




Sequence representation approaches for sequence-based protein prediction tasks that use deep learning

Feifei Cui , Zilong Zhang  and Quan Zou 

Corresponding author: Quan Zou, Institute of Fundamental and Frontier Sciences, University of Electronic Science and Technology of China, Chengdu, China, Yangtze Delta Region Institute (Quzhou), University of Electronic Science and Technology of China, Quzhou, Zhejiang, China, and Hainan Key Laboratory for Computational Science and Application, Hainan Normal University, Haikou, China. Tel.: +86 170-9226-1008; E-mail: zouquan@nclab.net

Abstract

Deep learning has been increasingly used in bioinformatics, especially in sequence-based protein prediction tasks, as large amounts of biological data are available and deep learning techniques have been developed rapidly in recent years. For sequence-based protein prediction tasks, the selection of a suitable model architecture is essential, whereas sequence data representation is a major factor in controlling model performance. Here, we summarized all the main approaches that are used to represent protein sequence data (amino acid sequence encoding or embedding), which include end-to-end embedding methods, non-contextual embedding methods and embedding methods that use transfer learning and others that are applied for some specific tasks (such as protein sequence embedding based on extracted features for protein structure predictions and graph convolutional network-based embedding for drug discovery tasks). We have also reviewed the architectures of various types of embedding models theoretically and the development of these types of sequence embedding approaches to facilitate researchers and users in selecting the model that best suits their requirements.

Key words: sequence representation; deep learning; protein sequence embedding; end-to-end learning; transfer learning

Introduction

Machine learning techniques for extracting knowledge from data have been widely used in bioinformatics [1–8]. Traditional machine learning models, which include support vector machine (SVM), random forest (RF) and Bayesian networks, have been initially applied, for example, to predict gene expression and gene function from sequences; to identify DNA motifs such as transcription factor binding sites; and to predict protein–protein interactions, secondary structures, protein–nucleic interactions and protein functions [9–26]. In these models, feature extraction should be preliminarily conducted, usually through other applications, and selected features

are used as input to machine learning programs. In protein tasks, these features include protein amino acid composition information, physicochemical properties of amino acids and protein evolutionary information features, and they are used as feature vectors to encode proteins [27, 28]. For the traditional machine learning approach of using extracted features, preprocessing is required for learning based on statistical measures to assign a score to each feature, and the features are ranked according to their scores and are either kept to form feature vectors that represent protein sequences or removed. The feature extraction methods are typically univariate and consider features independently. In this approach, if feature

Feifei Cui received her PhD degree from the University of Tokyo, Japan. She is currently a postdoctoral researcher at the University of Electronic Science and Technology of China. Her research interests include bioinformatics, deep learning and biological data mining.

Zilong Zhang is currently working as a postdoctoral researcher in the University of Electronic Science and Technology of China. He received his PhD degree from the University of Tokyo, Japan in 2020. His research interests include single-cell sequencing data analysis, bioinformatics and machine learning.

Quan Zou is a professor at the Institute of Fundamental and Frontier Sciences, University of Electronic Science and Technology of China. He received his PhD from Harbin Institute of Technology, PR China in 2009. He is a senior member of IEEE and ACM. His research is in the areas of bioinformatics, machine learning and parallel computing.

© The Author(s) 2021. Published by Oxford University Press. All rights reserved. For Permissions, please email: journals.permissions@oup.com

selection fails, the machine learning algorithm will not be able to realize satisfactory performance. In addition, the extraction of some features, such as the position-specific scoring matrix (PSSM) [29], which is a type of protein evolutionary information feature, requires substantial computation time. The protein representation that use traditional machine learning based on feature extraction are computationally complex and has some limitations, therefore is typically suitable for small-scale data.

Deep learning, which is a flexible evolution of machine learning, has recently become increasingly popular in bioinformatics because it typically outperforms traditional approaches, especially with large training datasets [30–32]. In recent years, biological data have been available on a large scale; for instance, the number of available protein sequences has increased exponentially in the past decade (Supplementary Figure S1). In contrast to traditional machine learning techniques, deep learning approaches do not require third-party applications to obtain feature information for sequence numerical representation. Instead, they can learn rich data representations directly from raw input sequences.

Representation learning, also namely the amino acid encoding or embedding in which sequence data are converted to numerical vectors, influences predicting model performance [33, 34]; therefore, a wide variety of sequence embedding schemes, which are described in detail in Table 1, have been used in protein prediction tasks. For example, a commonly employed representation scheme, namely, one-hot encoding, has been applied for the prediction of protein function [35] and to protein classification prediction problems [36–39], such as the prediction of nucleic binding proteins and protein motifs and protein subcellular localization, and computer-aided drug development. The word2vec technique [40, 41], which is a widely used word-embedding method in natural language processing (NLP), accepts a large corpus of text as input and assigns a corresponding vector representation to each unique word in the corpus; it has recently received substantial attention in proteomics [42, 43] because words in similar contexts can be embedded in similar vector spaces and a large-scale biological data set can be regarded as a corpus. The doc2vec model [44], which is an extension of word2vec, can learn embedding vectors for entire sentences or documents and can be used to represent biological sequences (e.g. DNA, RNA and protein sequences). Previous studies suggest that the embeddings that are learned from many types of variants of word2vec and doc2vec models can be applied for prediction in universal binary classification problems, such as disordered protein identification and protein family classification, and for prediction in protein functional problems, for example, for the prediction of properties of related proteins [45]. More recently, a new language model for NLP, namely, the bidirectional encoder representation from transformers (BERT), was proposed by researchers from Google [46]. BERT attracted substantial attention upon publication due to its state-of-the-art performance on many natural language understanding tasks. Within bioinformatics, BERT for biomedical text mining, namely, BioBERT [47], significantly outperforms the previous state-of-the-art models. Therefore, we can foresee the application of BERT in other bioinformatics studies. Another type of sequence representation learning method is a new statistical representation that uses deep learning for protein sequences, which was proposed by Alley. It can realize a unified representation for protein engineering because the learned representation of proteins is semantically rich and structurally, evolutionary and biophysically grounded [48].

Nevertheless, to the best of our knowledge, in bioinformatics, the protein sequence representation learning approaches have yet to be comprehensively classified. Therefore, here, we categorized sequence representation approaches that are used in deep learning applications of protein tasks into four groups: end-to-end learning-based sequence representation approaches, non-contextual sequence embedding methods, transfer learning-based sequence representation approaches and other representation approaches for some specific tasks (such as extracted features-based representations for protein structure prediction [49, 50] or graph computation-based representations for drug discovery tasks [51–53]), as presented in Table 1. Moreover, we review the development of the sequence representation technology and the architectures of types of embedding models theoretically, and we compare the sequence representation approaches of each group, for the purpose of facilitating researchers and users in selecting the model that best suits their requirement.

Overview of protein sequence representation approaches

Deep learning applications in proteomics, such as predicting protein–protein interactions or secondary structures from protein primary sequences, consist of three main parts: representation learning for amino acid sequences (amino acid encoding or embedding), deep neural network architecture and prediction outputs. Among these parts, the representation of sequence data is a key step for machine learning-based protein predictions [54, 55]. Four categories, namely, one-hot encoding-based simple embedding mechanism as the representative for end-to-end learning approaches (sequence representation in Figure 1a), non-contextual embedding-based and transfer learning-based representation approaches (both of them are the pretraining sequence representation type as described in Figure 1b), and other representation approaches that base on extracted features or graph computation for some specific tasks, are formed according to the learning mechanism of a various types of protein sequence representation models. The thing needs to be explained is that the sequence-based protein prediction tasks are similar to the problems of NLP, thus the representation learning approaches especially the representation approaches of the first three groups have the benefit of the development of the language model. Basically, except representation approaches of the fourth group, from the emergence of end-to-end learning-based representation methods to non-contextual embedding, and to transfer learning-based representation, the protein representing techniques can be described over the time.

In most traditional machine learning models for protein tasks that has struggled for decades since 2000s, many types of features are extracted by independent modules (in protein tasks, features are manually extracted, such as protein sequence attributes and physicochemical properties of amino acids) and as input for the task model. The training of independent feature extraction modules is conducted separately from the training of the task model, and the result of each module substantially affects the final task results. To address this problem, end-to-end learning, which is a popular type of deep learning process in NLP, is introduced for protein task modeling with the explosion of the number of available protein sequences since the early 2010s. Then, to avoid the sensitivity to the training dataset that is observed in the task architecture that uses end-to-end scheme,

Table 1. Description and samples of a wide variety of sequence representation schemes

Representation types	Research topics (tasks)	Sequence representation implement	Embedded dimension	Task model architecture	Ref.
End-to-end learning (One-hot encoding -based simple embedding)	DBP	One-hot vector of amino acids;	128	CNN_LSTM	[36]
		Embedding layer			
	Protein function and classification	One-hot vector of amino acids;	1,2,4,8,16,32	CNN, LSTM, CNN_LSTM	[30]
		Embedding layer			
	Protein function	One-hot vector of amino acids;	128	CNN	[35]
Non-contextual embedding (shallow architecture)	Word2vec	trigrams;			
		Embedding layer			
	Protein family classification, SS, disordered protein identification	One-hot vector of amino acids;	128	CNN_LSTM	[37]
		Convolutional layer			
	Drug discovery: protein-compound interaction	n-grams; Skip-gram	100	SVM	[42]
LSTM-based representation (ELMo and derived model)	Doc2vec	n-grams; Skip-gram	600	CNN	[43]
	Protein functional properties: absorption, enantioselectivity, localization	n-grams; Skip-gram	64	Gaussian process model	[45]
	LSTM-based representation (ELMo and derived model)	CharCNN; BiLSTM	3076	CNN	[61]
		Uncontextualized embedding; BiLSTM	3068	CNN_BiLSTM	[66]

(Continued)

Table 1. Continued

Representation types	Research topics (tasks)	Sequence representation implement	Embedded dimension	Task model architecture	Reference
Transfer Learning	Protein function: Stability of designed proteins, functional effects of mutations, phenotype of distant functional variants	One-hot vector of amino acids; mLSTM	1024	mLSTM	[48]
	Transformer-based representationd (BERT and derived model)	WordPiece tokenization; Multi-layer bidirectional transformer encoder	768	Multi-layer bidirectional transformer encoder	[47]
	Structure, evolutionary understanding, protein engineering (SS, contact prediction, remote homology detection, fluorescence landscape prediction, stability landscape prediction)	Token embedding, position embedding segment embedding; Multi-layer bidirectional transformer encoder	768	Multi-layer bidirectional transformer encoder	[66]
Other representations	Extracted features	Protein features: seven representative physio-chemical properties, 20D PSSM, 30D Markov model sequence profiles	57	BRNN (BiLSTM)-based iterative learning	[49]
		Distances between residue pairs		DNN	[50]
	Graph representation (GCN-based)	Normalized node features (sequence-based features and features computed from structure) and edge feature (the distance between two residues); multiple layers of graph convolution		DCNN	[51]
Other representations	Drug discovery: Prediction of interfaces between protein pairs	Node information (drug nodes and protein nodes) and edge information (protein-protein, drug-drug and drug-protein edges); two-layer multimodal graph			
	Polypharmacy: PPI, drug-protein target interaction, drug-drug interaction	Node embedding: d ; edge embedding: $d \times d$ ($d = 32, 64$)		Non-linear, multi-layer GCN	[52]
	Drug-target interactions identification	Nodes of DPP features: 51 or 53 Edge embedding matrix: $T \times T$ $T = (N_{drug} \times N_{proteins})$		DNN	[53]

L is the length of the sequence, n is the number of amino acids to compose the gram, N_{drugs} and $N_{proteins}$ are the number of drugs and proteins, D refers to dimensional, e.g. 20D refers to 20-dimensional. Abbreviations: DBP, DNA-binding proteins; SS, secondary structure; ASA, accessible surface area; CN, contact number; HSE, half sphere exposure; MSA, multiple sequence alignment; PPI, protein-protein interaction; HLA-II, human leukocyte antigen class II; CNN, convolution neural network; LSTM, long short-term memory; CNN_LSTM, CNN and LSTM; BRNN, bidirectional recurrent neural networks; BiLSTM, bidirectional LSTM; mLSTM, multiplicative LSTM; SVM, support vector machine; ELMo, embeddings from language models; BERT, bidirectional encoder representations from transformers; GCN, graph convolutional network; DCNN, diffusion-convolutional neural network; DNN, deep neural network; DPP, drug-protein pairs.

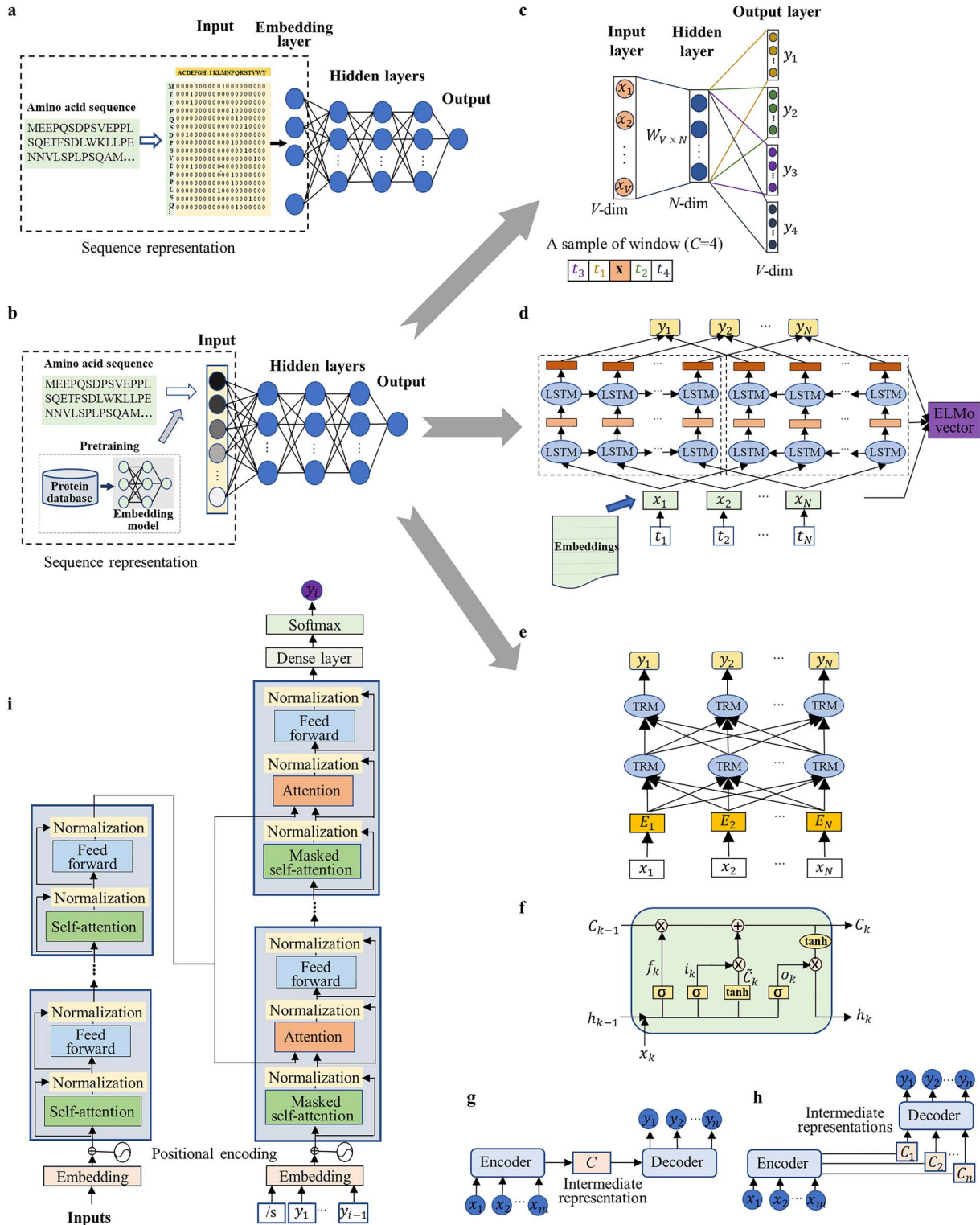


Figure 1. Architectures of different types of sequence representation schemes. (A) Task model architecture with end-to-end representation learning. Sequence representation learning, a one-hot encoding-based simple embedding scheme (the components in the dotted-line), is combined with the training of the subsequent network layers. (B) Task model architecture with pretraining-based sequence representation, including non-contextual embedding and transfer learning-based representation models. The pretraining embedding models include word2vec model (non-contextual embedding), LSTM-based and transformer-based models (transfer learning-based embedding), architectures of which are presented in C–E, respectively. (C) A training model of word2vec embedding scheme: skip-gram model (a sample of window size $C = 4$). The number of the surrounding words of the target word is C , and the window size is $C + 1$, here C equals to four. (D) An architecture of LSTM-based representation scheme: ELMo. The ELMo vector is the final sequence representation that is formed by embedded token representation (x_1, x_2, \dots, x_N) (colored in green), results of two ($L = 2$) hidden layers (rectangles in light and dark red). The token representation is obtained by embeddings (text block in green) that are usually calculated by some context-independent embedding approaches. The oval in blue of each layer refers to LSTM cell, detailed architecture of which is

unsupervised pretraining-based representation learning for protein sequences has emerged in protein modeling since a range of unsupervised pretraining technologies for language model learning have shown remarkable performance on many NLP tasks. In addition, it has been shown that pretrained models can learn universal language representations on a large unlabelled corpus, which are beneficial for downstream tasks because the training of a new model from scratch can be avoided. Recent studies have shown that the pretrained models that have emerged in NLP can be applied for learning protein representations from massive amino acid sequence datasets. The pretrained learning approaches, such as word2vec embedding [40] as the representative for non-contextual embedding mechanism, and ELMo-based (Embeddings from Language Models, ELMo [56]) and BERT-based representation model [46] as sequence representation based on transfer learning that have just emerged in the past 2 years, have become popular one after another. The pretrained learning embedding models aim at learning a continuous representation for each word in a document, which is implemented by unsupervised learning models. The architecture of these learning models has advanced from shallow to deep due to complicated factors, such as the development of computational power, training skills and machine learning methods. Traditional non-contextual embedding utilizes word2vec models as representatively shallow, three-layer neural networks, such as skip grams (Figure 1c), whereas transfer learning-based representation model architectures, which include long short-term memory-based (LSTM-based) architectures (e.g. ELMo as presented in Figure 1d) and transformer-based architectures (e.g. BERT) model, as presented in Figure 1e), are deep neural networks.

To reveal the comparison of each type of representation learning approaches and to select a suitable model architecture for downstream tasks, the architecture of sequence representation learning and its working principle must be presented in detail. In the section below, we introduce the categories of amino acid sequence representation learning approaches and describe in detail the mainly architectures of each type of embedding model (architectures of different representation schemes are presented in Figure 1).

End-to-end representation learning for protein sequences

The term 'end-to-end' means that a machine learning model can directly convert input data into a prediction output bypassing the independent intermediate steps that are typically conducted in a traditional pipeline [57, 58]. Within the end-to-end learning process, between the input data and the final output (from end-to-end), the intermediate layers of the task architecture are trained as one large part that can be treated as a black box, which demonstrates that it emphasizes the entire sequence task rather than a part of the system.

To input primary protein sequences into the task model, the amino acid sequences should be encoded and embedded into a continuous numerical vector that can be computed by the

computer. A one-hot encoding-based simple embedding scheme (sequence representation in Figure 1a) is the originally widely used protein representation scheme in the end-to-end learning model of protein tasks. In simple terms, the process is as follows: The amino acids in the protein sequence are represented by one-hot vectors and embedded into a continuous vector that is fed to the next layers of the task model. The embedding learning of the protein sequence is combined with the training of the subsequent network layers.

One-hot encoding, which is a sparse encoding (colored in yellow in Figure 1a), represents an amino acid by a vector (a binary variable) of length 20 that contains a single one and 19 zeros, where 20 refers to the total number of common amino acids that are required for protein synthesis in all organisms. The process of one-hot encoding can be simply described as follows: Typically, each amino acid in a sequence is represented by an integer value that ranges from 0 to 19, and each integer is converted into a binary vector (called a 'one-hot vector').

Then, the binary variable is multiplied by a weight matrix (the parameters that are learned from the model training) to obtain a new numeric vector — the final embedded representation of an amino acid. It can be expressed as $v_i = Wx_i$, where x_i is a 20-dimensional one-hot vector of the i -th amino acid in the sequence; $W \in \mathcal{R}^{d \times 20}$ is the weight matrix, which can be updated by fitting a deep neural network; and v_i is the final embedded vector of the i -th amino acid in the protein sequence. d is the dimension of the embedded vector v_i and is the parameter that we should determine. Regarding the dimension of the embedded vector, a value between 100 and 128 may be a satisfactory choice based on previous experience. We refer to this simple process as regular embedding, and it is usually implemented by an embedding layer of Keras (a deep-learning application programming interface that is written in Python) or convolutional layers.

Previous studies have shown that the end-to-end learning model may be able to capture the physicochemical similarities and differences between amino acids, but the learned characteristic information of amino acids differs among tasks and model architectures [30]. Arguably, the task models that use end-to-end learning enable the capture of the characteristic information of amino acids that is related to the specified task, namely, the end-to-end learning models may learn meaningful numerical representations for protein sequences.

Two main observations are made regarding the end-to-end learning scheme: (1) End-to-end learning requires a large amount of labeled data for training, and for each task, the larger and higher quality the dataset, the better the prediction performance that can be realized. (2) The embedding dimensionality is a critical hyperparameter that should be determined according to the task architecture and should not be too small. In addition, there is a limitation when using end-to-end sequence representation scheme. The protein-related task with end-to-end learning is a type of supervised deep learning, a large set of labeled data is required for learning a valuable representation of amino acid sequences to the realize satisfactory prediction performance. However, the construction of large-scale labeled datasets is expensive

presented in (F). (E) An architecture of transformer-based representation model: BERT. Representations of inputs (rectangles in dark yellow) are obtained by summing three embedding vectors of inputs: token embeddings, segment embeddings and position embeddings. Ovals in blue refer to encoder blocks of transformers, and architecture of transformer is presented in I. (F) LSTM cell architecture at the k -th time step. (G) The classic encoder-decoder framework. (H) Framework of the encoder-decoder with attention mechanism. (I) Transformer architecture. Encoders (blocks in gray) are stacked one the left, while stacked decoders (blocks in gray) are on the right. A vector of positional encoding is added with the embedding vector of input words before the first encoder layer and the first decoder layer. The outputs of the final encoder layer are passed into each of decoders. Results of decoders are computed one by one by using all of results that are predicted at previous steps, as presented that y_i is generated through $/s, y_1, \dots$, and y_{i-1} , where $/s$ refers to the start of sequence. Abbreviations: LSTM, long short-term memory; ELMo, embeddings from language models; BERT, bidirectional encoder representations from transformers.

due to the high annotation costs; therefore, comparatively, not so many high-quality labeled data are available in the protein databases (such as the manually annotated and reviewed data of UniProtKB/Swiss-Prot in [Supplementary Figure S1](#)).

Non-contextual embedding learning for protein sequences

Non-contextual word-embedding techniques aim at learning a global word embedding by ignoring the differences in the meanings of words among contexts. Within bioinformatics, word2vec was the most popular non-contextual word-embedding scheme for biological sequences prior to the emergence of representation models with deep architecture. The word2vec algorithm [40, 41] uses a neural network model (a skip-gram model) to learn word embeddings from a large corpus of text, and the process can be described in three steps: (1) forming a vocabulary from all the unique words in the text. The final objective is to learn a numerical vector for each of the words in this vocabulary; (2) constructing a model for training. Two model architectures, namely, continuous bag-of-words [59, 60] and skip-grams [40, 59, 60], can be utilized for the word2vec algorithm, while skip-grams ([Figure 1c](#)) are widely used both in NLP and bioinformatics; and (3) training the model using samples of type (*input_word*, *output_word*), which are selected by sliding context windows across the text. The words in the context include the center word (target word) and surrounding words (context words) of the center word, which are used as input words and output words, respectively, to form the training samples. The size of the context window is determined by the window width w , namely, the number of words on either side of the target word.

Regarding the training model, the skip-gram ([Figure 1c](#)) model is trained to learn embeddings that predict the probability of a surrounding word occurring given a center word, in which the center word (the target word) is at the input layer and the context words (the surrounding words) are on the output layer. $\{(x, t_i)\}$ denotes training samples from one contextual window, namely, $\{(x, t_1), (x, t_2), \dots, (x, t_C)\}$, where x is the input word (target word) and $\{x_1, x_2, \dots, x_V\}$ is its one-hot vector representation as units of the input layer, the dimension of which is the vocabulary size V , and C is the number of the surrounding words in the window and the window size $w = C + 1$. t_i is the actual i -th output context word and can be represented by a one-hot vector $\{t_{ij}\}$, for $j = 1, 2, \dots, V$, and t_{ij} will only be 1 when the j -th word in the vocabulary is the output word; otherwise, $t_{ij} = 0$. y_i for $i = 1, 2, \dots, C$ is the prediction result of the i -th output word t_i ; hence, a total of C multinomial distributions (y_1, y_2, \dots, y_C) are outputs on the output layer, and each of them can be represented as $\{y_{ij}\}$, for $i = 1, 2, \dots, C$ and $j = 1, 2, \dots, V$, the value of which is the predicted probability that the i -th output context word is the j -th word in the vocabulary.

The training objective of the model is to minimize the prediction error on the units between the actual output t_{ij} and the prediction output y_{ij} by the backpropagation algorithm. However, the trained model is not required, and the objective of training is to obtain the final updated weights between the input layer and the hidden layer, which are represented as matrix $W_{V \times N}$. It is proven that each row of the weight matrix is the embedded vector representation of the target word in the previous study [60]. The parameter N is the dimension of the word vector and should be customized. Through the word2vec scheme, a vocabulary of words and the corresponding vector of each unique word

in the vocabulary are acquired by training on the large corpus of text, and the word embeddings can be used for the downstream tasks. The following points should be considered:

- (1) The pretrained word embeddings of word2vec are static global embeddings because the embedded vector of a word is always the same regardless of where it occurs. The differences in meaning of a word among sentences cannot be distinguished as each word has only one embedding.
- (2) Similar words can be embedded close to one another in the vector space.
- (3) An input word with a V -dimensional one-hot representation is embedded into an N -dimensional continuous vector, and V is typically much larger than N , namely, the word2vec model can embed words from a discrete space with high dimensionality into a continuous space with low dimensionality.
- (4) The window width w and the dimensionality of the word-embedding vector N are important embedding hyperparameters and should be adjusted in the embedding model.
- (5) Within bioinformatics, splitting the sequence into words is the key step in the word2vec embedding scheme. In previous studies, protein sequences have been typically represented by lists of n -grams that represent n consecutive amino acids of a protein sequence as a single unit.

Transfer learning-based representation learning for protein sequence

Transfer learning-based representation learning schemes can be roughly divided into LSTM-based and transformer-based embeddings, both of which use the deep neural network architecture. Pretrained models that are based on popular language models such as ELMo (Embeddings from Language Models) [56] and BERT [46] are representative models of the two types of deep contextualized word representations.

LSTM-based embedding schemes

(ELMo and its derived models). LSTM-based embedding schemes refer to sequence representation approaches that use unsupervised learning that is implemented by the architectures of multiple layers of bidirectional or unidirectional LSTMs, which take the form of next-token prediction. A model is trained on a huge dataset of unlabelled sequences to predict the most likely next token in a sequence based on all previous tokens of this sequence, which is represented as $p(x_i | x_1, x_2, \dots, x_{i-1})$, where x_i is the token at position i of the sequence and each token is typically a character or a word in the sequence. For protein tasks, such as secondary structure prediction and localization prediction [61], the LSTM-based sequence representation approaches are typically conducted on the character level, namely, each amino acid in the sequence as one token is the input of the representation learning model.

The representation learning models that are based on ELMo [56] and its variants are widely used LSTM-based embedding schemes. The ELMo model ([Figure 1d](#)) is a task-specific linear combination of the intermediate layer representations from the bidirectional LSTM language model, which is a contextual and non-static word-embedding scheme. The process of training the bidirectional LSTM model on a large dataset and computing the ELMo vectors for the downstream tasks can be broadly described

by the following steps: (1) Prepare and preprocess data. An unsupervised dataset for pretraining the LSTM-based representation model and a supervised dataset for training the downstream task model are required. A vocabulary file that contains all unique tokens in the unsupervised dataset should be created. The training data for the representation learning model are tokenized sequences. (2) Train the bidirectional representation model on the large dataset of unsupervised sequences. The model combines both a forward language model and a backward representation model, which contain L -layer forward and backward LSTMs (L hidden layers), respectively. The model training aims at predicting the next token for each input token in the sequence, in which for a sequence of N tokens, (t_1, t_2, \dots, t_N) , both the forward and backward representation models compute the probability of the sequence by modeling the probabilities of each token, namely, $p(t_k|t_1, \dots, t_{k-1})$ and $p(t_k|t_{k+1}, \dots, t_N)$, respectively, for $k = 1, 2, \dots, N$. Thus, the loss of training is calculated as the sum of the log likelihoods of the forward and backward directions: $\text{Loss} = \sum_{k=1}^N (\log p(t_k|t_1, \dots, t_{k-1}) + \log p(t_k|t_{k+1}, \dots, t_N))$. Finally, the trained model and a set of $2L + 1$ representations for each token, which include the hidden states of each of L hidden layers in two directions and the input of the embedded vector, namely, $R_k = \{x_k, h_{k,j}^{\text{forward}}, h_{k,j}^{\text{backward}} | j = 1, 2, \dots, L\}$, are acquired. (3) Compute the ELMo vector for a specified downstream task by training the task model on the supervised dataset. The learned set of representations can form the ELMo vector for each token, namely, $\text{ELMo}_k^{\text{task}} = E(R_k; \Theta^{\text{task}}) = \gamma^{\text{task}} (\sum_{j=0}^L s_j^{\text{task}} h_{k,j})$, in which Θ^{task} denotes the parameters of the linear combination for computing the ELMo vector, which include the softmax-normalized weights s_j^{task} for the representation vectors of the token and the scalar parameter γ^{task} that enables the task model to scale the ELMo vector, and they can be learned by training the task-specific model.

The following special explanations are provided:

- (i) Given a sequence of N tokens (t_1, t_2, \dots, t_N) , the input vector representations of each token in sequence (x_1, x_2, \dots, x_N) are context-independent token representations that are acquired from other neural language models, such as CNN-BIG-LSTM (large number of LSTM units along with character-level convolutional neural networks) [62] and character convolutional neural networks (CharCNN) [63].
- (ii) In each of the L hidden layers, there is one LSTM cell (Figure 1f), but N tokens are fed to the cell unit at N time steps, in each of which the networks share the weights. The outputs of the LSTM cell at the k -th time step, namely, the hidden state h_k , can be regarded as a representation of the token t_k , which are computed as:

$$\begin{aligned}
 f_k &= \sigma(W_{\text{forget}} \bullet [h_{k-1}, x_k] + b_{\text{forget}}) \\
 i_k &= \sigma(W_{\text{input}} \bullet [h_{k-1}, x_k] + b_{\text{input}}) \\
 \tilde{C}_k &= \tanh(W_{\text{cell}} \bullet [h_{k-1}, x_k] + b_{\text{cell}}) \\
 C_k &= f_k * C_{k-1} + i_k * \tilde{C}_k \\
 o_k &= \sigma(W_{\text{output}} \bullet [h_{k-1}, x_k] + b_{\text{output}}) \\
 h_k &= o_k * \tanh(C_k),
 \end{aligned}$$

where f_k , i_k and o_k are the outputs from neural network layers of LSTM networks, which are called the forget gate layer, input gate layer and output gate layer, respectively, and \tilde{C}_k and C_k are the candidate and updated cell states, respectively. The input

of the k -th time step x_k and the output of the last step h_{k-1} are concatenated as the input of three gate layers, which are the sigmoid layers, by outputting a vector of values that are between 0 and 1 to control or select the information. Similarly, the tanh layer outputs a vector of values that are between -1 and 1 to create a new candidate cell state \tilde{C}_k that can be added to the state to update the cell state. The cell state is the key element of LSTM networks, which runs through all time steps with only minor linear interactions to realize long-term memory. The output state of each time step h_k is converted into input for the next time step to realize short-term memory.

(3) In the computation of the ELMo vector, the representations from each LSTM layer $h_{j,k}$ for $j = 1, 2, \dots, L$ correspond to the concatenation of the forward hidden state and backward hidden state, namely, $[h_{k,j}^{\text{forward}}, h_{k,j}^{\text{backward}}]$, whereas $h_{j,k}$ for $j = 0$ corresponds to the vector representation that is learned from the other context-independent word-embedding model, namely, x_k .

The following points should be considered: (1) The ELMo vector can be derived not only from the linear combination of the intermediate layer representations but also from only the top layer of the LSTM network, which can be referred to as a variant of the ELMo vector. (2) The learned ELMo vector $\text{ELMo}_k^{\text{task}}$ can be directly fed as the input to the supervised model of the downstream task, while the ELMo enhanced representation, such as $[x_k; \text{ELMo}_k^{\text{task}}]$, which is obtained by concatenating the ELMo vector with x_k can also be passed to the downstream task model. In addition, the ELMo vector can be used at the output layer of the downstream task by concatenating it with the hidden states of the downstream model. (3) The parameters of the bidirectional LSTM model, which include the number of LSTM layers L , the dimension projection and the units of the L layers, should be assigned. Bidirectional LSTMs can capture various types of contextual information. Previous studies on NLP [64] have shown that the lower layers easily capture syntactic information, while the higher layers represent semantic information; therefore, the selection of L is important. In the ELMo model, $L = 2$ bidirectional LSTM layers are used. Assume the dimension projection, namely, the dimension of the hidden state vector $h_{k,j}^{\text{forward}}$ or $h_{k,j}^{\text{backward}}$, is M . Then, the bidirectional L -layer LSTMs have $8 \times M$ units because there are four neural network layers (three gate layers and a tanh layer, four yellow rectangles as presented in Figure 1f) in each of the forward and backward LSTM layers, and the dimension of h_k , which is the concatenation of the hidden-state vectors, is $2M$, which is the same as the dimension of the input token embedded vector x_k .

For protein tasks, the LSTM-based sequence representation approaches are usually conducted on the character level, namely, each amino acid in a sequence is input as a token into the sequence embedding model. The input amino acid sequence is typically embedded by the uncontextualized token representation before the LSTM language model-based embedding, which can map each amino acid of the input sequence to a fixed-length vector without considering the information of neighboring words, such as CharCNN. The final representation of each amino acid in sequence is acquired from the hidden states of the forward and backward directions of each LSTM layer. In addition to the directional LSTMs, the multilayers of unidirectional LSTMs are also used for the embedding scheme. For instance, the UniRep (the unified representation) approach [48] uses a model with 1900-hidden-unit multiplicative LSTM to learn amino acid embeddings, in which the UniRep representation is constructed as the average of the 1900-unit hidden states of the model, which can integrate information across distant amino acids. The LSTM-based model can capture

general features of protein fitness landscapes that extend beyond task-specific training data.

Transformer-based embedding scheme

(BERT and its derived models). The transformer model [65] uses the encoder-decoder architecture, which is formed by stacked encoders and decoders. Each encoder contains two parts: a self-attention layer and a feed-forward layer (dense layer). Each decoder also contains a two-layer network that corresponds to the encoder, but there is an attention layer between the two layers. The most popular language model at present, namely BERT, is based on transformers that are widely used in NLP, such as machine translation, speech recognition and text mining. In bioinformatics, BERT-based models have been used for biomedical text mining [47], structure prediction and protein engineering [66].

The encoder-decoder framework (Figure 1g) is a widely used architecture in deep learning, the process of which is to generate a target (y_1, y_2, \dots, y_n) through the intermediate representation of the input sentence $x = (x_1, x_2, \dots, x_m)$. The encoder and decoder can choose different models according to the tasks of NLP, such as text processing and speech recognition, for which the recurrent neural network (RNN) model is typically used, and image processing, for which the CNN model is always used. Here, we consider the RNN encoder-decoder as an example. The encoder aims at encoding the input into an intermediate semantic representation C through a nonlinear transformation, which is represented as $C = f_c(h_1, h_2, \dots, h_m)$, where $h_t = f_{enc}(x_t, h_{t-1})$ is a hidden state of the encoder model at time t , while the decoder generates output at each time step according to the intermediate semantic representation C through a nonlinear transformation $y_i = f_{dec}(C, y_{i-1})$, for $i = 1, 2, \dots, n$; f_{enc} , f_c and f_{dec} are nonlinear functions. In the encoder-decoder framework, there are no differences among the semantic encodings C of all input words that are used to generate the output word at each time step, namely, all words of the input source have the same influence on the generation of the target output y_i . If the input source is long, as the semantic information is reflected by only one semantic vector C , the information of each input word has disappeared, and a substantial amount of detailed information will be lost. Therefore, the introduction of the attention mechanism into the encoder-decoder architecture is important.

In the encoder-decoder model with attention mechanism (Figure 1h), the attention distribution coefficient is incorporated into the semantic encoding of each input word, namely, the intermediate semantic representation C , is replaced by a semantic encoding that is adjusted by the attention distribution according to the current output. The semantic encoding in the attention model can be represented as $C_i = f_c(a_{it}, f_{enc}(x_t, h_{t-1})) = f_c(a_{it}, h_t)$, for $i = 1, 2, \dots, n$, $t = 1, 2, \dots, m$, where m is the length of the input source sentence, n is the number of outputs, and a_{it} is the attention distribution coefficient for the t -th input word of the source sentence when outputting the i -th target word. f_{enc} denotes the transformation function of the input in the encoder model, and f_c denotes the transformation function for generating the semantic representation of the whole input sentence according to the intermediate representation of each input word and is essentially a weighted sum function. Then, C_i can be represented as $C_i = \sum_{t=1}^m a_{it} h_t$, where h_t refers to the semantic encoding to which the encoder maps the input x_t . Then, in the decoder, the conditional probability of generating the output at time step i can be defined as $p(y_i | y_1, y_2, \dots, y_{i-1}, x) = f_d(y_{i-1}, S_i, C_i)$,

where S_i is the hidden state for time i , which is computed as $S_i = f_{dec}(S_{i-1}, y_{i-1}, C_i)$. The semantic vector C_i , which was computed previously, depends on a sequence of encoding representations (h_1, h_2, \dots, h_m) , and there are many approaches to computing the weight a_{it} , among which the classical computation is $a_{it} = \frac{\exp(e_{it})}{\sum_{t=1}^m \exp(e_{it})}$, where $e_{it} = f_a(S_{i-1}, h_t)$, e_{it} refers to the alignment of the output at time i , namely, the match score between the output words at time i and all time steps of the input sequence, and f_a usually uses the tanh activation function [67].

In the encoder-decoder architecture with attention mechanism, an attention mechanism is utilized between the target and all elements of the input source, while encoder-decoder architecture with self-attention refers to the use of an attention mechanism between the internal elements of the input source or the elements of the target. The self-attention model dynamically generates the weights between connections, which can capture the detailed features between words in the same sentence, such as syntactic or semantic features in a language model. Moreover, the introduction of self-attention into a model can facilitate the capture of the long-distance interdependent features in sentences and can help increase the parallelism of the calculations. Therefore, self-attention is gradually being widely used.

As discussed above, the transformer model (Figure 1i) consists of stacked encoders and decoders that contain self-attention and attention. The main process of the transformer model can be described as follows: (1) first, the inputs are processed via an embedding operation, and positional encoding information is added to them, after which they are passed to the encoder. (2) Encoder layers are stacked together. Each encoder layer contains two sublayers, namely, a self-attention and a fully connected feed-forward network, both of which are added with residual connection and normalization. (3) Decoder layers are also stacked together, each of which is passed with the outputs of the final encoder layer. Compared to the encoder layers, a third sublayer, namely, the attention layer is inserted and the self-attention sublayer in decoder stack is modified by the masked self-attention. Masking, which is an important technology of the transformer model, refers to the hiding of values to make them have no effect when the parameters are updated during model training. Two types of masking are typically used in the transformer model: padding masking and sequence masking. Padding masking refers to filling various positions with meaningless values or truncating the sequences to the same length; thus, these positions can be assigned very large negative values so that the probabilities of these positions will be close to 0. Sequence masking, which is only used for the self-attention of the decoder, renders the decoder unable to see future information because, for a sequence, the output at time i of the decoder should depend only on the outputs before time i , not the outputs after time i . (4) The output layers include a dense layer and a softmax layer, from which the probability of the target is obtained as the final result of the model.

The following observations are made: (1) The encoder can be calculated in parallel, whereas decoding cannot be conducted for the whole sequence at the same time but is conducted entry by entry, similar to the RNN model. (2) A vector of positional encoding is added with the embedding vector of input words before the first encoder layer and the first decoder layer, which can determine the position of each word in the sequence. The reason for adding positional encoding information is that the transformer model lacks the ability to capture sequential sequences; hence, regardless of how the structure of the sentence is disrupted, the results of the transformer are similar. The positional encoding can be obtained via various approaches, such as calculating with

sine and cosine functions [65] or learning from the embedding model [68–71].

BERT [46], which is based on a bidirectional transformer encoder and could be the state-of-art language representation model at present, contains two main technologies: a transformer architecture and unsupervised pretraining. The main components of the BERT model include (1) representations of inputs, which are obtained by summing three embedding vectors of inputs: token embeddings, segment embeddings and position embeddings; (2) transformer blocks, which contain multilayer bidirectional transformer encoders. The base and large BERT models are proposed by previous studies [46], in which the size of the transformer blocks (the number of layers) is 12 and 24, the hidden size of the model is 768 and 1024, and the number of self-attention heads is 12 and 16, respectively. There are two steps in the BERT framework: pretraining the BERT model and fine-tuning the BERT model. The BERT model should be pretrained by using the unsupervised tasks, and during fine-tuning, the pretrained parameters are used to initialize the BERT model, and the initialized model is fine-tuned using labeled data from the downstream tasks.

From architectures of these types of contextual embedding approaches, BERT (transformer-based representation) are deep architectures with more than 12 layers of neural networks, while the ELMO (LSTM-based representation) are a shallow architecture that only uses three layers of bidirectional LSTMs. Previous studies demonstrate that a model with deep networks can outperform a shallow model when training on a large amount of unlabelled data. Moreover, both BERT and its variants (transformer-based representation) use an encoder-decoder framework that is based on an attention mechanism and can outperform sequential models such as RNN- or LSTM-based models, including by improving the parallelism of the model and alleviating the problems that are posed by the long-term dependency. However, the BERT model applies the masked language model with a bidirectional transformer encoder architecture that can meet the needs of language representation models for the encoding of complete input sentence when only using encoders and not decoders because the BERT model performs very well on tasks of many types.

Other sequence representation methods for some specific protein tasks

Except the widely used representation learning types of the first three groups mentioned above, there are also some task-specific protein sequence representation approaches, such as extracted features-based representation and graph computation-based representation as described in Table 1. For example, in the article [49] about the protein structure prediction by Zhou group, physio-chemical properties of amino acids, PSSM profile and the hidden Markov Model sequence profiles are extracted as features for representing protein sequence to predict the protein structural properties using bidirectional RNNs. A recent research by Andrew et al. [50] presents a deep learning-based system for protein structure prediction, namely AlphaFold, and has received a lot of attention recently because of a considerable advance in protein-structure prediction. In this research, a set of sequence and multiple sequence alignment (MSA) features including sequence MSA results, PSSM profiles, hidden Markov Model sequence profiles, residue index and so on, are calculated by a number of tools (PSI-BLAST [72], HHblits [73], HHpred web server [74], etc.), and are used as

input for the prediction neural network. The articles about drug discovery, design and polypharmacy side effects [51–53] apply the graph convolutional networks (GCN) for predictions. In these works, both node features and edge features are calculated for constructing graphs, and results demonstrate that GCN can learn the features for the interaction information of drug-protein, protein-protein and drug-drug. Though they cannot be widely used for all of protein prediction tasks that use deep learning, these representations are effective to some specific tasks.

Discussion and outlook

According to the theoretical analysis and architecture introduction of each type of representation schemes, the advantages and disadvantages of them are clearly indicated. It is beneficial for the researchers to choose a suitable representation scheme for their own task requirements. For end-to-end representation learning, great quantity of data with labels is essential to learn feature information of sequences by training deep neural networks. Therefore, end-to-end learning-based representation approaches can be applied for the downstream tasks that large labeled data sets are available. Moreover, CNN, RNN (especially LSTM) or combination of them are usually applied as the neural network layers that construct the models of these downstream tasks such as the prediction tasks about protein function and classification (e.g. the prediction of DNA-binding protein, RNA-binding protein, protein-protein interaction and subcellular localization) [30, 35–37]. For non-contextual embedding schemes such as word2vec and doc2vec, downstream tasks that use these representation approaches, such as the predictions of protein family classification, functional properties or disordered protein identification, cannot achieve very good performance because protein sequences must be split into words (i.e. 'n-grams' that is mentioned previous) but there are no criterion or universal methods for sequence division. Thence, the non-contextual embedding approaches are gradually replaced by the contextual embedding schemes such as transfer learning-based representation approaches. Even though both of them are unsupervised learning representation approaches based on the pretrained models, non-contextual embedding approaches mentioned above for protein sequence representation are on word level in which sequence splitting and length of words (i.e. parameter n in 'n-grams') are uncertain, while transfer learning-based representation approach for protein sequences can be conducted on the character level (i.e. each single amino acid in a sequence). Another point that needs to be addressed is the non-contextual embedding architectures are shallow, while the architecture of contextual embedding models is based on deep neural networks. Because of that, more useful feature information of sequence can be learned by the deep representing models. Moreover, transfer learning-based representation approaches can be applied for prediction tasks with large dataset as well as tasks with small dataset. Hence, the state-of-art transfer learning-based representation techniques such as LSTM-based representation (ELMo model) and transformer-based representation (BERT model) can be used for protein prediction tasks. For ELMo-based representation model (LSTM-based model), the final ELMo vector for sequence representation can be used as input features for the prediction model of downstream tasks, while for BERT-based representation model (transformer-based model) the prediction model of downstream task should be constructed in the same way as the BERT model. Moreover, theoretically speaking, BERT-based representation model can learn more and better

useful features of sequences than ELMo-based representation model because that the ability of extracting features of transformer is higher than LSTM, and the train data and parameters that are needed for BERT model are more than ELMo model. LSTM-based representation (ELMo and its derived models) has become popular in some tasks such as the prediction of protein secondary structure, protein function and evolutionary understanding [48, 61], in which excellent performance are achieved. The application of BERT (transformer-based representation model) for protein-related prediction tasks is in the exploratory stage. In addition, there are some special representation approaches for certain specific tasks. For example, graph representation (GCN model) is usually applied for representing the associations between protein–protein pairs, protein–drug pairs or drug–drug pairs in tasks about drug discovery [51–53, 75]; manually extracted feature-based representation methods are often used in prediction of protein structure [49, 50, 76] in which a set of features including various types of protein sequence features and multiple sequence alignment features such as physicochemical properties of amino acids and distance between pairs of residues extracted by various of algorithms or tools are used for representing protein sequence and are then used as input into the task model. It is because that for graph representation nodes and edges in the graph can represent proteins/drugs and interactions between them, respectively; while for manually extracted feature-based representation, more information about the structure may be conveyed from many kinds of extracted sequence features by training deep neural network that can improve the performance of protein-structure prediction.

Due to the availability of a large amount of protein sequence data, sequence-based deep learning techniques have become popular in protein engineering tasks, in which protein embeddings are a major factor in controlling model performance. In this review, we concentrate on the analysis of types of protein embedding approaches. We divided the protein embedding approaches into four large categories and reviewed the development and variants of these types of embedding approaches, and we theoretically described and compared the architectures of types of embedding models. We hope this study will help researchers select suitable protein embedding methods for protein engineering tasks.

Unfortunately, due to the relative immaturity of the application of deep learning technologies in protein engineering, there remains a lack of standardization for protein embedding in this field. Additional comparisons of embedding models that are used for protein tasks must be performed to assess the available methods for protein embedding. Moreover, due to the huge and increasing availability of protein sequences and the increasing power of deep learning technology, the development of effective protein embedding methods to better utilize this technology in bioinformatics is imperative.

Key Points

- This paper reviewed the development and detailed model architectures of protein sequence representation learning approaches.
- We classified protein sequence representation learning approaches according to the language models in nature language processing field.
- We discussed the advantages and disadvantages of each sequence representation category.

- This paper also provided a theoretical guidance for researchers who aim to choose a suitable sequence representation approach for their tasks.

Supplementary Data

Supplementary data are available online at <http://bib.oxfordjournals.org/>

Funding

The work was supported by the National Key R&D Program of China (2018YFC0910405), and the National Natural Science Foundation of China (No. 61922020, No. 61771331, No. 91935302).

Conflicts of interest

The authors declare that they have no conflicts of interest.

References

1. Larranaga P, Calvo B, Santana R, et al. Machine learning in bioinformatics. *Brief Bioinform* 2006;**7**:86–112.
2. Liu B, Gao X, Zhang H. BioSeq-Analysis2.0: an updated platform for analyzing DNA, RNA, and protein sequences at sequence level and residue level based on machine learning approaches. *Nucleic Acids Res* 2019;**47**:e127.
3. Liu KW, Chen W. iMRM: a platform for simultaneously identifying multiple kinds of RNA modifications. *Bioinformatics* 2020;**36**:3336–42.
4. Zhao T, Hu Y, Peng J, et al. DeepLGP: a novel deep learning method for prioritizing lncRNA target genes. *Bioinformatics* 2020;**36**:4466–72.
5. Zhao T, Hu Y, Cheng L. Deep-DRM: a computational method for identifying disease-related metabolites based on graph deep learning approaches. *Brief Bioinform* 2020;**10**. doi: 10.1093/bib/bbaa212.
6. Hu Y, Zhang HH, Liu B, et al. rs34331204 regulates TSPAN13 expression and contributes to Alzheimer's disease with sex differences. *Brain* 2020;**143**(11): e95.
7. Xu L, Liang G, Liao C, et al. An efficient classifier for Alzheimer's disease genes identification. *Molecules* 2018;**23**:3140.
8. Zeng X, Zhu S, Liu X, et al. deepDR: a network-based deep learning approach to in silico drug repositioning. *Bioinformatics* 2019;**35**:5191–8.
9. Maienschein-Cline M, Dinner AR, Hlavacek WS, et al. Improved predictions of transcription factor binding sites using physicochemical features of DNA. *Nucleic Acids Res* 2012;**40**:e175–5.
10. Jansen R, Yu H, Greenbaum D, et al. A Bayesian networks approach for predicting protein–protein interactions from genomic data. *Science* 2003;**302**:449–53.
11. Guo Y, Yu L, Wen Z, et al. Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences. *Nucleic Acids Res* 2008;**36**:3025–30.
12. Zhang X, Liu S. RBPPred: predicting RNA-binding proteins from sequence using SVM. *Bioinformatics* 2017;**33**:854–62.
13. Cai C, Han L, Ji ZL, et al. SVM-Prot: web-based support vector machine software for functional classification of a protein from its primary sequence. *Nucleic Acids Res* 2003;**31**:3692–7.

14. Su R, Wu H, Xu B, et al. Developing a multi-dose computational model for drug-induced hepatotoxicity prediction based on Toxicogenomics data. *IEEE/ACM Trans Comput Biol Bioinform* 2018;**16**:1231–9.
15. Wei L, Wan S, Guo J, et al. A novel hierarchical selective ensemble classifier with bioinformatics application. *Artif Intell Med* 2017;**83**:82–90.
16. Wei L, Xing P, Zeng J, et al. Improved prediction of protein–protein interactions using novel negative samples, features, and an ensemble classifier. *Artif Intell Med* 2017;**83**:67–74.
17. Li C-C, Liu B. MotifCNN-fold: protein fold recognition based on fold-specific features extracted by motif-based convolutional neural networks. *Brief Bioinform* 2020. doi: [10.1093/bib/bbz133](https://doi.org/10.1093/bib/bbz133).
18. Liu B, Zhu Y, Yan K. Fold-LTR-TCP: protein fold recognition based on triadic closure principle. *Brief Bioinform* 2020. doi: [10.1093/bib/bbz139](https://doi.org/10.1093/bib/bbz139).
19. Wang H, Ding Y, Tang J, et al. Identification of membrane protein types via multivariate information fusion with Hilbert-Schmidt independence criterion. *Neurocomputing* 2020;**383**:257–69.
20. Li J, Pu Y, Tang J, et al. DeepAVP: a dual-channel deep neural network for identifying variable-length antiviral peptides. *IEEE J Biomed Health Inform* 2020;**24**:3012–19.
21. Shen Y, Tang J, Guo F. Identification of protein subcellular localization via integrating evolutionary and physicochemical information into Chou's general PseAAC. *J Theor Biol* 2019;**462**:230–9.
22. Shen Y, Ding Y, Tang J, et al. Critical evaluation of web-based prediction tools for human protein subcellular localization. *Brief Bioinform* 2019;**21**:1628–40.
23. Cheng L. Computational and biological methods for gene therapy. *Curr Gene Ther* 2019;**19**:210–0.
24. Cheng L, Zhao H, Wang P, et al. Computational methods for identifying similar diseases. *Mol Therapy Nucleic Acids* 2019;**18**:590–604.
25. Xu L, Liang G, Liao C, et al. K-skip-n-gram-RF: a random Forest based method for Alzheimer's disease protein identification. *Front Genet* 2019;**10**:33.
26. Xu L, Jiang S, Wu J, et al. An in silico approach to identification, categorization and prediction of nucleic acid binding proteins. *Brief Bioinform* 2020. doi: [10.1093/bib/bbaa171](https://doi.org/10.1093/bib/bbaa171).
27. Shao J, Yan K, Liu B. FoldRec-C2C: protein fold recognition by combining cluster-to-cluster model and protein similarity network. *Brief Bioinform* 2020. doi: [10.1093/bib/bbaa144](https://doi.org/10.1093/bib/bbaa144).
28. Zhu XJ, Feng CQ, Lai HY, et al. Predicting protein structural classes for low-similarity sequences by evaluating different features. *Knowl-Based Sys* 2019;**163**:787–93.
29. Jones DT. Protein secondary structure prediction based on position-specific scoring matrices. *J Mol Biol* 1999;**292**:195–202.
30. ElAbd H, Bromberg Y, Hoarfrost A, et al. Amino acid encoding for deep learning applications. *BMC Bioinformatics* 2020;**21**:1–14.
31. Jin S, Zeng X, Xia F, et al. Application of deep learning methods in biological networks. *Brief Bioinform* 2020. doi: [10.1093/bib/bbaa043](https://doi.org/10.1093/bib/bbaa043).
32. Zeng X, Lin W, Guo M, et al. A comprehensive overview and evaluation of circular RNA detection tools. *PLoS Comput Biol* 2017;**13**:e1005420.
33. Jurtz VI, Johansen AR, Nielsen M, et al. An introduction to deep learning on biological sequence data: examples and solutions. *Bioinformatics* 2017;**33**:3685–90.
34. Liu X, Hong Z, Liu J, et al. Computational methods for identifying the critical nodes in biological networks. *Brief Bioinform* 2020;**21**:486–97.
35. Kulmanov M, Khan MA, Hoehndorf R. DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics* 2018;**34**:660–8.
36. Qu Y-H, Yu H, Gong X-J, et al. On the prediction of DNA-binding proteins only from primary sequences: a deep learning approach. *PLoS One* 2017;**12**:e0188129.
37. Almagro Armenteros JJ, Sønderby CK, Sønderby SK, et al. DeepLoc: prediction of protein subcellular localization using deep learning. *Bioinformatics* 2017;**33**:3387–95.
38. Tang Y-J, Pang Y-H, Liu B. IDP-Seq2Seq: identification of intrinsically disordered regions based on sequence to sequence learning. *Bioinformatics* 2020. doi: [10.1093/bioinformatics/btaa667](https://doi.org/10.1093/bioinformatics/btaa667).
39. Xu L, Liang G, Wang L, et al. A novel hybrid sequence-based model for identifying anticancer peptides. *Gen* 2018;**9**:158.
40. Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality. In: *Advances in neural information processing systems*, Curran Associates Inc., 57 Morehouse LaneRed Hook, NY, United States. 2013, 3111–9.
41. Goldberg Y, Levy O. word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*. 2014.
42. Asgari E, Mofrad MR. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLoS One* 2015;**10**:e0141287.
43. Zhang H, Liao L, Cai Y, et al. IVS2vec: a tool of inverse virtual screening based on word2vec and deep learning techniques. *Methods* 2019;**166**:57–65.
44. Le Q, Mikolov T. Distributed representations of sentences and documents. In: *International conference on machine learning*, 2014, 1188–96.
45. Yang KK, Wu Z, Bedbrook CN, et al. Learned protein embeddings for machine learning. *Bioinformatics* 2018;**34**:2642–8.
46. Devlin J, Chang M-W, Lee K, et al. Bert: pre-training of deep bidirectional transformers for language understanding. *arXiv preprint* 2018;**1810**:04805.
47. Lee J, Yoon W, Kim S, et al. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 2020;**36**:1234–40.
48. Alley EC, Khimulya G, Biswas S, et al. Unified rational protein engineering with sequence-based deep representation learning. *Nat Methods* 2019;**16**:1315–22.
49. Heffernan R, Yang Y, Paliwal K, et al. Capturing non-local interactions by long short-term memory bidirectional recurrent neural networks for improving prediction of protein secondary structure, backbone angles, contact numbers and solvent accessibility. *Bioinformatics* 2017;**33**:2842–9.
50. Senior AW, Evans R, Jumper J, et al. Improved protein structure prediction using potentials from deep learning. *Nature* 2020;**577**:706–10.
51. Fout A, Byrd J, Shariat B, et al. Protein interface prediction using graph convolutional networks. In: *Advances in neural information processing systems*, Curran Associates Inc., 57 Morehouse LaneRed Hook, NY, United States. 2017, 6530–9.
52. Zitnik M, Agrawal M, Leskovec JJB. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics* 2018;**34**:i457–66.

53. Zhao T, Hu Y, Valsdottir LR, et al. Identifying drug–target interactions based on graph convolutional network and deep neural. *Brief Bioinform* 2020. doi: [10.1093/bib/bbaa044](https://doi.org/10.1093/bib/bbaa044).
54. Tan JX, Li SH, Zhang ZM, et al. Identification of hormone binding proteins based on machine learning methods. *Math Biosci Eng* 2019;**16**:2466–80.
55. Fu X, Cai L, Zeng X, et al. StackCPPred: a stacking and pairwise energy content-based prediction of cell-penetrating peptides and their uptake efficiency. *Bioinformatics* 2020;**36**:3028–34.
56. Peters M, Neumann M, Iyyer M et al. Deep Contextualized Word Representations. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana, 2018, p. 2227–37. Association for Computational Linguistics.
57. Zeng X, Zhu S, Hou Y, et al. Network-based prediction of drug–target interactions using an arbitrary-order proximity embedded deep forest. *Bioinformatics* 2020;**36**:2805–12.
58. Hong Z, Zeng X, Wei L, et al. Identifying enhancer–promoter interactions with neural network based on pre-trained DNA vectors and attention mechanism. *Bioinformatics* 2020;**36**:1037–43.
59. Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space. *arXiv preprint:13013781* 2013.
60. Rong X. word2vec parameter learning explained. *arXiv preprint:14112738* 2014.
61. Heinzinger M, Elnaggar A, Wang Y, et al. Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinformatics* 2019;**20**:723.
62. Jozefowicz R, Vinyals O, Schuster M, et al. Exploring the limits of language modeling. *arXiv preprint:160202410* 2016.
63. Kim Y, Jernite Y, Sontag D, et al. Character-aware neural language models. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. Phoenix. Arizona: AAAI Press, 2016, 2741–9.
64. Peters ME, Neumann M, Iyyer M, et al. Deep contextualized word representations. *arXiv preprint* 2018;**1802**:05365.
65. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: *Advances in neural information processing systems*, Curran Associates Inc., 57 Morehouse Lane Red Hook, NY, United States. 2017, 5998–6008.
66. Rao R, Bhattacharya N, Thomas N, et al. Evaluating protein transfer learning with TAPE. In: *Advances in Neural Information Processing Systems*, Curran Associates Inc., 57 Morehouse Lane Red Hook, NY, United States. 2019, 9689–701.
67. Bahdanau D, Cho K, Bengio YJ. Neural machine translation by jointly learning to align and translate. *arXiv* 2014, 14090473.
68. Gehring J, Auli M, Grangier D, et al. Convolutional sequence to sequence learning. *arXiv* 2017, 170503122.
69. Liu B, Li K. iPromoter-2L2.0: identifying promoters and their types by combining smoothing cutting window algorithm and sequence-based features. *Mol Therapy-Nucleic Acids* 2019;**18**:80–7.
70. Cheng L, Hu Y. Human disease system biology. *Curr Gene Ther* 2018;**18**:255–6.
71. Cheng L. Omics data and artificial intelligence: new challenges for gene therapy. *Curr Gene Ther* 2020;**20**(1):1.
72. Altschul SF, Madden TL, Schäffer AA, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* 1997;**25**:3389–402.
73. Remmert M, Biegert A, Hauser A, et al. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nat Methods* 2012;**9**:173–5.
74. Söding J, Biegert A, Lupas ANJNar. The HHpred interactive server for protein homology detection and structure prediction. *Nucleic Acids Res* 2005;**33**:W244–8.
75. Sun M, Zhao S, Gilvary C, et al. Graph convolutional networks for computational drug development and discovery. *Brief Bioinform* 2020;**21**:919–35.
76. Wang T, Qiao Y, Ding W, et al. Improved fragment sampling for ab initio protein structure prediction using deep neural networks. *Nat Machine Intell* 2019;**1**:347–55.