Science and Engineering, Galgotias University, Greater Noida, India

[2]Department of Computer Science and Engineering, Amity University Haryana, Gurugram, India

[3]Department of Computer Engineering and Applications, GLA University, Mathura, India

[4]Department of Computer Science & Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, India

[5]Department of Computer Science and Engineering, Shri Vishnu Engineering College for Women (A), Bhimavaram, India

[6]Department Electrical Engineering, Samrat Ashok Technological Institute, Vidisha, India

[7]Department of Mechatronics, Faculty of Engineering, Ain Shams University, Cairo, Egypt

[8]Department of Industrial Engineering, College of Engineering, King Saud University, Riyadh, Saudi Arabia

AUTHOR CONTRIBUTIONS

Umesh Kumar Lilhore and V. V. R. Maheswara Rao were responsible for the validation, software, data curation, and writing of the original draft. Surjeet Dalal and Sarita Simaiya were responsible for conceptualizing and writing the original draft. Neetu Faujdar and Yogesh Kumar Sharma were responsible for writing – the original draft and visualization. K. B. V. Brahma Rao and Shilpi Tomar wrote, reviewed, and edited the manuscript. Yogesh Kumar Sharma and V. V. R. Maheswara Rao were responsible for the formal analysis. AM, Enab, and Mehdi wrote the original draft and the resources and supervision. The author(s) read and approved the final manuscript.

ACKNOWLEDGMENTS

The authors extend their appreciation to King Saud University for funding this work through the Researchers Supporting Project number (RSPD2024R685), King Saud University, Riyadh, Saudi Arabia.

CONFLICT OF INTEREST STATEMENT

The authors declare no competing interests.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Umesh Kumar Lilhore https://orcid.org/0000-0001-6073-3773
Sarita Simaiya https://orcid.org/0000-0001-7686-8496
Surjeet Dalal https://orcid.org/0000-0002-4325-9237
Yogesh Kumar Sharma https://orcid.org/0000-0003-1934-4535

K. B. V. Brahma Rao https://orcid.org/0000-0002-5719-9810
V. V. R. Maheswara Rao https://orcid.org/0000-0002-0503-7211
Shilpi Tomar https://orcid.org/0000-0001-9828-4918
Ehab Ghith https://orcid.org/0000-0002-4338-9867
Mehdi Tlija https://orcid.org/0000-0003-2661-5102

REFERENCES

1. Zhou Y, Tan K, Shen X, He Z. A protein structure prediction approach leveraging transformer and CNN integration. arXiv preprint arXiv:2402.19095. 2024.
2. Zhu Y-H, Liu Z, Liu Y, Ji Z, Dong-Jun Y. ULDNA: integrating unsupervised multi-source language models with LSTM-attention network for high-accuracy protein–DNA binding site prediction. *Brief Bioinform*. 2024;25(2):bbae040.
3. Ahmed NY, Alsanousi WA, Hamid EM, et al. An efficient deep learning approach for DNA-binding proteins classification from primary sequences. *Int J Comput Intell Syst*. 2024;17(1):1-14.
4. Wang Y, Ding P, Wang C, He S, Gao X, Bin Y. RPI-GGCN: prediction of RNA–protein interaction based on interpretability gated graph convolution neural network and co-regularized variational autoencoders. *IEEE Trans Neural Netw Learn Syst*. 2024;35(8):1-15.
5. Kumar N, Choudhury S, Bajiya N, Patiyal S, Raghava GPS. Prediction of anti-freezing proteins from their evolutionary profile. *bioRxiv*. 2024;10:7.
6. Ali S, Sahoo B, Zelikovsky A, Chen P-Y, Patterson M. Benchmarking machine learning robustness in Covid-19 genome sequence classification. *Sci Rep*. 2023;13(1):4154.
7. Yeung W, Zhou Z, Mathew L, et al. Tree visualizations of protein sequence embedding space enable improved functional clustering of diverse protein superfamilies. *Brief Bioinform*. 2023;24(1):bbac619.
8. Motmaen A, Dauparas J, Baek M, Abedi MH, Baker D, Bradley P. Peptide-binding specificity prediction using fine-tuned protein structure prediction networks. *Proc Natl Acad Sci*. 2023;120(9):e2216697120.
9. Yao J, Ling Y, Hou P, Wang Z, Huang L. A graph neural network model for deciphering the biological mechanisms of plant electrical signal classification. *Appl Soft Comput*. 2023;137:110153.
10. Goto K, Tamehiro N, Yoshida T, et al. Novel machine learning method allerStat identifies statistically significant allergen-specific patterns in protein sequences. *J Biol Chem*. 2023;299(6):101-129.
11. Ho Z, Yang Y, Ma Z, Wong K-c, Li X. Learning the protein language of proteome-wide protein–protein binding sites via explainable ensemble deep learning. *Commun Biol*. 2023;6(1):73.
12. Wang E, Wang F, Yang Z, et al. A graph convolutional network–based method for chemical-protein interaction extraction: algorithm development. *JMIR Med Inform*. 2020;8(5):e17643.
13. Lilhore UK, Imoize AL, Lee C-C, et al. Enhanced convolutional neural network model for cassava leaf disease identification and classification. *Mathematics*. 2022;10(4):580.
14. Llinares-López F, Berthet Q, Blondel M, Teboul O, Vert J-P. Deep embedding and alignment of protein sequences. *Nat Methods*. 2023;20(1):104-111.
15. Onyema EM, Lilhore UK, Saurabh P, et al. Evaluation of IoT-enabled hybrid model for genome sequence analysis of patients in healthcare 4.0. *Measurement: Sensors*. 2023;26:100679.
16. Lilhore UK, Simaiya S, Dalal S, Damaševičius R. A smart waste classification model using hybrid CNN-LSTM with transfer learning for sustainable environment. *Multimedia Tools Applicat*. 2024;83(10):29505-29529.
17. Pattnaik D, Thakur SB, Dash PM, Jena S, Sahu S, Behera SK. Molecular medical diagnosis of COVID-19 and omicron variant. *J Pharmaceut Neg Results*. 2022;18:6332-6347.
18. Singh D, Roy J. A large-scale benchmark study of tools for the classification of protein-coding and noncoding RNAs. *Nucleic Acids Res*. 2022;50(21):12094-12111.
19. Hashem A, Motalib Hossain MA, Marlinda AR, et al. Nucleic acid-based electrochemical biosensors for rapid clinical diagnosis: advances, challenges, and opportunities. *Crit Rev Clin Lab Sci*. 2022;59(3):156-177.
20. Le NQ, Khanh EK, Yapp Y, Yu-Yen O, Yeh H-Y. iMotor-CNN: identifying molecular functions of cytoskeleton motor proteins using 2D convolutional neural network via Chou's 5-step rule. *Anal Biochem*. 2019;575:17-26.
21. Ben-Bassat I, Chor B, Orenstein Y. A deep neural network approach for learning intrinsic protein–RNA binding preferences. *Bioinformatics*. 2018;34(17):i638-i646.

22. Zhang Z, Park CY, Theesfeld CL, Troyanskaya OG. An automated framework for efficiently designing deep convolutional neural networks in genomics. *Nat Mach Intell*. 2021;3(5):392-400.
23. Lilhore UK, Dalal S, Faujdar N, et al. Hybrid CNN-LSTM model with efficient hyperparameter tuning for prediction of Parkinson's disease. *Sci Rep*. 2023;13(1):14605.
24. Zhao T, Yang H, Valsdottir LR, Zang T, Peng J. Identifying drug–target interactions based on graph convolutional network and deep neural network. *Brief Bioinform*. 2021;22(2):2141-2150.
25. Le NQ, Khanh Q-TH, Nguyen T-T-D, Yu-Yen O. A transformer architecture based on BERT and 2D convolutional neural network to identify DNA enhancers from sequence information. *Brief Bioinform*. 2021;22(5):bbab005.
26. Niu M, Lin Y, Zou Q. sgRNACNN: identifying sgRNA on-target activity in four crops using ensembles of convolutional neural networks. *Plant Mol Biol*. 2021;105:483-495.
27. Yuvaraj N, Srihari K, Chandragandhi S, Raja RA, Dhiman G, Kaur A. Analysis of protein-ligand interactions of SARS-CoV-2 against selective drug using deep neural networks. *Big Data Min Anal*. 2021;4(2):76-83.
28. Park S, Seok C. GalaxyWater-CNN: prediction of water positions on the protein structure by a 3D-convolutional neural network. *J Chem Informat Model*. 2022;62(13):3157-3168.
29. Lv Z, Ding H, Wang L, Zou Q. A convolutional neural network using dinucleotide one-hot encoder for identifying DNA N6-methyladenine sites in the rice genome. *Neurocomputing*. 2021;422:214-221.
30. Dalal S, Lilhore UK, Radulescu M, Simaiya S, Jaglan V, Sharma A. A hybrid LBP-CNN with YOLO-v5-based fire and smoke detection model in various environmental conditions for environmental sustainability in smart city. *Environ Sci Pollut Res*. 2024;37:1-18.
31. Zhang D, Kabuka MR. Protein family classification from scratch: a CNN based deep learning approach. *IEEE/ACM Trans Comput Biol Bioinformat*. 2020;18(5):1996-2007.
32. Mitra S, Saha S, Hasanuzzaman M. A multiview deep neural network model for chemical-disease relation extraction from imbalanced datasets. *IEEE J Biomed Health Informat*. 2020;24(11):3315-3325.
33. Cheng J, Liu Y, Ma Y. Protein secondary structure prediction based on integration of CNN and LSTM model. *J Visual Commun Image Represent*. 2020;71:102844.
34. Deng L, Liu Y, Shi Y, Zhang W, Yang C, Liu H. Deep neural networks for inferring binding sites of RNA-binding proteins by using distributed representations of RNA primary sequence and secondary structure. *BMC Genom*. 2020;21(13):1-10.
35. Tavakoli N. Seq2image: sequence analysis using visualization and deep convolutional neural network. *In 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE; 2020:1332-1337.
36. Pang L, Wang J, Zhao L, Wang C, Zhan H. A novel protein subcellular localization method with CNN-XGBoost model for Alzheimer's disease. *Frontiers in Genetics*. 2019;9:751.
37. Pu L, Govindaraj RG, Lemoine JM, Hsiao-Chun W, Brylinski M. DeepDrug3D: classification of ligand-binding pockets in proteins with a convolutional neural network. *PLoS Comput Biol*. 2019;15(2):e1006718.
38. Wang L, Wang H-F, Liu S-R, Yan X, Song K-J. Predicting protein–protein interactions from matrix-based protein sequence using convolution neural network and feature-selective rotation forest. *Sci Rep*. 2019;9(1):9848.
39. Rifaioglu S, Ahmet TD, Martin MJ, Cetin-Atalay R, Atalay V. DEEPred: automated protein function prediction with multitask feed-forward deep neural networks. *Sci Rep*. 2019;9(1):1-16.
40. Taju SW, Nguyen T-T-D, Hoang DX. DeePromoter: robust promoter predictor using deep learning with DNA sequence features. *Sci Rep*. 2019;9(1):1-11.
41. Han Y, Ming L, Zhang S, Shen W. Graph enhanced convolutional neural networks for protein binding sites prediction. *BMC Bioinform*. 2019;20:1-11.

How to cite this article: Lilhore UK, Simaiya S, Dalal S, et al. ProtienCNN-BLSTM: An efficient deep neural network with amino acid embedding-based model of protein sequence classification and biological analysis. *Computational Intelligence*. 2024;40(4):e12696. doi: 10.1111/coin.12696