Interdisciplinary Sciences: Computational Life Sciences (2025) 17:744–769
https://doi.org/10.1007/s12539-024-00673-4

ORIGINAL RESEARCH ARTICLE

EnDM-CPP: A Multi-view Explainable Framework Based on Deep Learning and Machine Learning for Identifying Cell-Penetrating Peptides with Transformers and Analyzing Sequence Information

Lun Zhu[1] · **Zehua Chen**[1] · **Sen Yang**[1,2]

Received: 4 June 2024 / Revised: 28 October 2024 / Accepted: 1 November 2024 / Published online: 23 December 2024

Abstract

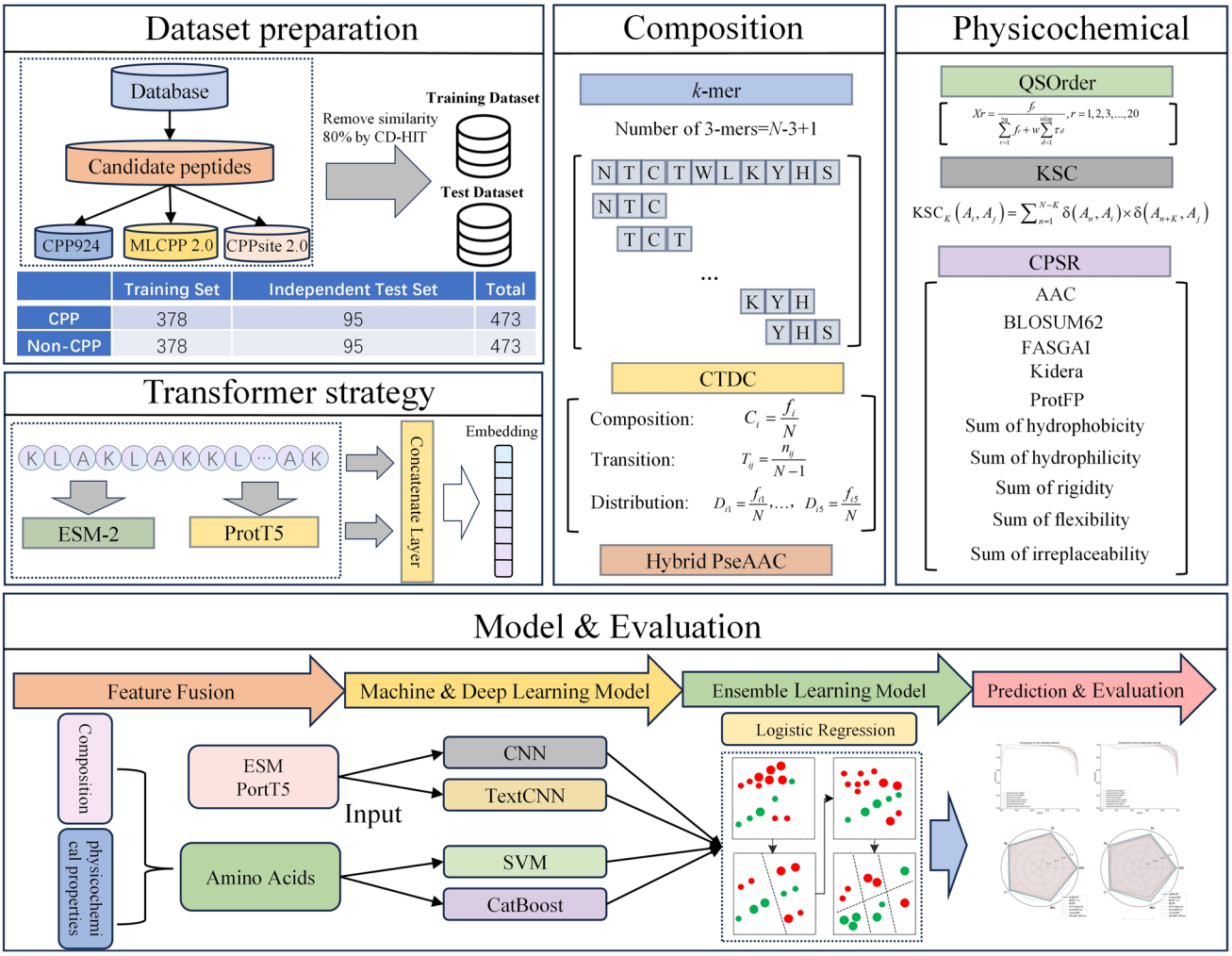
Cell-Penetrating Peptides (CPPs) are a crucial carrier for drug delivery. Since the process of synthesizing new CPPs in the laboratory is both time- and resource-consuming, computational methods to predict potential CPPs can be used to find CPPs to enhance the development of CPPs in therapy. In this study, EnDM-CPP is proposed, which combines machine learning algorithms (SVM and CatBoost) with convolutional neural networks (CNN and TextCNN). For dataset construction, three previous CPP benchmark datasets, including CPPsite 2.0, MLCPP 2.0, and CPP924, are merged to improve the diversity and reduce homology. For feature generation, two language model-based features obtained from the Transformer architecture, including ProtT5 and ESM-2, are employed in CNN and TextCNN. Additionally, sequence features, such as CPRS, Hybrid PseAAC, KSC, etc., are input to SVM and CatBoost. Based on the result of each predictor, Logistic Regression (LR) is built to predict the final decision. The experiment results indicate that ProtT5 and ESM-2 fusion features significantly contribute to predicting CPP and that combining employed features and models demonstrates better association. On an independent test dataset comparison, EnDM-CPP achieved an accuracy of 0.9495 and a Matthews correlation coefficient of 0.9008 with an improvement of 2.23%–9.48% and 4.32%–19.02%, respectively, compared with other state-of-the-art methods. Code and data are available at https://github.com/tudou1231/EnDM-CPP.git.

✉ Sen Yang
ys@cczu.edu.cn

[1] School of Computer Science and Artificial Intelligence, Aliyun School of Big Data, School of Software, Changzhou University, Changzhou 213164, China

[2] The Affiliated Changzhou No. 2 People's Hospital of Nanjing Medical University, Changzhou 213164, China

Graphical Abstract



Keywords Cell-penetrating peptide · Transformer-based feature · Peptide intrinsic feature · Stacking model

1 Introduction

Cell-Penetrating Peptides (CPPs) have a unique biological function of efficiently crossing cell membranes, and their permeability to cells is almost unrestricted by cell type and species [1, 2]. Since the first CPP was discovered on the transmembrane protein transduction domain TAT of the human immunodeficiency virus (HIV) in 1991, several multi-cationic CPPs have been identified, including the fruit fly antennal allogeneic protein Antp and the structural protein vp22 of herpes simplex virus [3]. CPP sequences with 5–30 short-chain amino acid residues have demonstrated their irreplaceable value in various biomedical applications, especially in facilitating the intracellular delivery of macromolecular drugs that are difficult to penetrate biological membranes. CPPs have been used to deliver small-molecule drugs, nucleic acids, proteins, and other biomolecules [4] and have important implications for cell therapy and regenerative medicine. CPPs have broad application prospects in multiple medical fields, including drug delivery for neurological diseases, antibacterial therapy, skin disease treatment, and metabolic diseases. In addition, CPPs have also played an important role in vaccine development by improving antigen delivery and enhancing immune responses [5]. With the swift improvement of next-generation sequencing technology, we can discover new peptides, including potential novel CPPs. Therefore, fast and accurate identification of CPPs from these large amounts of peptide data has become an urgent need. However, using traditional experimental methods to predict CPPs is time-consuming and expensive [6, 7]. In view of this, adopting computational methods for rapid prediction becomes very important. Although machine

learning and deep learning technologies are more efficient than experimental methods in terms of time and cost, their reliability still falls short. Therefore, improving the prediction accuracy of CPPs remains an urgent issue.

Some computational methods have been established previously to predict CPPs. In 2005, Hällbrink et al. [8] used the *z*-scale of physicochemical descriptors to identify CPPs, laying the foundation for identifying CPPs using computational methods. In 2010, Dobchev et al. [9] concentrated on using artificial neural networks (ANN) and principal component analysis (PCA) to predict CPPs. They demonstrated the feasibility of deep learning in identifying and predicting CPPs through experimental results. In 2018, Manavalan et al. [10] proposed the MLCPP method for recognizing CPPs in protein sequences, providing a powerful tool for CPP recognition and application. In the same year, Pandey et al. [11] proposed the KELM-CPPpred model based on kernel extreme learning machines for predicting and identifying CPPs, further promoting the development of CPP prediction methods. In 2020, Fu et al. [12] proposed the StackCPPred model to improve the recognition accuracy of CPPs by combining multiple prediction models, providing an effective computational framework for CPP prediction. In the same year, Arif et al. [13] proposed TargetCPP for CPP prediction by using optimized multi-scale features and gradient-boosting decision trees, providing new perspectives and tools for research in the field of drug delivery. In 2021, de Oliveira et al. [14] proposed the BChemRF CPPred to improve the accurate prediction ability of CPPs by combining machine learning frameworks and multiple classifiers. In 2022, Manavalan et al. [15] proposed MLCPP 2.0. As an updated version of MLCPP, MLCPP 2.0 adopts a new stacking method and improves peptide sequence information to achieve robust CPP prediction. In the same year, Park et al. [16] proposed AiCPP. AiCPP is a deep learning-based model used to screen and optimize CPPs quickly. Finally, in 2023, Zhang et al. [17] proposed Siamese CPP. This is a sequence based twin network, and its innovative deep learning framework can automatically predict CPPs. SiameseCPP improves prediction accuracy by learning discriminative representations of CPPs.

CPPs are a class of short peptides that can penetrate cell membranes and help drug molecules enter cells, thus improving therapeutic efficacy [1]. As the application of bioinformatics in drug development continues to expand, the prediction of CPPs through data-driven approaches has become particularly crucial. Multiple existing CPP prediction datasets contain different CPP sequences and properties, respectively, but these datasets suffer from limited sample size, insufficient diversity, and homologous sequence interference when used individually.

This paper proposes a high-quality dataset to predict CPPs, which is obtained by merging three datasets, namely CPPsite 2.0, MLCPP 2.0 and CPP924 [15, 18, 19]. Firstly, we reviewed each original dataset, including removing duplicates and ensuring data consistency. Secondly, by the CD-HIT tool [20], sequences with more than 80% homology in all datasets were removed to reduce redundancy and bias and ensure the datasets' independence and reliability. Finally, the obtained dataset enlarged the sample size and improved the data coverage.

Feature selection is crucial in improving the performance of machine learning models in the study of CPP recognition. Previous studies have some limitations in feature selection, mainly in the following aspects: (1) Feature redundancy and correlation: many previous studies have adopted features with high redundancy and correlation, which may lead to overfitting and a decrease in model performance. (2) Some studies have chosen features with excessively high dimensionality, increasing computational complexity and potentially masking the role of crucial features. (3) The features selected in some studies are not strongly correlated with the biological features of CPPs, which affects the biological interpretability of the model. To address the above issues, the researcher selected six features, CPSR, CTDC, Hybrid PseAAC, *k*-mer, QSOrder, and KSC, as the optimal features to input into the machine learning model [21].

In the feature selection process of deep learning models, we referred to large-scale protein language models based on protein sequences. TAPE provides a multi-task learning framework with several different models, such as Transformer and LSTM [22]. TAPE evaluates the effectiveness of these models by their performance on various protein tasks, thereby generating effective protein sequence embeddings. ProtBERT is a model based on the BERT architecture. ProtBERT uses a Mask Language Model (MLM) for training, which can capture local and global features in protein sequences [23]. ProtT5 is a bidirectional encoder model based on Transformer architecture, which uses the sequence to sequence (Seq2Seq) to train the target. It can handle complex sequence transformation tasks and is suitable for various tasks such as protein function prediction and structural modeling [24]. The ESM-2 model predicts protein structure and function by learning evolutionary information from numerous protein sequences. We have comprehensively considered the advantages and disadvantages of the above feature extraction methods. Among them, TAPE performs well in some specific tasks, but its generalization performance and specificity are insufficient in CPP feature extraction. The feature extraction of CPPs not only requires capturing local and global information of the sequence but also dealing with complex sequence patterns and transformations. The training objective of ProtBERT's masked language model mainly focuses on filling in missing labels, which is not flexible enough for complex sequence generation and transformation tasks. Compared to ProtBERT, ProtT5 has more robust

capabilities and adaptability in handling CPP feature extraction tasks due to its Seq2Seq training target and bidirectional encoder architecture. ESM-2 can effectively capture global contextual features from protein sequences. It is particularly adept at extracting evolutionary information from sequence data. This is advantageous for identifying and predicting complex patterns and functional characteristics in CPP. Therefore, we chose to input the merged features of ProtT5 and ESM-2 into the deep learning model. The purpose of this method is to utilize each model's advantages to obtain more accurate and comprehensive CPP recognition results. By using the advanced feature representation capabilities of these two models, our model demonstrates the ability to accurately predict the properties of CPPs within the deep learning framework.

As shown in Fig. 1, we innovatively constructed a specialized dataset, which was rigorously filtered and customized to ensure its close association with CPP predictions. Figure 1 provides a detailed description of how to use deep learning models ESM-2 and ProtT5 for feature extraction. We have selected six key fusion features—CPSR, CTDC, Hybrid PseAAC, *k*-mer, QSOrder, and KSC—and presented their expression formulas. EnDM-CPP is a multi-perspective framework that fuses deep learning and machine learning, integrating four models: SVM, CatBoost, TextCNN, and CNN. We adopted a unique method of extracting probability

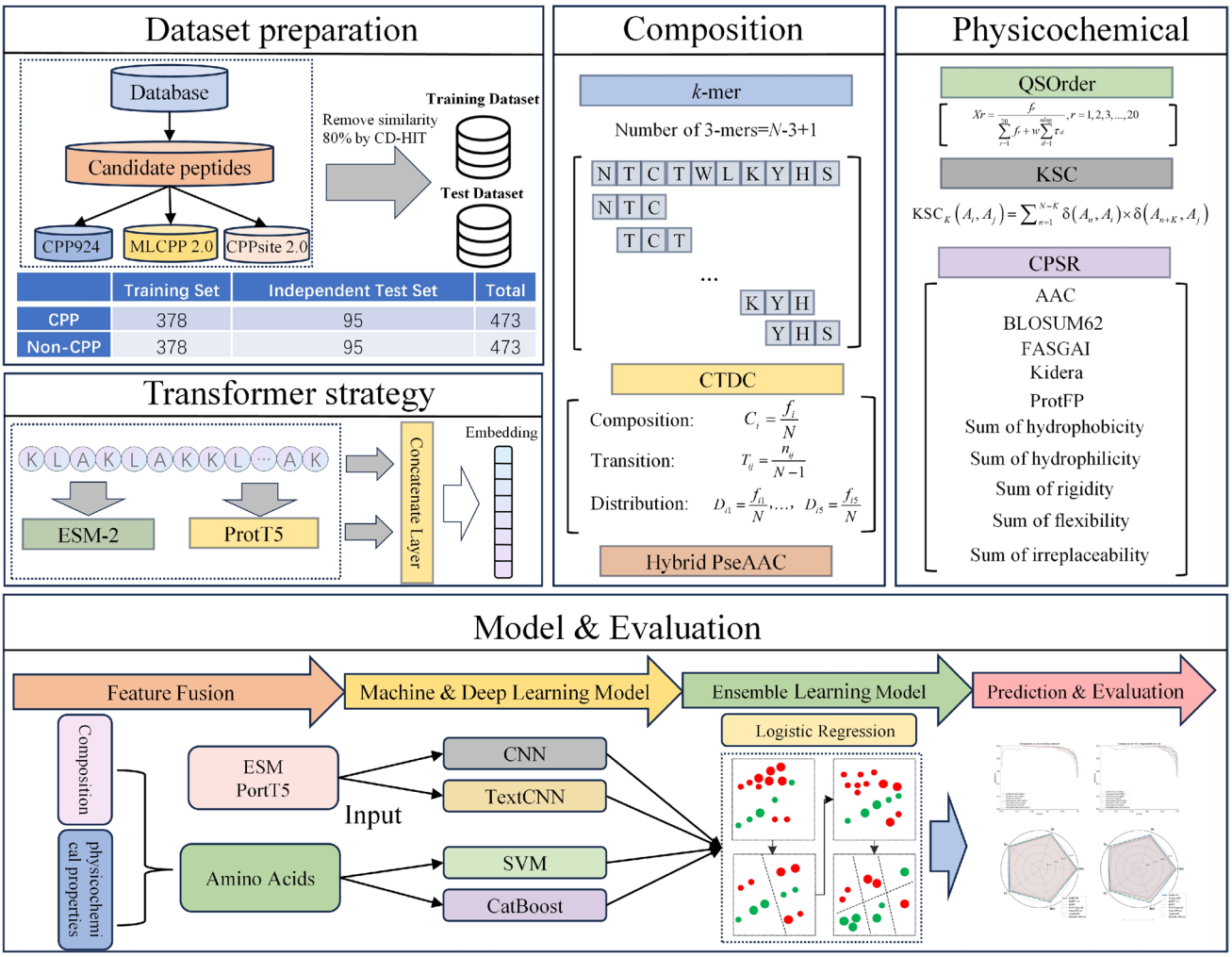


Fig. 1 The overall process of the EnDM-CPP framework can be summarized in the following three steps: (1) The three datasets, CPPsite 2.0, MLCPP 2.0, and CPP924, are merged to create an initial dataset, and those sequences with more than 80% similarity are eliminated using the CD-HIT tool to improve the accuracy and efficiency of sequence analysis. (2) Several traditional coding methods include CPSR, KSC, QSOrder, *k*-mer, CTDC, and Hybrid PseAAC, and transformer-based deep learning coding methods include ESM-2 and ProtT5. (3) Implement a stacked integrated learning strategy which involves a two-layer model. In the first layer, four different base learners are used as models, and each model independently predicts the data. Then, the predictions of these four models are fed into the second layer model as new features. The second layer model synthesizes these features to make the final prediction. Additionally, the performance of this stacked integrated learning strategy is compared with other methods to confirm its effectiveness

outputs from each model and fusing them to create a new feature vector, which not only enhances the information fusion ability of the model but also improves the accuracy of prediction. Next, this new feature vector was input to a secondary LR model. This secondary model was trained to generate the results from each previous model and produce the final predicted values [25]. This hierarchical and integrated approach maximizes the strengths of each feature and model, lowers the risk of overfitting, and improves the generalization ability and reliability of the predictive model.

2 Data and Methods

2.1 Data

A strictly reliable dataset is crucial for constructing and evaluating prediction models. The quality of the dataset directly affects the effectiveness and accuracy of the model. We merged MLCPP 2.0, CPPsite 2.0, and CPP924 datasets to create a more comprehensive and diverse dataset for CPP prediction research.

The MLCPP 2.0 dataset consists of multiple training sets, such as C2Pred, CellPPD, CPPred-RF, KELM-CPPpred, MLCPP, and BChemRF CPPred. CPPsite 2.0 provides rich CPP data, including information for predicting its secondary and tertiary structures. This database covers a wide range of sources and types of CPPs, providing a broad analysis perspective [19]. CPP924 is an experimentally validated dataset [12, 17, 26, 27], which guarantees the reliability of the data. However, as an older dataset, it contains a certain percentage of homologous sequences, which may lead to data redundancy during model training. The above individual datasets may have fewer sample sizes and more homologous sequences in specific CPP subcategories, which results in poor generalization ability and predictive performance of the model. We merged three datasets: MLCPP 2.0, CPPsite 2.0, and CPP924. Subsequently, the CD-HIT tool removed sequences with high homology similarity. To study the impact of different similarity thresholds on the model, we selected 75%, 80%, and 85% to process the dataset separately and trained the model using the processed dataset. Then, we tested the models trained on the three datasets using the same independent test set. This was done to evaluate the impact of different similarity thresholds on the model performance. Through this method, we can analyze and compare the effects of varying similarity thresholds on model accuracy, AUC, F1 score, and MCC. This enables us to identify the optimal similarity threshold, enhancing the model's generalization ability and predictive performance.

Table 1 shows that when the similarity threshold is 80%, the ACC, AUC, F1 score, and MCC evaluation metrics of the model are superior to other thresholds. Therefore, we chose a threshold of 80% to process the merged dataset. Through this method, we have successfully constructed a brand-new, diverse, and reliable CPP dataset. This dataset consists of 473 non-CPPs and 473 CPPs. Then, the whole dataset was divided in the ratio of 8:2, where 80% of the data was used as a training dataset for model training and tuning. At the same time, the remaining 20% was used as an independent test dataset for evaluating the model's performance and generalization ability. This balanced design of positive and negative samples allows the model to learn how to distinguish between these two types of samples more effectively. In addition, the diversity of samples not only enhances the model's ability to recognize different CPPs but also effectively reduces the risk of overfitting and enhances the robustness of the model. The statistics of the training and independent test datasets are presented in Table 2.

Table 1 The impact of different CD-HIT similarity thresholds on model performance

Similarity threshold (%)	ACC	AUC	F1 score	MCC
75	0.8842	0.9614	0.8829	0.7683
80	0.9157	0.9763	0.9157	0.8317
85	0.9105	0.9728	0.9100	0.8210

Table 2 Composition of the training and independent test datasets

Label	Training dataset	Independent test dataset	Total
CPP	378	95	473
Non-CPP	378	95	473

2.2 Methods

2.2.1 Feature Extraction

In this study, we face the challenge of transforming primary biological peptides into effective numerical values to accurately identify CPPs. We adopted a multi-feature integration strategy to mine the discriminative information of CPPs from multiple perspectives comprehensively. This strategy incorporates the following eight sequence-based feature descriptors: (1) Composite Protein Sequence Representation (CPSR), (2) Composition, Transition, and Distribution (CTDC), (3) Hybrid Pseudo Amino Acid Composition (Hybrid PseAAC), (4) *k*-mer, (5) Quasi-Sequence-Order (QSOrder), (6) *K*-Spaced Amino Acid Pairs (KSC), (7) ProtT5, (8) Evolutionary Scale Modeling (ESM-2). A specific discussion of the input characterization scheme will be developed in detail in the following sections.

2.2.1.1 Traditional Features

a. Composite Protein Sequence Representation (CPSR)

CPSR is a feature extraction method for protein sequence analysis in bioinformatics. It aims to comprehensively characterize protein sequences by synthesizing multiple types of sequence information. The core idea of CPSR is to combine different feature descriptors to capture various biological properties and functional information of protein sequences [21, 28]. Table 3 shows the total feature space generated. The feature space contains 11 sets, and the number of features corresponding to each set is shown in Table 3. We use CPSR feature encoding to merge into a 60-dimensional vector.

b. Composition, Transition, and Distribution (CTDC)

CTDC is a bioinformatics method for analyzing and characterizing protein or peptide sequences. Its primary function is to extract sequence features. These features are suitable for deeper bioinformatics analysis, such as sequence alignment, structural prediction, and functional annotation. We used CTDC features and encoded them into 39-dimensional vectors. CTDC consists of three main components [38, 39]:

(1) Composition: This part involves calculating the frequency or proportion of each amino acid in the sequence. It reflects the relative abundance of each amino acid in the sequence, thus providing the essential chemical characterization of the sequence. The calculation formula for its composition is

$$C_i = \frac{f_i}{N} \tag{1}$$

where f_i and N are the frequency of amino acid appearances in the sequence and the sequence length, respectively.

(2) Transition: Transition refers to the frequency of changes from one attribute to another in a sequence. For example, it can be the transition frequency from hydrophobic amino acids to hydrophilic amino acids. This reflects the change in chemical properties in the sequence. The calculation formula is

$$T_{ij} = \frac{n_{ij}}{N-1} \tag{2}$$

where n_{ij} is the number of occurrences of amino acids A_j.

(3) Distribution: Distributional characterization describes the positional distribution of specific types of amino acids as they appear in a sequence. This type of analysis helps to understand the spatial layout of specific types of amino acids in a sequence, affecting the protein's three-dimensional structure and function. For amino acids A_i, the calculation formulas for the first, 25%, 50%, 75%, and last occurrence positions in the sequence are

$$D_{i1} = \frac{f_{i1}}{N}, D_{i2} = \frac{f_{i2}}{N}, D_{i3} = \frac{f_{i3}}{N}, D_{i4} = \frac{f_{i4}}{N}, D_{i5} = \frac{f_{i5}}{N} \tag{3}$$

Among them, f_{ik} is the position where the amino acid A_i appears in the sequence.

c. Hybrid Pseudo Amino Acid Composition (Hybrid PseAAC)

PseAAC expresses protein characteristics by analyzing the composition of amino acids and their arrangement order in protein sequences [40]. Hybrid PseAAC, as a developmental version of PseAAC, has been extended to incorporate more biological features based on PseAAC. These features include the physicochemical properties of proteins, secondary structures, and molecular evolutionary information, which provide a richer dimension for the comprehensive analysis of protein sequences [41].

By combining physical and chemical properties, we can better understand the functional roles of amino acids in sequences. The introduction of secondary structural information helps capture the folding state and functional domains of proteins. Molecular evolutionary information provides essential clues in the process of protein evolution. These extended features enhance their biological relevance. We tested Hybrid

Table 3 Feature-based sequence representation

Feature space	Number of features
Amino acid composition [29]	20
Sequence length	1
BLOSUM62-derived indices of the peptide [30]	10
FASGAI vectors of the peptide [31]	6
Kidera factors of the peptide [32]	10
ProtFP descriptors of the peptide [33]	8
Sum of hydrophobicity [34]	1
Sum of hydrophilicity	1
Sum of rigidity [35]	1
Sum of flexibility [36]	1
Sum of irreplaceability [37]	1

PseAAC and PseAAC using SVM models based on our dataset. This can evaluate the impact of these two feature extraction methods on the prediction model. The specific results are shown in Table 4.

From Table 4, we can see that Hybrid PseAAC outperforms PseAAC in all evaluation metrics. Hybrid PseAAC can more accurately depict the complex characteristics of protein sequences. We used Hybrid PseAAC features and encoded them into 60-dimensional vectors. Mathematically, Hybrid PseAAC is usually represented by the following vector:

$$\left[\frac{f_1}{N}, \frac{f_2}{N}, \ldots, \frac{f_{20}}{N}, P_1, P_2, \ldots, P_L\right] \tag{4}$$

Among them, f_i represents the frequency of the i^{th} type of amino acid in the protein sequence. N is the total length of the protein sequence. $P_1, P_2, \ldots, P_L$ represent additional protein features, such as physicochemical properties of amino acids, secondary structure information of proteins, evolutionary information of the sequence, etc. L is the number of additional features. The first 20 elements represent standard AAC, which indicates the relative frequency of the 20 standard amino acids in the protein sequence. The subsequent elements $P_1, P_2, \ldots, P_L$ are additional features that can be determined based on specific research objectives and available data.

d. *k*-mer

It captures specific patterns and features of a sequence by extracting all possible consecutive amino acid subsequences of length k in the sequence. This method can reveal important local information in protein sequences, which reflects the structural and functional characteristics of the sequence [42]. By analyzing the k-mer composition in a protein sequence, key local features of the sequence can be extracted. The vital local features include patterns of amino acid alignment in specific regions of the sequence, which may be associated with specific functional or structural properties of the protein. The k-mer generates a 21-dimensional vector for each peptide sequence in this paper. Assuming a sequence S of length N, the k-mer feature extraction can be expressed by the following formula:

$$k\text{-mer}(S,k) = \{S[i : i+k] \,|\, i = 1, 2, \ldots, N-k+1\} \tag{5}$$

$S[i : i+k]$ represents a subsequence of length k starting from position i of sequence S. N is the total length of sequence S. The result is a set of all possible k-mer subsequences.

e. Quasi-Sequence-Order (QSOrder)

QSOrder feature is a method that integrates the composition and physicochemical properties, along with the relative positions of amino acids within a sequence, which is used for in-depth analysis of protein sequences. Firstly, QSOrder provides fundamental chemical characteristics for protein sequences by calculating the frequency of appearance of various amino acids. Subsequently, QSOrder further analyzed the arrangement order of amino acids in the sequence to reveal the functional and structural associations of specific regions in the sequence. An important feature of QSOrder is the consideration of the relative positions between amino acids in the sequence, such as the distance between two specific amino acids, which helps to reveal global properties and potential functional regions in the sequence. In this paper, QSQorder feature encoding generated 56-dimensional vectors. Assuming the protein sequence is composed of N amino acids, each of which can be any of the 20 standard amino acids A_i. For each amino acid, its composition ratio C_i in the sequence is

$$C_i = \frac{f_i}{N} \tag{6}$$

Among them, f_i is the number of times amino acid A_i appears in the sequence. Sequential features involve calculating the relative positional information of amino acid pairs in a sequence. For example, a feature Q_d can be defined to represent the relationship between two amino acids in a sequence separated by d amino acids:

$$Q_d = \sum_{i=1}^{N-d} \delta\left(A_i, A_{i+d}\right) \tag{7}$$

f. *K*-Spaced Amino Acid Pairs (KSC)

KSC is a method that focuses on the deep characteristics and pattern capture of protein sequences, achieved by considering amino acid pairs at specific intervals within the sequence [43]. In this paper, KSC generated a 20-dimensional vector for each peptide sequence. For example, when K is set to 2, the analysis will involve the relationship between each amino acid and the third amino acid after it in the sequence. This analysis of spaced amino acids helps to reveal key features in protein sequences, such as local structure and functional relevance. In this

Table 4 Comparison of the impact of hybrid PseAAC and PseAAC feature extraction methods on the prediction model

Model	ACC	Se	Sp	F1 score	AUC	MCC
PseAAC	0.8894	0.8762	0.9032	0.8900	0.9677	0.7793
Hybrid PseAAC	0.9000	0.8969	0.9032	0.9015	0.9694	0.8000

paper, we chose a K value of 3. Assuming a protein sequence consists of N amino acids, each of which can be any of the 20 standard amino acids. The KSC feature for a specific interval K can be calculated using the following formula

$$\mathrm{KSC}_K\left(A_i, A_j\right) = \sum_{n=1}^{N-K} \delta\left(A_n, A_i\right) \times \delta\left(A_{n+K}, A_j\right) \tag{8}$$

A_i and A_j are types of amino acids in the sequence, $\delta(x, y)$ is an indicator function that takes the value 1 when $x = y$ and 0 otherwise.

$\mathrm{KSC}_K\left(A_i, A_j\right)$ represents the count of the amino acid pair $\left(A_i, A_j\right)$ at an interval of k in the sequence.

2.2.1.2 Deep Features

a. ProtT5

ProtT5 is a deep learning-based language model specifically developed for the analysis of protein sequences. It is built on the T5 (Text-to-Text Transfer Transformer) model, a sequence-to-sequence model using a transformer architecture. ProtT5 was initially designed for natural language processing and later applied similar architectures to bioinformatics to understand and predict protein behavior and function. ProtT5 can capture contextual information about each residue in a sequence. ProtT5 is based on a transformer model; it can learn deep feature representations of protein sequences, including the physical and chemical properties, spatial structure, and behavior of amino acids in different biological environments [24]. ProtT5 is pretrained on a large number of protein sequences, allowing it to understand a wide range of protein languages, which helps identify new or unannotated CPPs. ProtT5 can be fine-tuned according to specific tasks, which means it can be optimized for specific datasets of CPPs to improve prediction accuracy. In this paper, ProtT5 generated a 1024-dimensional vector for each peptide sequence.

The core of ProtT5 relies on the self-attention mechanism of the Transformer. Its mathematical expression is as follows:

$$\mathrm{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d_{\boldsymbol{K}}}}\right)\boldsymbol{V} \tag{9}$$

$\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are query, key, and value matrices, which are transformations of amino acid coding. $d_{\boldsymbol{K}}$ is the dimension of the key vector, which is used to adjust the size of the dot product.

In the application of CPP recognition, ProtT5 can be fine-tuned to optimize the following objective functions:

$$L_{\mathrm{CE}} = -\sum_{i=1}^{N'} y_i \cdot \log\left(p_i\right) + \left(1 - y_i\right) \cdot \log\left(1 - p_i\right) \tag{10}$$

Among them, L_{CE} is the cross entropy loss function, N' is the number of training samples, y_i is the true label of the i-th sample, and p_i is the probability that the model predicts the i-th sample to be CPP.

b. Evolutionary Scale Modeling (ESM-2)

ESM-2 is an advanced bioinformatics method that incorporates deep learning techniques and evolutionary biology principles. It is designed for protein sequence analysis. Although structurally similar to ProtT5, ESM-2 focuses on analyzing protein evolution and the diversity of protein functions [44].

The core of ESM-2 lies in analyzing many protein sequences and their evolutionary relationships to extract deep evolutionary features of protein families. ESM-2 assists in predicting the three-dimensional structure of proteins and identifying the potential functions of unknown proteins by capturing complex patterns and relationships in sequences. Meanwhile, it also reveals the trends and pathways of protein family evolution over time. In this paper, ESM-2 generated a 1024-dimensional vector for each peptide sequence.

2.2.2 Methods

2.2.2.1 Machine Learning

Machine learning techniques have been widely used in the prediction studies of CPPs. With the continuous progress of machine learning algorithms, these technologies have achieved certain results in improving the accuracy and speed of protein site prediction. This not only accelerates drug development but also deepens the understanding of protein structure–function interactions. In this paper, we adopted an ensemble method that combines tree-based models and SVM to predict CPPs effectively.

a. Support Vector Machine (SVM)

SVM is a supervised learning model [45] which has been widely used to predict CPPs. The core of SVM lies in its ability to find an optimal hyperplane. This hyperplane is defined in a high-dimensional feature space to maximize the spacing between CPP data points of different categories [46]. In the study, we conducted parameter optimization through grid search. SVM used the radial basis function (RBF) as the kernel function and set the regularization parameter C to 1.0.

Assuming there is a training set $\left\{\left(\boldsymbol{x}_1, y_1\right), \left(\boldsymbol{x}_2, y_2\right), \ldots, \left(\boldsymbol{x}_n, y_n\right)\right\}$, where $\boldsymbol{x}_i$ is the feature

vector and y_i is the class label for each sample (usually 1 or − 1). The mathematical expression of the SVM model is

$$\min_{\boldsymbol{w},\boldsymbol{b}} \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{11}$$

where $\boldsymbol{w}$ is the normal vector of the hyperplane, and $\boldsymbol{b}$ is the intercept term. The purpose of this function is to minimize the length of the hyperplane normal vector, which is equivalent to maximizing the interval between two categories. The constraint is $y_i\left(\boldsymbol{w}\cdot\boldsymbol{x}_i+\boldsymbol{b}\right)\geq 1, \forall i$, where $\boldsymbol{x}_i$ is the data point and y_i is the corresponding category label.

b. Categorical Boosting (CatBoost)

CatBoost is an efficient machine learning algorithm for handling classification and regression tasks. As an algorithm based on the Gradient Boosting Decision Tree (GBDT), CatBoost performs several innovative optimizations to enhance performance and reduce the risk of overfitting. CatBoost gradually constructs a decision tree through iterative methods. Each new tree will correct the prediction error of the previous tree to reduce the overall loss function gradually [47].

CatBoost is unique in that it introduces two techniques: Ordered Boosting and Decentralized Target Statistics. Ordered Boosting helps the algorithm to take into account the time series of the data when constructing the model, thus effectively reducing the possibility of overfitting. Decentralized Target Statistics improves the accuracy of the model in dealing with classification problems by reducing the bias in the computation. In the study, we used grid search for parameter optimization. The learning rate of CatBoost is set to 0.03, with 500 iterations and a tree depth of 6.

The core of CatBoost is the gradient boosting framework, which aims to minimize the following loss functions:

$$L=\sum_{i=1}^{N'} l\left(y_i,f\left(\boldsymbol{x}_i\right)\right)+\sum_{t=1}^{t}\Omega\left(f_t\right) \tag{12}$$

In this formula, N' is the number of training samples, l is the loss function, and y_i is the target value of sample i, $f\left(\boldsymbol{x}_i\right)$ is the predicted value of the model, f_t is the t^{th} tree, $\Omega\left(f_t\right)$ is a regularization term used to reduce model complexity.

c. Logistic Regression (LR)

Logistic regression is a statistical technique used for model construction and analysis [48], especially in exploring the relationship between independent and binary categorical dependent variables. LR's simplicity and high efficiency make it an ideal choice for the final output layer of many stacked models.

LR can integrate the results identified by various basic models in the previous hierarchy and evaluate their importance for the final prediction. In the study, logistic regression utilized L2 regularization and the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) optimization algorithm.

The core of the LR model is the S-shaped logistic function, which is used to estimate probabilities:

$$P(Y=1|\boldsymbol{X})=\frac{1}{1+e^{-\left(\beta_0+\beta_1X_1+\beta_2X_2+\cdots+\beta_nX_n\right)}} \tag{13}$$

In this equation, $P(Y=1|\boldsymbol{X})$ is the probability that the peptide sequence is CPPs given the input feature $\boldsymbol{X}$. $\beta_0,\beta_1,\ldots,\beta_n$ are model parameters, including the intercept term β_0 and coefficients β_1 to β_n for each feature. $X_1,X_2,\ldots,X_n$ are characteristics of peptide sequences.

2.2.2.2 Deep Learning

Deep learning technology has not only achieved significant success in fields such as computer vision (CV) and natural language processing (NLP), but also plays an increasingly vital role in the biomedical field [49]. Deep learning plays a significant role in biomedicine, including disease diagnosis, drug discovery, gene sequence analysis, and protein structure prediction. In the prediction of CPPs, deep learning can extract key features from protein sequences. These features are critical for understanding and predicting peptide penetration.

CNN is excellent at recognizing local patterns in protein sequences. For example, CNN can recognize specific amino acid arrangements in sequences. TextCNN is a derivative of CNN designed for text and sequence data. Unlike traditional CNN, which mainly processes image data, TextCNN uses one-dimensional convolution to analyze sequence data, making it ideal for text processing and biological sequence analysis. TextCNN can capture global and local dependencies in protein sequence analysis, which helps to understand the overall context of the sequence deeply. In contrast to regular CNN, TextCNN's filters slide in a single dimension of the sequence rather than on a two-dimensional plane, making TextCNN more effective and suitable for processing sequence data.

a. Convolutional Neural Networks (CNN)

CNN can automatically extract key features from sequence data through its convolutional layers. These include recognizing local patterns of amino acids and more complex spatial structures, which are key factors in understanding the functionality of CPPs. In addition, CNN excels in dealing with nonlinear relationships in biological sequence data, which allows it to capture complex patterns efficiently.

CNN can deeply analyze the structural and functional characteristics of CPPs. Its robust feature extraction ability enables the model to capture critical features of CPPs more sensitively, thereby improving the accuracy of recognition [50]. In CNN, we first added a convolutional layer with 32 filters and a kernel size of 3 and used the ReLU activation function. Subsequently, we added a pooling layer with a pooling size of 2. Next, we used a Flatten layer to flatten the feature map and added a Dropout layer with a loss rate 0.2 to prevent overfitting. Finally, we used the Softmax activation function through an output layer for classification.

b. TextCNN

TextCNN is a variant of CNN designed specifically for text data, but it also demonstrates excellent performance in CPP prediction tasks [51]. The convolutional layers of TextCNN can capture local features in sequences, such as specific combinations and patterns of amino acids. Each convolutional layer contains multiple convolutional kernels. These kernels slide over the sequence to capture patterns of different lengths. This capability allows TextCNN to recognize short sequence structures and also detect more extended sequence patterns, thereby providing a comprehensive analysis of the diversity of CPPs.

Compared to traditional CNN, TextCNN focuses more on capturing local patterns of sequences. It is designed and optimized specifically for sequence data and has shown higher efficiency and accuracy, especially when dealing with shorter biological sequences [52]. However, traditional CNN may focus more on capturing a more comprehensive range of spatial features. In TextCNN, we selected three convolutional layers. Each layer uses filters of sizes 3, 4, and 5, and a pooling layer of size 2 follows each convolutional layer. Each convolutional layer uses 36 filters, and the activation function is ReLU. Next is a fully connected layer that also uses the ReLU activation function. Finally, through an output layer, we used Softmax for classification.

2.2.2.3 Construction of EnDM-CPP When designing and implementing the EnDM-CPP model, our goal is to improve the predictive performance of CPPs by integrating learning strategies and building an efficient prediction framework that can integrate different data representations and algorithm advantages. This framework uses the capabilities of four basic models, SVM, CNN, TextCNN, and CatBoost, to ensure that the model can learn the complex biological characteristics of CPPs from multiple dimensions.

After an in-depth analysis of previous studies on CPP prediction, we found that although StackCPPred enhances the prediction performance by integrating different machine learning models, it still lacks the ability to extract deep biological features. Although TargetCPP focuses on predicting specific targets, its comprehensiveness and generalization ability are insufficient when faced with a wider range of CPP characteristics. Although MLCPP and its upgraded version, MLCPP 2.0, provide a solid underlying framework for CPP prediction, they lack the sophistication of deep learning methods in feature extraction and long sequence data processing. Meanwhile, SiameseCPP performs well in learning inter-sample similarity through its twin network architecture, which effectively predicts certain CPP types. However, it faces challenges in generalizing to new CPP types.

EnDM-CPP successfully overcomes the limitations of these studies while absorbing their advantages and constructing a unique prediction framework. By integrating multiple advanced machine learning and deep learning techniques, EnDM-CPP not only improves the prediction accuracy of CPPs but also enhances the model's capability in processing complex bioinformatics data, especially in deep biological feature extraction and lengthy sequence data processing. In addition, EnDM-CPP improves the model's generalization ability, enabling it to identify and predict unseen types of CPPs effectively.

SVM is introduced based on its ability to efficiently find the optimal interface between different classes in a high-dimensional space, which is crucial for processing complex bioinformatics data. Our training dataset contains 756 samples, each with 256 features. After the SVM model training is completed, the model outputs the probability of two categories for each input sample; thus, the output dimension is (756, 2). The CatBoost model is used because it can handle many categorical feature data, which is particularly effective in handling bioinformatics data such as amino acid sequences in CPPs. At the output layer, CatBoost also provides probability predictions for two categories with the same output dimension as SVM.

We chose CNN and TextCNN as models for the deep learning part. CNN can capture local patterns in CPP sequences, such as specific amino acid combinations and sequence features. In this paper, the deep features are obtained by fusing ESM-2 and ProtT5. The dimensions of each feature are (756, 42, 1024), and the fused feature dimensions are (756, 42, 2048), where 42 represents the sequence length, and 2048 is the feature vector dimension of each sequence element. The convolution formula of CNN is expressed as

$$\boldsymbol{Y} = f(\boldsymbol{WX} + \boldsymbol{b}) \tag{14}$$

Here $\boldsymbol{W}$ is the weight of the convolution, $\boldsymbol{b}$ is the bias term, and f is the activation function ReLU. Fill with "same", so the length of the output sequence remains unchanged and the dimension becomes (756, 42, 32). After passing through the maximum pooling layer, its formula is expressed as

$$P_i = \max_{j=1,2} X_{i+j} \tag{15}$$

As a result of pooling over the length of the sequence, the length of the sequence is halved, and the dimension becomes (756, 21, 32). After the flattening layer, the dimension becomes (756, 672). To prevent overfitting, we added a Dropout layer with a parameter set to 0.2. In the final fully connected layer, the dimension is (756, 2) because the output represents the probabilities of two categories.

The input feature dimension of TextCNN is the same as CNN, which is (756, 42, 2048). Then, it undergoes three rounds of convolution and max pooling. The first round of the convolution branch uses 3-sized convolutional kernels with "same" padding and a step size of 1. The dimensionality of the sequence remains unchanged, but the feature dimensions are changed to the number of convolutional kernels, i.e., (756, 42, 36). After pooling, the maximum pooling halves the sequence length, resulting in dimensions of (756, 21, 36). The second convolutional branch uses 4-sized convolutional kernels with the same convolutional operations and dimension changes. The output dimension is (756, 42, 36), which becomes (756, 21, 36) after pooling. The third convolutional branch uses 5-sized convolutional kernels, and the convolutional and pooling operations are the same as those of the first two branches. The final dimension is (756, 21, 36). Then, the outputs of the three branches are combined in the merged layer, resulting in a dimension of (756, 21, 108), where 108 is the superposition of three 36-dimensional features. After the flattening layer, the dimension becomes (756, 2268). The calculation formula for its connecting layer is as follows

$$\boldsymbol{Y} = f(\boldsymbol{WX} + \boldsymbol{b}) \tag{16}$$

Among them, $\boldsymbol{W}$ is the weight matrix, $\boldsymbol{b}$ is the paranoid vector, f is the ReLU activation function, and the output dimension (756, 32). Finally, a fully connected layer is used as the output layer, and the final output dimension is (756, 2).

We used LR as the final decision layer. This is not only because logistic regression can efficiently handle classification problems but also because it can integrate the features generated by the aforementioned basic models and evaluate their importance. If the output of each base model is represented as a feature vector, the stacked model can be represented by the following formula.

Set $\boldsymbol{x}$ as input CPP sequence features. $f_{\mathrm{SVM}}(\boldsymbol{x})$, $f_{\mathrm{CatBoost}}(\boldsymbol{x})$, $f_{\mathrm{CNN}}(\boldsymbol{x})$ and $f_{\mathrm{TextCNN}}(\boldsymbol{x})$ will form a new feature vector $\left[f_{\mathrm{SVM}}(\boldsymbol{x}), f_{\mathrm{CatBoost}}(\boldsymbol{x}), f_{\mathrm{CNN}}(\boldsymbol{x}), f_{\mathrm{TextCNN}}(\boldsymbol{x})\right]$. Finally, the LR model f_{LR} uses $\boldsymbol{X}_{\mathrm{new}}$ as input to make the final classification decision:

$$P_{\mathrm{final}} = f_{\mathrm{LR}}\left(\boldsymbol{X}_{\mathrm{new}}\right) \tag{17}$$

Here $f_{\mathrm{LR}}\left(\boldsymbol{X}_{\mathrm{new}}\right)$ is the final predicted output, representing the probability that the given sequence is CPPs. By using logistic regression layers for feature fusion, the advantages of various models are effectively combined, thereby improving the accuracy of CPP prediction.

In order to elaborate on the construction and operation mechanism of the EnDM-CPP model, we provide a model architecture diagram of EnDM-CPP, as shown in Fig. 2. The diagram introduces the machine learning features and maps the flow of the deep features. In addition, it also elaborates on the stacking and fusion methods of various basic models, as well as how logistic regression gathers the outputs of these models to form the final prediction results.

2.2.3 Evaluation

Evaluation metrics can be used to measure the model performance of a binary classification problem, i.e., to distinguish whether it is a CPP or not. In order to evaluate EnDM-CPP, the following common binary classification evaluation metrics are used in this article, including Sensitivity (Se), Specificity (Sp), Matthew's Correlation Coefficient (MCC), Accuracy (ACC), Average Precision (AP), F1 score, and Area Under the ROC Curve (AUC). In this paper, to ensure the reliability of EnDM-CPP, we used a fivefold cross-validation and independent test dataset for testing. fivefold cross-validation can better evaluate the model's generalization ability on unseen data. fivefold cross-validation means that the dataset is divided into five parts; four parts are used to train the model each time, and the remaining one is used for testing. This process is repeated five times to ensure that each part of the data is used as a test. This method helps to reduce the effects of serendipity and overfitting.

$$\text{Sensitivity} = \frac{n_{\mathrm{TP}}}{n_{\mathrm{TP}} + n_{\mathrm{FN}}} \tag{18}$$

The ratio of actual CPP samples is accurately identified using the sensitivity measurement model, i.e., among all the samples that are truly CPPs, how many are correctly recognized as CPPs? The higher the sensitivity, the fewer CPP samples are missed in the model detection.

$$\text{Specificity} = \frac{n_{\mathrm{TN}}}{n_{\mathrm{FP}} + n_{\mathrm{TN}}} \tag{19}$$

The ratio of actual non-CPP samples correctly identified by the specificity measurement model, i.e., among all the samples that are truly non-CPPs, how many are correctly recognized as non-CPPs? The higher the specificity, the less likely the model misclassifies non-CPP samples as CPPs.

$$\text{MCC} = \frac{n_{\mathrm{TP}} \times n_{\mathrm{TN}} - n_{\mathrm{FP}} \times n_{\mathrm{FN}}}{\sqrt{\left(n_{\mathrm{TP}} + n_{\mathrm{FN}}\right)\left(n_{\mathrm{TN}} + n_{\mathrm{FP}}\right)\left(n_{\mathrm{TP}} + n_{\mathrm{FP}}\right) + \left(n_{\mathrm{TN}} + n_{\mathrm{FN}}\right)}} \tag{20}$$

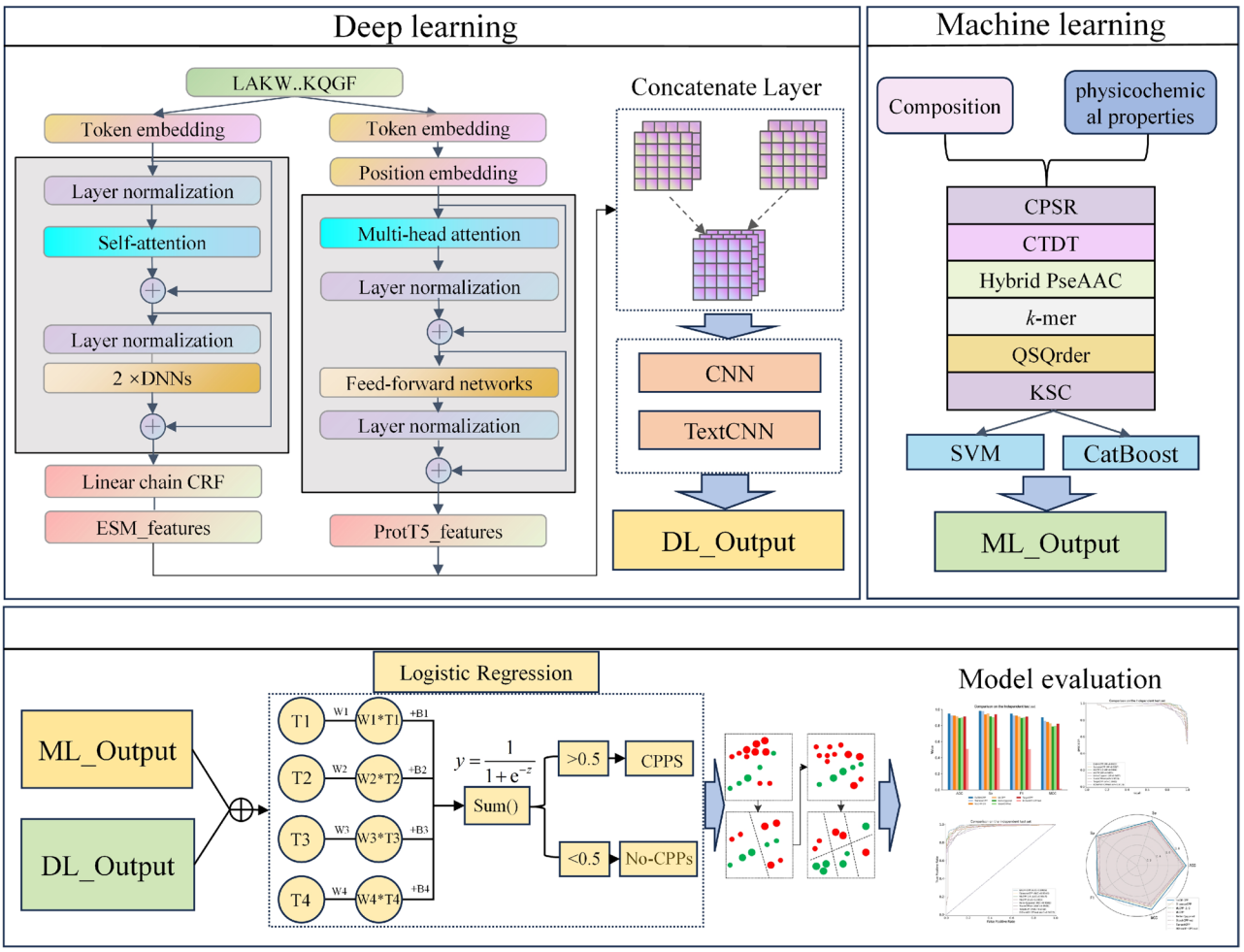


Fig. 2 The architecture of the EnDM-CPP stacked integrated learning model: (1) Use ESM-2 and ProtT5 features based on the transformer model along with CNN and TextCNN for deep learning prediction. (2) Use six traditional feature extraction methods, along with SVM and CatBoost, for machine learning prediction. (3) Merge these prediction results and use LR as the final decision-making mechanism for performance evaluation and model comparison

MCC comprehensively considers n_{TP}, n_{TN}, n_{FP}, and n_{FN}, which can fully reflect the performance of the classifier. The value range of MCC is [− 1,1]. Values closer to 1 indicate better classification performance. A value of 0 indicates that the classification performance of the model is the same as that of random classification. A value of − 1 indicates that the classification results of the model are completely opposite to reality.

$$\text{Accuracy} = \frac{n_{TP} + n_{TN}}{n_{TP} + n_{FP} + n_{FN} + n_{TN}} \tag{21}$$

Accuracy represents the proportion of correctly predicted samples by the model to the total samples.

$$\text{AP} = \int_0^1 p(r)dr \tag{22}$$

Average precision (AP) is the area under the PR curve. It combines the model's accuracy and recall performance at all thresholds. The higher the AP, the higher the model's accuracy and recall at different thresholds. AP is a comprehensive indicator for measuring model performance.

$$F_1 = \frac{2 \times n_{TP}}{2 \times n_{TP} + n_{FP} + n_{FN}} \tag{23}$$

The F1 score is the harmonic mean of precision and sensitivity. It is a good indicator of how well a model balances precision and sensitivity. The higher the F1 score, the better the model balances precision and sensitivity.

$$\mathrm{AUPRC} = \int_0^1 p(r)dr \tag{24}$$

AUPRC measures the relationship between precision and recall at different thresholds. The value of AUPRC is obtained by calculating the area under the PR curve. A higher AUPRC value indicates that the model can maintain high precision and recall when dealing with imbalanced datasets.

n_{TP}, n_{TN}, n_{FP}, and n_{FN}, respectively, represent the number of cases correctly identifying CPPs as CPPs, the number of cases correctly identifying non-CPPs as non-CPPs, the number of cases incorrectly identifying non-CPPs as CPPs, and the number of cases incorrectly identifying CPPs as non CPPs.

3 Results

3.1 Selection of Features in Machine Learning

In order to construct a model that can efficiently and accurately distinguish CPPs, it is crucial to select an appropriate feature extraction method. In the machine learning part of the stacked model in this paper, we adopted a combination of six different feature encoding methods. The purpose of selecting these methods is to capture the critical features of CPPs comprehensively. These key features include the distribution pattern, arrangement order, physicochemical properties, frequency of occurrence, composition, and spatial structure of amino acids. By fusing the numerical vectors of these six feature encoding methods, we have established a comprehensive feature space that includes CPSR, CTDC, Hybrid PseAAC, *k*-mer, QSOrder, and KSC. We named this feature set "ML_ALL".

In the classification task, we chose SVM as the classifier. To ensure the robustness of the model evaluation, we used a five-fold cross-validation to evaluate the effectiveness of various single-feature encoding methods and their combinations in the prediction of CPPs. The prediction results are summarized in Table 5, which demonstrates the performance of different single and fusion feature coding strategies.

In addition, we also used ROC curves and histograms to evaluate the impact of different feature extraction algorithms on the stability of the prediction model. Figure 5 shows the ROC curves and histograms on the validation dataset, comparing the performance of more than twenty different single and fused feature extraction methods.

The data in Table 5 show that different feature extraction algorithms produce varying predictive results across various evaluation metrics.

From data analysis in Table 5, it can be seen that in SVM, Hybrid PseAAC performs the best in four evaluation metrics, namely ACC, SP, F1 score, and MCC, when using a single feature extraction method. These values are 90.08%, 91.05%, 89.93% and 80.17% respectively. Meanwhile, *k*-mer and CTDC both achieved their highest values on Se, reaching 89.89%. On AP, CPSR performed the best at 96.7%. In Extra Trees, for a single feature extraction method, CPSR performs best on ACC, Se, F1 score, and MCC metrics with 90.87%, 88.83%, 90.64%, and 81.81%, respectively. QSQ performs optimally on Sp and AP with 94.74% and 97.27%.

The above results reveal that different feature extraction methods have performance differences on different datasets. However, compared to single feature extraction methods, fusion feature extraction methods perform better in predicting the results. ML_ALL outperforms the single features in most evaluation metrics (including ACC, Se, AUC, AP, etc.) in both SVM and Extra Tree. The Sp of ML_ALL is tied with Hybrid PseAAC for the highest performance in the SVM model and is only slightly lower than QSQ in Extra Tree. In summary, combining CPSR, CTDC, Hybrid PseAAC, *k*-mer, QSOrder, and KSC will produce the best results.

As shown in Fig. 3, ML_ALL has the largest coverage area under the ROC curve and obtains the highest values on both SVM and ERT. The performance of ML_ALL on the PR curve is equally outstanding. Moreover, the histograms also display that ML_ALL feature coding outperforms all individual coding methods on both ACC and MCC.

In addition, Shapley Additive Explanation (SHAP) was applied to elucidate the importance of the fused features in the model. SHAP provides a way to quantify the contribution of each feature to the model prediction [53]. This holds significant reference value for analyzing the prediction mechanisms of complex models, particularly in our research, where understanding the impact of each feature on the model output is a crucial step.

Figure 4a displays the SHAP value distribution for the top 20 features, where red dots represent positive correlation and blue dots represent negative correlation. The larger the dot size, the more significant the feature's impact on the model. Through this graph, we observe that the top 20 features contain both features from contrastive learning and features from pre-trained models. Figure 4b provides the average SHAP values of the top 20 features through a bar chart. The length of each bar represents the absolute magnitude of the average SHAP value of the corresponding feature, which

Table 5 Prediction results of different machine learning feature encodings on the validation dataset

Model	Features	ACC	Se	Sp	AP	F1 score	MCC
SVM	ML_ALL	0.9074	0.9043	0.9105	0.9691	0.9067	0.8148
	Hybrid PseAAC	0.9008	0.8910	0.9105	0.9642	0.8993	0.8017
	CPSR	0.8981	0.8963	0.9000	0.9670	0.8975	0.7963
	k-mer	0.8981	0.8989	0.8974	0.9631	0.8977	0.7963
	CTDC	0.8968	0.8989	0.8947	0.9608	0.8966	0.7937
	KSC	0.8955	0.8910	0.9000	0.9606	0.8945	0.7910
	APAAC	0.8955	0.8883	0.9026	0.9609	0.8942	0.7911
	QSOrder	0.8942	0.8936	0.8947	0.9611	0.8936	0.7884
	CKSAAP	0.8889	0.8484	0.9289	0.9598	0.8837	0.7801
	CTDT	0.8677	0.8644	0.8711	0.9438	0.8667	0.7354
	DPC	0.8664	0.8218	0.9105	0.9496	0.8595	0.7355
	EAAC	0.8664	0.8324	0.9000	0.9436	0.8611	0.7343
	CTD	0.8624	0.8511	0.8737	0.9477	0.8602	0.7250
	QSQ	0.8505	0.8324	0.8684	0.9415	0.8471	0.7014
	CTDD	0.8492	0.8271	0.8711	0.9190	0.8451	0.699
	DDE	0.8479	0.8112	0.8842	0.9381	0.8414	0.6974
	CKSAAGP	0.8399	0.8032	0.8763	0.9223	0.8331	0.6815
	GDPC	0.8373	0.7979	0.8763	0.9154	0.8299	0.6765
	GAAC	0.8333	0.7926	0.8737	0.9146	0.8255	0.6686
	BINARY	0.8294	0.7766	0.8816	0.9237	0.8191	0.6621
	EGAAC	0.8294	0.8085	0.8500	0.9140	0.8250	0.6592
	CTriad	0.8214	0.7793	0.8632	0.9199	0.8128	0.6449
	KSCTriad	0.8161	0.7739	0.8579	0.9191	0.8072	0.6342
	BLOSUM62	0.8122	0.7553	0.8684	0.8987	0.8000	0.6280
	GTPC	0.8069	0.7553	0.8579	0.8998	0.7955	0.6167
	ZSCALE	0.7910	0.7420	0.8395	0.8873	0.7793	0.5844
Extra Trees	ML_ALL	0.9101	0.8910	0.9289	0.9733	0.9079	0.8206
	CPSR	0.9087	0.8883	0.9289	0.9726	0.9064	0.8181
	Hybrid PseAAC	0.9048	0.875	0.9342	0.9713	0.9014	0.8108
	QSQ	0.9034	0.859	0.9474	0.9727	0.8985	0.8099
	CTDC	0.8995	0.8723	0.9263	0.9659	0.8962	0.8000
	k-mer	0.8942	0.8484	0.9395	0.9721	0.8886	0.7915
	KSC	0.8942	0.8617	0.9263	0.9651	0.8901	0.7899
	EAAC	0.8942	0.8431	0.9447	0.9646	0.8880	0.7923
	DDE	0.8902	0.8856	0.8947	0.9594	0.8892	0.7804
	CKSAAP	0.8876	0.9043	0.8711	0.9659	0.8889	0.7756
	QSOrder	0.8836	0.8404	0.9263	0.9674	0.8778	0.7699
	APAAC	0.8836	0.8271	0.9395	0.9645	0.8761	0.7718
	BLOSUM62	0.8783	0.8404	0.9158	0.9471	0.8729	0.7586
	DPC	0.8730	0.8644	0.8816	0.9535	0.8713	0.7461
	CTD	0.8704	0.8351	0.9053	0.9516	0.865	0.7424
	ZSCALE	0.8704	0.8191	0.9211	0.9458	0.8627	0.7444
	CTDT	0.8690	0.8271	0.9105	0.9446	0.8627	0.7405
	GDPC	0.8585	0.8218	0.8947	0.9361	0.8524	0.7187
	CKSAAGP	0.8519	0.8032	0.9000	0.9363	0.8436	0.7068
	EGAAC	0.8505	0.8032	0.8974	0.9325	0.8424	0.7039
	KSCTriad	0.8386	0.8112	0.8658	0.9254	0.8333	0.6781
	CTDD	0.8360	0.7952	0.8763	0.9235	0.8283	0.6739
	CTriad	0.8360	0.8112	0.8605	0.9245	0.8311	0.6726
	GAAC	0.8294	0.7872	0.8711	0.9209	0.8211	0.6608

Table 5 (continued)

Model	Features	ACC	Se	Sp	AP	F1 score	MCC
	GTPC	0.8294	0.7872	0.8711	0.9153	0.8211	0.6608
	BINARY	0.8188	0.8750	0.7632	0.9335	0.8277	0.6419

The combination of CPSR, CTDC, Hybrid PseAAC, *k*-mer, QSOrder, and KSC is called ML_ALL

intuitively reflects the average impact of each feature on the model output.

To comprehensively analyze the importance of features, Fig. 4c shows the SHAP value distribution of the last 20 features on a single sample. Although the SHAP values of these features are relatively small, they still play an important role in specific samples. Figure 4d shows the average SHAP values of the last 20 features through a bar chart. Analyzing these features gives us a more comprehensive understanding of their importance.

We selected features by taking the absolute average of SHAP values and tried other different thresholds. The results show that the top 20 features significantly impact model performance, but other features also play important roles in specific situations. These results highlight the necessity of considering different types of features comprehensively rather than relying solely on a single feature encoding method. The combination of six features provides a wider range of information for protein sequence characterization than a single-feature encoding algorithm, thus demonstrating superior predictive performance.

3.2 Selection of Features in Deep Learning

In this paper, we adopted a fusion of two feature encoding methods in the stacked model's deep learning section: ESM-2 and ProtT5. ESM-2 focuses on capturing changes in proteins during evolution, thus providing important features relevant to biology. ProtT5 captures features in sequences through its bidirectional contextual understanding capability. By combining ESM-2 and ProtT5, we can simultaneously leverage the advantages of both models to capture both sequence evolution and contextual information. In this study, the feature dimension generated by ESM-2 is (946, 40, 1280), while that generated by ProtT5 is (946, 40, 1024). To ensure the effectiveness of the combined features, we reduced the dimension of ESM-2's features to (946, 40, 1024), to maintain consistency with ProtT5's feature dimensions. Subsequently, we fused the two in the second dimension, resulting in the final feature dimension (946, 80, 1024). We named this fused feature set "D_ALL".

We chose CNN as the deep classifier. We compared ESM-2, ProtT5, Word2Vec, Tasks Assessing Protein Embeddings (TAPE), and Binary Protein Fingerprint (BPF) together. We evaluated the effectiveness of various single-feature encoding methods and their combinations in CNN model predictions on the validation set. The prediction results are summarized in Table 6, which demonstrates the performance of different single and fusion feature coding strategies.

From the data analysis in Table 6, it can be seen that among the single feature indicators, ESM-2 achieves the highest values on ACC, SP, AUC, AP, F1 score, and MCC, reaching 91.8%, 95.26%, 97.7%, 97.67%, 91.46%, and 83.79%, respectively. In contrast, ProtT5 performed the best on Se at 88.56%. These results show that ESM-2 and ProtT5 have larger performance improvements compared to other features: ACC improves by 2.12%–4.23%, Se improves by 0.26%–2.66%, Sp improves by 4.21%–6.05%, AUC improves by 1.26%–2.23%, AP improves by 1.29%–1.67%, F1 score improves by 1.97%–4.16%, while MCC improves by 4.4%–8.62%. These data emphasize the advantages of ESM-2 and ProtT5 in feature extraction, which enhance the model's performance on several key performance metrics. However, we also observed that the fused machine learning feature ML_ALL does not perform as well as expected in the deep learning model. Based on this, we decided not to integrate these machine learning features into the deep learning algorithm in the subsequent stacked model design. This decision was based on a careful analysis of the available data and aimed at optimizing the overall performance and accuracy of the model.

Through ablation experiments, compared to a single feature, the fused feature D_ALL showed higher values in all indicators except for Sp, which was on par with ESM-2. Specifically, D_ALL exhibited an ACC of 92.06%, Se of 88.83%, AUC of 97.91%, AP of 98.08%, F1 score of 91.76%, and MCC of 84.29%. This indicates that by merging the features of ESM-2 and ProtT5, we can integrate the advantages of both to further enhance the predictive ability and generalization performance of the model across multiple dimensions. Additionally, we conducted a comprehensive analysis by plotting ROC curves and histograms.

The bar chart in Fig. 5 directly compares the performance of several single features with our integrated feature set D_ALL on CNN. As shown, D_ALL outperforms other single features in all evaluation metrics. Among the single features, ProtT5 and ESM-2 show better results. Meanwhile, the ROC curve graph shows that D_ALL has the largest area under the curve, demonstrating its superior performance.

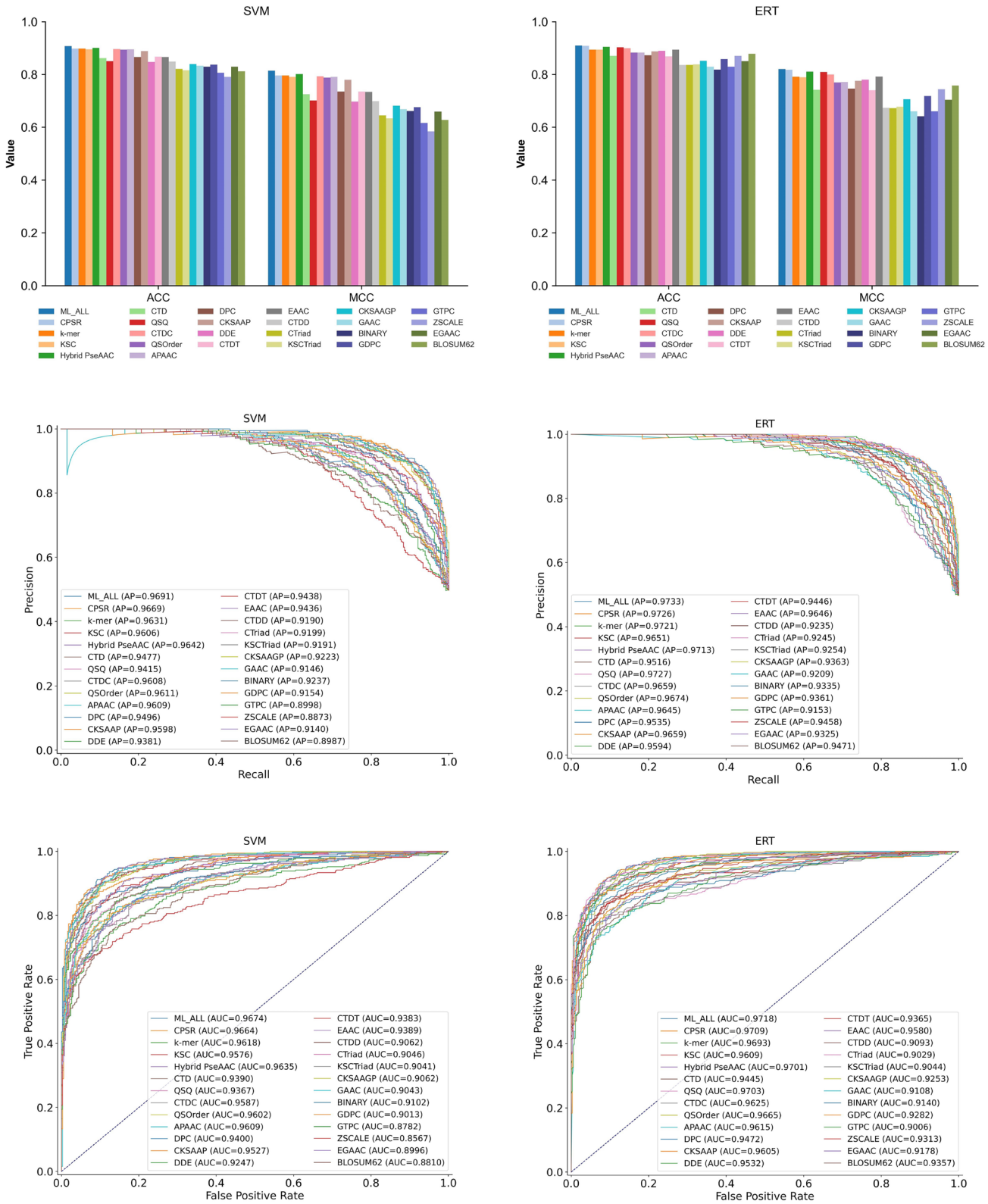


Fig. 3 ROC and PR curves and bar charts for different feature encodings

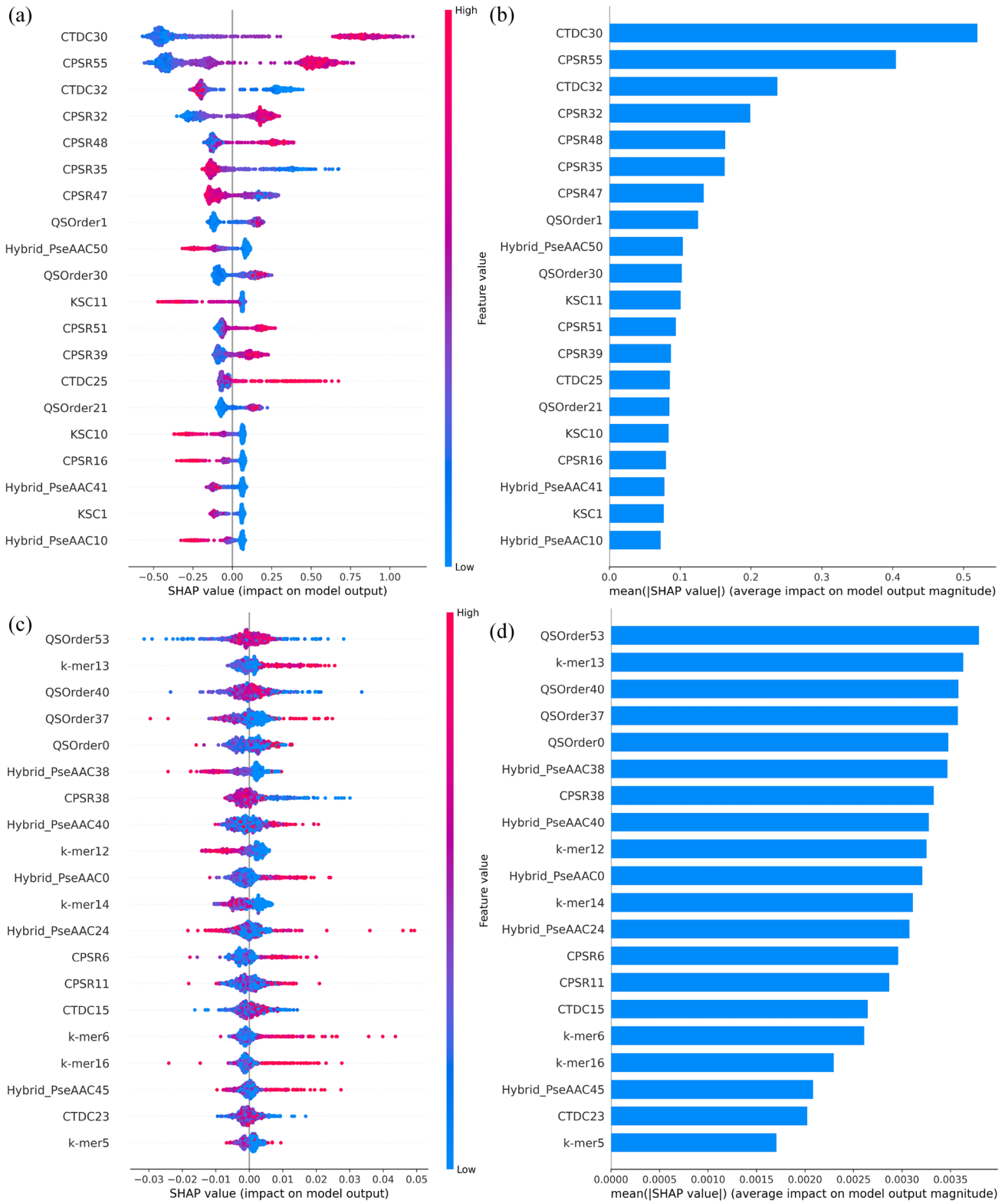


Fig. 4 Important features of SHAP interpretation. **a** The SHAP value distribution of the 20 most important features. **b** The importance bar chart of the top 20 features. **c** The SHAP value distribution of the last 20 features. **d** The importance bar chart of the last 20 features

Table 6 Prediction results of different deep learning feature encodings

Model	Features	ACC	Se	Sp	AUC	AP	F1 score	MCC
CNN	D_ALL	0.9206	0.8883	0.9526	0.9791	0.9808	0.9176	0.8429
	ESM-2	0.9180	0.8830	0.9526	0.9770	0.9767	0.9146	0.8379
	ProtT5	0.9114	0.8856	0.9368	0.9663	0.9715	0.9086	0.8237
	Word2Vec	0.8968	0.8830	0.9105	0.9608	0.9638	0.8949	0.7939
	BPF	0.8902	0.8750	0.9053	0.9537	0.9567	0.8880	0.7807
	ML_ALL	0.8889	0.8830	0.8947	0.9658	0.9674	0.8877	0.7778
	TAPE	0.8757	0.8590	0.8921	0.9536	0.9600	0.8730	0.7517

The combination of CPSR, CTDC, Hybrid PseAAC, *k*-mer, QSOrder, and KSC is called ML_ALL, and the combination of ESM-2 and ProtT5 is called D_ALL

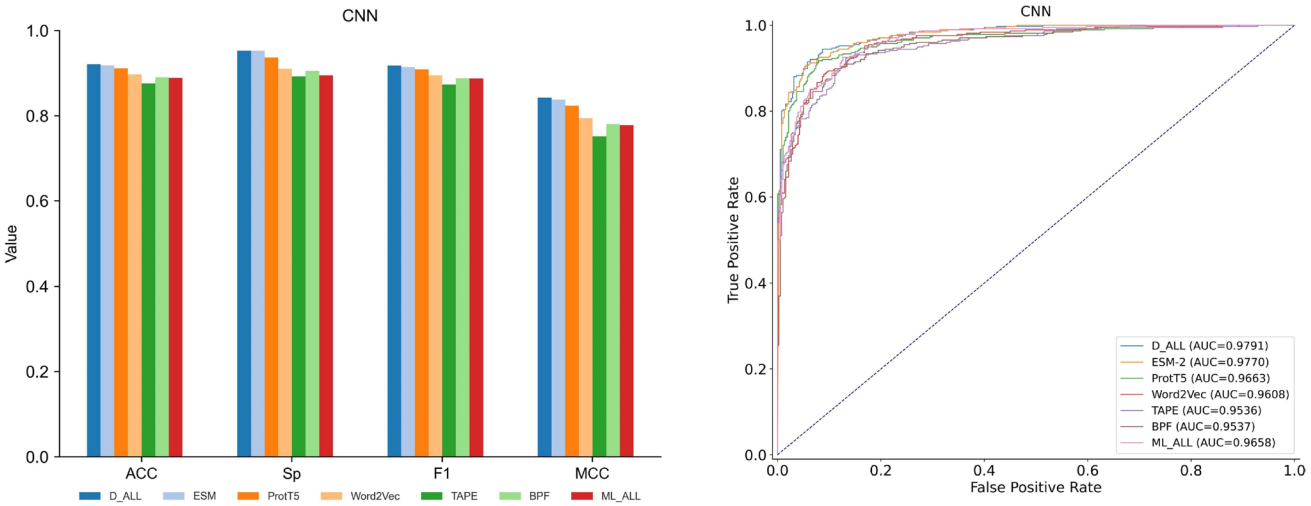


Fig. 5 A comparison of different deep features in the CNN model is presented through bar charts and ROC curves

3.3 Comparison Between Different Classification Algorithms

In the paper, we compared 15 independent classifiers. These classifiers include eight machine learning models and seven deep learning models. The machine learning models are CatBoost, LightGBM, SVM, XGBoost, Random Forest, Extreme Learning Machine (ELM), LR, and AdaBoost. The deep learning models are TextCNN, CNN, Multi-Layer Perceptron (MLP), TextRNN, Long Short-Term Memory Network (LSTM), Gated Recurrent Unit (GRU), and Recurrent Neural Network (RNN). We explain the criteria for selecting these classifiers for comparison and provide an overview of the advantages and disadvantages of each classifier. Detailed content is available in the supplementary document. In addition, we also explored four integrated classifiers based on different combinations of models. These integrated models are: (1) an integrated model based on TextCNN, CNN, SVM, and CatBoost; (2) an integrated model based on TextCNN, CNN, SVM, CatBoost, and LightGBM; (3) an integrated model based on TextCNN, CNN, CatBoost, and LightGBM; (4) an integrated model based on TextCNN, CNN, SVM, and LightGBM.

In a single machine learning algorithm, we uniformly used ML-ALL. This comprehensive feature strategy provides a broader perspective than a single feature, enhancing the model's exploration ability in multidimensional data spaces. Meanwhile, this integrated feature can reduce the model's dependence on the noise of any data source, thus enhancing its adaptability and generalization to new situations. ML-ALL not only enriches the information base of the model but also plays a crucial role in improving the accuracy of CPP recognition. In the single deep learning algorithm application, we chose D_ALL, which allows the model to deeply dig into the sequence data and learn abstract features of the sequences through its advanced network structure. This deep feature extraction provides a way to reveal deep patterns in biological sequences and is critical to understanding complex bioinformatics data. For the integrated learning method, we combined ML_ALL and D_ALL to take full advantage of their complementary strengths in pattern recognition. We have improved the comprehensive performance

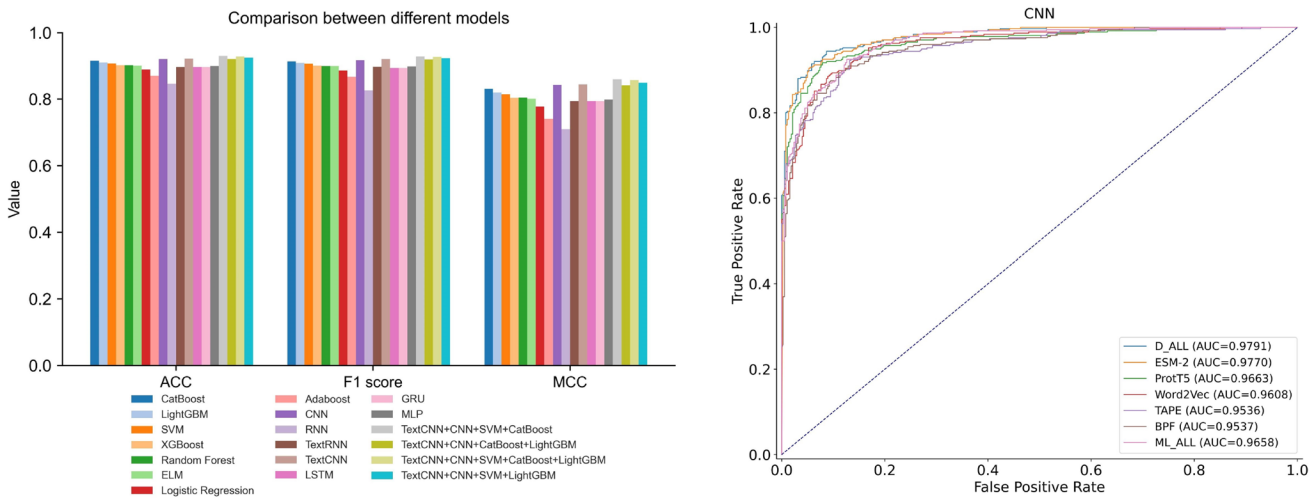


Fig. 6 A comparison of performance differences between various models using bar charts and ROC curves

Table 7 Performance comparison of different models on the validation dataset

Model	ACC	Se	Sp	AP	F1 score	MCC
CatBoost	0.9153	0.8989	0.9316	0.9747	0.9135	0.8311
LightGBM	0.9101	0.8989	0.9211	0.9753	0.9086	0.8203
SVM	0.9074	0.9043	0.9105	0.9691	0.9067	0.8148
XGBoost	0.9021	0.8963	0.9079	0.9730	0.9011	0.8043
Random forest	0.9021	0.8856	0.9184	0.9712	0.9000	0.8046
ELM	0.9008	0.8936	0.9079	0.9674	0.8996	0.8016
Logistic Regression	0.8889	0.8750	0.9026	0.9620	0.8868	0.7780
Adaboost	0.8704	0.8537	0.8868	0.9037	0.8676	0.7411
TextCNN	0.9220	0.9096	0.9342	0.9739	0.9206	0.8441
CNN	0.9206	0.8883	0.9526	0.9808	0.9176	0.8429
MLP	0.8995	0.8910	0.9079	0.9637	0.8981	0.7990
TextRNN	0.8968	0.9122	0.8816	0.9688	0.8979	0.7941
LSTM	0.8968	0.8750	0.9184	0.9300	0.8940	0.7943
GRU	0.8968	0.8723	0.9211	0.9526	0.8937	0.7945
RNN	0.8466	0.7367	0.9553	0.9237	0.8269	0.7097
TCSC	0.9299	0.9149	0.9447	0.9800	0.9285	0.8601
TCSCL	0.9286	0.9122	0.9447	0.9792	0.9270	0.8575
TCSL	0.9246	0.9096	0.9395	0.9780	0.9231	0.8495
TCCL	0.9206	0.9069	0.9342	0.9724	0.9191	0.8415

TextCNN + CNN + SVM + CatBoost (TCSC), TextCNN + CNN + SVM + CatBoost + LightGBM (TCSCL), TextCNN + CNN + SVM + LightGBM (TCSL), TextCNN + CNN + CatBoost + LightGBM (TCCL)

of the model in analyzing biological information by integrating traditional intuitive features with modern deep features. Its decision layer is integrated through LR models.

Figure 6 shows the ROC curves and histograms obtained by applying various classification algorithms to the training dataset. In addition, Table 7 shows the values of ACC, Se, Sp, AUC, AP, F1 score, and MCC obtained by five-fold cross-validations with these 19 different methods.

The data analysis in Table 7 shows that in the machine learning algorithms, CatBoost performs best on ACC, Sp, F1 score, and MCC with 91.53%, 93.16%, 91.35%, and 83.11%, respectively. SVM performs best on Se with 90.43%. LightGBM achieved the highest value on AP, with 97.53%. In the deep learning algorithms, TextCNN achieves he highest value on ACC, F1 score, and MCC with 92.2%, 92.06%, and 84.41%, respectively, while CNN achieves he highest value on Sp and AP with 95.26% and 98.08%, respectively.

TextRNN has the best performance on Se, with 91.22%. In the stacked model section, the model integrated with TextCNN, CNN, SVM, and CatBoost outperformed the other integrated models in all metrics except Sp, which was tied with the model integrated with TextCNN, CNN, SVM, CatBoost, and LightGBM at 94.47%. This model performs best on ACC, Se, AP, F1 score, and MCC with 92.99%, 91.49%, 98%, 92.85%, and 86.01%, respectively. This indicates that the integrated model outperforms the single deep learning and machine learning models in terms of overall performance, mainly attributed to the complementary properties of the different models and the effectiveness of the integration strategy.

It is clear from Fig. 6 that the single machine learning and deep learning algorithms underperform the stacked models in these metrics of ACC, F1 score, and MCC. In particular, the combined model of TextCNN, CNN, SVM, and CatBoost shows an advantage in these performance metrics. Observing the ROC curve plot on the right side, it can be seen that the learning method based on the integration of TextCNN, CNN, SVM, and CatBoost exhibits the largest AUC. It suggests that the method performs much better in recognizing the balance of true positive and false positive rates.

The above materials indicate that the integrated model can more comprehensively capture and utilize the patterns in the data by combining the strengths of multiple algorithms, thus demonstrating more robust and efficient performance when dealing with complex classification tasks. This further demonstrates the value of the strategy of integrating multiple models and features when facing challenging datasets.

3.4 Comparison with Existing State-of-the-Art Research Methods

In order to validate the effectiveness of the EnDM-CPP predictor presented in this paper, we compared it with methods that have performed well in the past few years. Specifically, in 2018, Manavalan et al. proposed the MLCPP method for recognizing CPPs in protein sequences, demonstrating the potential of computational models in CPP recognition. Pandey et al. proposed a KELM-CPPpred model based on kernel extreme learning machines in the same year. In 2020, Fu et al. proposed StackCPPred. It is a method that combines multiple prediction models to improve the accuracy of CPP recognition, which provides an effective computational framework for CPP recognition. In the same year, Arif et al. proposed TargetCPP for CPP prediction by using optimized multi-scale features and gradient-boosting decision trees. In 2021, de Oliveira et al. proposed BChemRF-CPPred, which combines machine learning frameworks and multiple classifiers. In 2022, Manavalan and Patra updated the MLCPP method and proposed MLCPP 2.0, further improving the recognition efficiency of CPPs. In 2023, Zhang et al. proposed SiameseCPP. SiameseCPP is a CPP prediction model based on a twin network, which uses a deep learning framework to predict CPP automatically.

In order to comprehensively evaluate the performance of the EnDM-CPP predictor, we conducted comparisons on

Table 8 Performance comparison of different models on the validation dataset

Model	ACC	Se	Sp	AUC	AP	F1 score	MCC
EnDM-CPP	0.9299	0.9149	0.9447	0.9783	0.9800	0.9285	0.8601
StackCPPred	0.9140	0.8963	0.9316	0.9485	0.9370	0.9120	0.8285
TargetCPP	0.9127	0.8936	0.9316	0.9736	0.9746	0.9106	0.8259
MLCPP 2.0	0.9087	0.8910	0.9263	0.9693	0.9697	0.9066	0.8179
MLCPP	0.9021	0.8670	0.9368	0.9727	0.9733	0.8981	0.8061
Kelm-Cpppred	0.8915	0.8963	0.8868	0.9599	0.9612	0.8915	0.7831
BChemRF-CPPred	0.8902	0.8484	0.9316	0.9568	0.9617	0.8849	0.7830

Table 9 Performance comparison of different models on the independent test dataset

Model	ACC	Se	Sp	AUC	AP	F1 score	MCC
EnDM-CPP	0.9495	0.9196	0.9806	0.9915	0.9923	0.9489	0.9008
SiameseCPP	0.9263	0.8763	0.9785	0.9746	0.9817	0.9239	0.8576
MLCPP 2.0	0.9221	0.9072	0.9376	0.9847	0.9866	0.9224	0.8447
MLCPP	0.9116	0.8742	0.9505	0.9853	0.9863	0.9099	0.8260
TargetCPP	0.9105	0.8825	0.9398	0.9769	0.9802	0.9097	0.8227
StackCPPred	0.8979	0.8928	0.9032	0.9568	0.9518	0.8993	0.7958
Kelm-Cpppred	0.8937	0.8763	0.9118	0.9581	0.9653	0.8938	0.7881
BChemRF-CPPred	0.8547	0.8330	0.8774	0.9485	0.9565	0.8541	0.7106

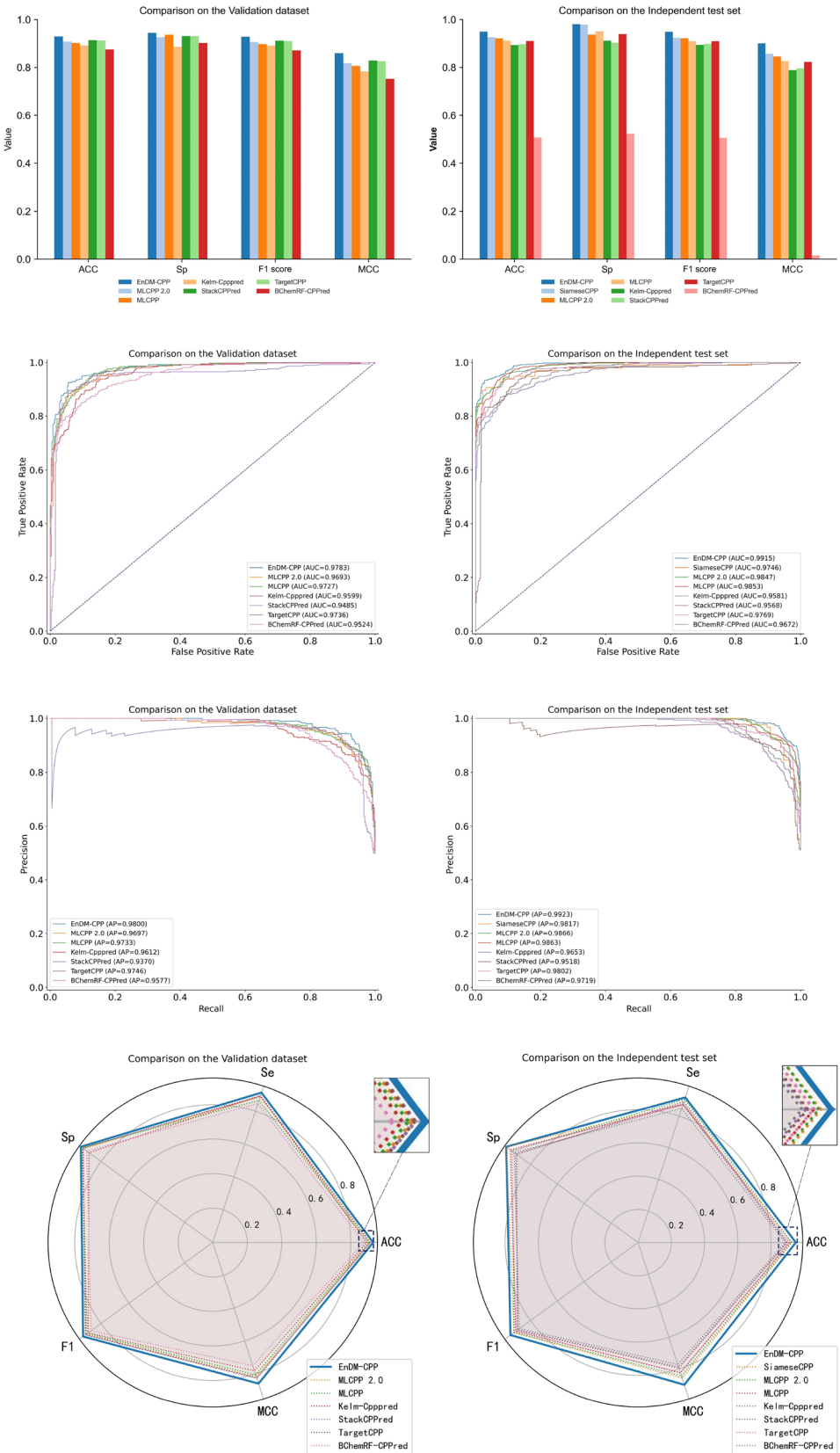


Fig. 7 The performance of the test and validation datasets, illustrated with ROC curves, PR curves, bar charts, and radar charts

Table 10 Performance comparison of different models on the CPP924 dataset

Model	ACC	Se	Sp	AUC	AP	F1 score	MCC
EnDM-CPP	0.9491	0.9010	0.9978	0.9872	0.9897	0.9468	0.9027
SiameseCPP	0.9347	0.9247	0.945	0.9784	0.9794	0.9347	0.8697
MLCPP 2.0	0.9286	0.9225	0.9391	0.9817	0.9847	0.9305	0.8617
Kelm-Cpppred	0.9145	0.9225	0.9065	0.9698	0.9750	0.9156	0.8292
TargetCPP	0.9105	0.8825	0.9398	0.9769	0.9802	0.9097	0.8227
MLCPP	0.9102	0.9032	0.9173	0.9696	0.9763	0.9100	0.8200
BChemRF-CPPred	0.9070	0.8903	0.9239	0.9668	0.9738	0.9059	0.8145
StackCPPred	0.8897	0.9182	0.8608	0.9402	0.9189	0.8933	0.7806

the validation set and the test set, respectively. The specific comparison results are shown in Tables 8 and 9. To thoroughly evaluate the performance of EnDM-CPP, we plotted histograms and ROC curves, as shown in Fig. 7. In addition, in order to better evaluate the generalization of the model, we conducted comparative analysis on the CPP924 dataset, and the results are shown in Table 10.

The data in Table 8 shows that the EnDM-CPP predictor outperforms existing methods in all evaluation metrics on the validation set. The scores for ACC, Se, Sp, AUC, AP, F1 score, and MCC are 92.99%, 91.49%, 94.47%, 97.83%, 98%, 92.85%, and 86.01%, respectively. The enhancements of EnDM-CPP in each index are as follows: ACC increases by 1.59–3.97 percentage points, Se improves by 1.86–6.65 percentage points, Sp grows by 0.79–5.79 percentage points, AUC improves by 0.47–2.98 percentage points, AP increases by 0.54–4.3 percentage points, F1 score improves by 1.65–4.36 percentage points, and MCC improves by 3.16–7.71 percentage points.

According to the data in Table 9, the EnDM-CPP predictor performs well in various evaluation metrics of the test set, with higher scores for ACC, Se, Sp, AUC, AP, F1 score, and MCC than other methods in the table. Specifically, EnDM-CPP achieves 94.95% in ACC, higher than other models such as SiameseCPP at 92.63% and MLCPP 2.0 at 92.21%. In terms of Se, EnDM-CPP reaches 91.96%, not only surpassing MLCPP 2.0's 90.72% but also leading other comparative models. On Sp, EnDM-CPP achieves 98.06%, which is superior to other models, indicating its high accuracy in distinguishing true CPPs from non-PPs. The AUC value is 99.15%, indicating the high stability and reliability of the model in prediction tasks. MCC further proves the advantages of EnDM-CPP. The MCC on the independent test set is 90.08%, much higher than other models. This indicates that EnDM-CPP can accurately predict positive and negative samples.

To further validate the model's generalization ability, we compared it with other methods on the CPP924 dataset (see Table 10). Although EnDM-CPP's Se is slightly lower than that of some models, it outperforms all other methods in the table across other metrics, particularly in F1 score and MCC, where it achieves 94.68% and 90.27%, respectively. This indicates that EnDM-CPP not only excels in accuracy and specificity but also effectively balances precision and recall. A comprehensive evaluation across different datasets highlights EnDM-CPP's excellent overall performance and strong generalization ability.

EnDM-CPP not only performs well on the validation set but also demonstrates its efficient predictive ability and robustness on the independent test set. This result further emphasizes the potential and effectiveness of EnDM-CPP in improving CPP prediction accuracy.

By looking at the histograms, ROC curves, PR curves, and radar charts in Fig. 7, we can see that the EnDM-CPP predictor compares with the six existing prediction methods (MLCPP 2.0, MLCPP, Kelm-Cpppred, StackCPPred, TargetCPP, and BChemRF-CPPred) on the validation dataset and with the seven existing prediction methods (SiameseCPP, MLCPP 2.0, MLCPP, Kelm-Cpppred, StackCPPred, TargetCPP and BChemRF-CPPred) between the differences in scores on the assessment metrics of Sp, ACC, AUC, MCC, F1 score and AP. EnDM-CPP has larger areas under both the ROC and PR curves in both the validation and independent test sets. The larger the area under the ROC curve and PR curve, the better the model performance. We also presented radar plots on the validation set and the independent test set, which compared the performance of various models on different evaluation metrics (ACC, Se, Sp, F1 score, MCC). Each vertex in the radar chart represents an evaluation metric, and the closer it is to the outer circle, the better the performance. The EnDM-CPP model uses a solid blue line, while other comparison methods use dashed lines. It is evident that the contour of the solid blue line extends the farthest on most vertices. This indicates that EnDM-CPP outperforms other models in terms of ACC, Se, Sp, F1 score, and MCC, further validating its efficiency and stability in predicting CPPs.

4 Unbalanced Experiment

In this study, we conducted experiments on the issue of imbalanced datasets to explore the impact of downsampling methods on model performance. The original class distribution of our dataset was 473 positives and 970 negatives, and the downsampling was 473 positives and 473 negatives. In the experiment, we used CTDC as a feature extraction method. The SVM was used as a model to train and evaluate the unbalanced dataset and the downsampled balanced dataset, respectively.

Table 11 shows the performance comparison of SVM models on an imbalanced dataset and a downsampled balanced dataset. The results show that the downsampled balanced dataset significantly enhanced the model's performance. Specifically, the accuracy on the balanced dataset increases from 78.89% to 87.89%, AUC increases from 88.45% to 97.58%, AUPRC rises from 93.71% to 97.76%, F1 score increases from 84.15% to 87.7%, and MCC rises from 52.56% to 76.02%. In addition, we plotted the ROC curve and the PR curve to compare the two more intuitively, as shown in Fig. 8.

Figure 8 shows the ROC and PR curves on imbalanced and balanced datasets. It can be seen that the areas under the curves (AUC and AUPRC) of the balanced dataset are larger. By comparing the changes in model performance before and after downsampling, we found that downsampling not only improved the overall performance of the model, but also made the model more balanced in identifying positive and negative samples.

Table 11 Comparison of model performance metrics on imbalanced and balanced datasets

Dataset	ACC	AUC	AUPRC	F1 score	MCC
Balanced dataset	0.8789	0.9758	0.9776	0.8770	0.7602
Imbalanced dataset	0.7889	0.8845	0.9371	0.8415	0.5256

5 Discussion

This paper introduces a novel CPP prediction method based on stacked ensemble learning named EnDM-CPP. Considering the high similarity and lack of diversity of homologous sequences in existing CPP datasets, we screened critical data from a wide range of literature and databases and constructed a balanced dataset. This dataset contains 473 CPPs and an equal number of non-CPP samples, ensuring the breadth and representativeness of the data. By combining advanced machine learning and deep learning techniques, we demonstrated the powerful identification capabilities of our model. The following is a detailed summary of key findings and model strengths:

(1) We have integrated traditional and innovative feature extraction methods through screening literature and databases. ProtT5 especially performs excellently in capturing subtle differences in sequences. The combined application of these features provides a solid foundation for the efficient recognition of CPPs.

(2) EnDM-CPP integrates four basic models, SVM, CatBoost, CNN, and TextCNN, which are able to capture data features from different perspectives. Integrating these models can enhance their ability to recognize and process complex data patterns. For example, SVM ensures the accuracy of the model with its excellent

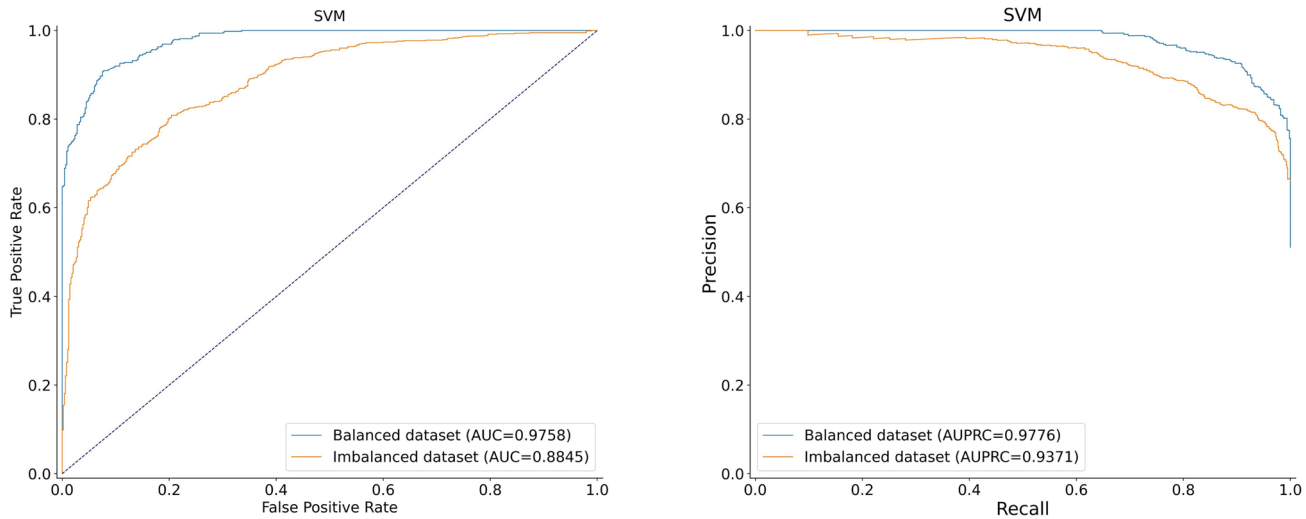


Fig. 8 ROC and PR curves for imbalanced and balanced datasets

classification effect. CatBoost can efficiently deal with classification problems, strengthening the model's expressiveness and adaptability to various data types. The power of CNN lies in its ability to learn essential features in sequence data through convolutional layers automatically. At the same time, TextCNN is particularly good at capturing local features and contextual information in sequences. Combining the two provides the model with cognitive capabilities in depth and width.

(3) The stacked ensemble learning method we adopt ensures stable performance on different datasets by combining predictions from multiple models. This reduces the risk of overfitting and improves the model's generalization ability. Choosing stacked ensemble learning can maximize the advantages of different models. By integrating the predictions of each layer model through a hierarchical structure, the final decision is made more accurate. The top-level LR optimizes the predictions of the bottom-level model, which provides a powerful integrated prediction mechanism.

In addition, comparing and analyzing EnDM-CPP with current popular CPP predictors, including SiameseCPP and MLCPP 2.0, EnDM-CPP showed advantages in various evaluation indicators. This not only demonstrates the excellent performance of our model on standard evaluation metrics but also demonstrates its potential and reliability in practical biological scenarios. Specifically, EnDM-CPP achieves the following performance improvements in the test set: ACC increases by 1.59–3.97 percentage points, Se increases by 1.86–6.65 percentage points, Sp increases by 0.79–5.79 percentage points, AUC increases by 0.47–2.98 percentage points, AP improves by 0.54–4.3 percentage points, F1 score increases by 1.65–4.36 percentage points, MCC increases by 3.16–7.71 percentage points. These achievements not only demonstrate the comprehensive leadership of the EnDM-CPP predictor in various evaluation indicators but also demonstrate its significant advantages compared to existing methods.

Although current feature selection methods are performed based on ACC and MCC metrics, this does not guarantee that the selected feature combinations are optimal. Due to the complexity of CPPs, other more effective feature combinations may exist. Our method has not yet been able to fully automate the identification and utilization of these potentially optimal features. In addition, we currently do not have a platform that allows researchers and developers to freely combine different features and evaluate the effectiveness of these combinations. Such a platform would greatly enhance the adaptability and innovation of models, especially in the rapidly changing field of bioinformatics. In conclusion, although current models have made significant progress in predicting CPPs, we still see room for improvement and expansion. Through continuous research and innovation, we hope to develop more accurate, adaptable, and widely applicable predictive models.

The dataset and code used in this paper have been publicly released to GitHub for direct access and use by other researchers. The purpose of this public sharing is to help peer researchers gain a deeper understanding of our work, reproduce our research findings, or expand their research based on it. Related resources and codes can be obtained through the following website: https://github.com/tudou1231/EnDM-CPP.git.

6 Conclusion

In this paper, the EnDM-CPP model we developed brings an innovative breakthrough in CPP prediction. First, a significant advantage of EnDM-CPP lies in constructing its training dataset. EnDM-CPP ensures a high degree of generalization capability by employing a low homology and diverse dataset. This means that the model can effectively cope with a wide range of unseen sequences of CPPs, which enhances the robustness and adaptability of the model in practical applications. In terms of feature selection, EnDM-CPP carefully selects six machine learning features and two deep learning features from a large number of candidate features. These features enable the model to capture the complex properties of CPPs comprehensively from multiple perspectives. This method, which combines machine learning and deep learning features, enables EnDM-CPP to capture CPP characteristics more comprehensively, thereby improving the accuracy of prediction.

EnDM-CPP's two-layer prediction framework is one of its core technologies. It effectively integrates selected features and optimizes prediction performance by cascading models. This architecture's advantage is that it can further refine and enhance the prediction results based on different models, thus realizing more accurate and reliable predictions than a single model.

In terms of performance evaluation, EnDM-CPP's performance is equally impressive. Compared with a variety of current advanced CPP prediction methods, EnDM-CPP excels in several key performance metrics, including but not limited to accuracy, sensitivity, and specificity. This not only proves EnDM-CPP's superiority but also highlights its potential application value in the field of CPP prediction.

The development of the EnDM-CPP model integrates advanced data processing, feature selection, and model design methods to provide a highly competitive tool in the prediction of CPPs. This progress not only promotes the in-depth research of CPPs but also provides new perspectives

and methods for the design of future drug delivery systems and the optimization of disease treatment strategies, which is expected to have a wide impact in the biomedical field.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s12539-024-00673-4.

Funding This research was funded by the Natural Science Foundation of Jiangsu Province of China (Grant No. BK20230626), partly supported by the open funds of the State Key Laboratory of Plant Environmental Resilience (Grant No. SKLPERKF2401), Supported by the Open Project of State Key Laboratory of Animal Biotech Breeding (Grant No. 2024SKLAB6-1) and Fourth Batch of Leading Innovative Talents Introduction and Training Projects under the Longcheng Talent Plan in Changzhou City (Basic Research and Innovation) (Grant No. CQ20230086), and also supported by Changzhou Sci&Tech Program (Grant No. CJ20241083).

Data Availability Publicly available datasets were analyzed in this study. Codes and data are available at https://github.com/tudou1231/EnDM-CPP.git.

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Ethical Approval This article does not contain any studies with animals performed by any of the authors.

References

1. Desale K, Kuche K, Jain S (2021) Cell-penetrating peptides (CPPs): an overview of applications for improving the potential of nanotherapeutics. Biomater Sci 9:1153–1188. https://doi.org/10.1039/D0BM01755H
2. Milletti F (2012) Cell-penetrating peptides: classes, origin, and current landscape. Drug Discov Today 17:850–860. https://doi.org/10.1016/j.drudis.2012.03.002
3. Cafaro A, Tripiciano A, Sgadari C et al (2015) Development of a novel AIDS vaccine: the HIV-1 transactivator of transcription protein vaccine. Expert Opin Biol Ther 15(1):13–29. https://doi.org/10.1517/14712598.2015.1021328
4. Guidotti G, Brambilla L, Rossi D (2017) Cell-penetrating peptides: from basic research to clinics. Trends Pharmacol Sci 38:406–424. https://doi.org/10.1016/j.tips.2017.01.003
5. Hasannejad-Asl B, Pooresmaeil F, Takamoli S et al (2022) Cell penetrating peptide: a potent delivery system in vaccine development. Front Pharmacol 13:1072685. https://doi.org/10.3389/fphar.2022.1072685
6. Reissmann S (2014) Cell penetration: scope and limitations by the application of cell-penetrating peptides. J Pept Sci 20:760–784. https://doi.org/10.1002/psc.2672
7. Su R, Hu J, Zou Q et al (2020) Empirical comparison and analysis of web-based cell-penetrating peptide prediction tools. Brief Bioinform 21:408–420. https://doi.org/10.1093/bib/bby124
8. Hällbrink M, Kilk K, Elmquist A et al (2005) Prediction of cell-penetrating peptides. Int J Pept Res Ther 11:249–259. https://doi.org/10.1007/s10989-005-9393-1
9. Dobchev DA, Mäger I, Tulp I et al (2010) Prediction of cell-penetrating peptides using artificial neural networks. Curr Comput Aided Drug Des 6:79–89. https://doi.org/10.2174/157340910791202478
10. Manavalan B, Subramaniyam S, Shin TH et al (2018) Machine-learning-based prediction of cell-penetrating peptides and their uptake efficiency with improved accuracy. J Proteome Res 17:2715–2726. https://doi.org/10.1021/acs.jproteome.8b00148
11. Pandey P, Patel V, George NV et al (2018) KELM-CPPpred: kernel extreme learning machine-based prediction model for cell-penetrating peptides. J Proteome Res 17:3214–3222. https://doi.org/10.1021/acs.jproteome.8b00322
12. Fu X, Cai L, Zeng X et al (2020) StackCPPred: a stacking and pairwise energy content-based prediction of cell-penetrating peptides and their uptake efficiency. Bioinformatics 36:3028–3034. https://doi.org/10.1093/bioinformatics/btaa131
13. Arif M, Ahmad S, Ali F et al (2020) TargetCPP: accurate prediction of cell-penetrating peptides from optimized multi-scale features using gradient boost decision tree. J Comput Aided Mol Des 34:841–856. https://doi.org/10.1007/s10822-020-00307-z
14. De Oliveira ECL, Santana K, Josino L et al (2021) Predicting cell-penetrating peptides using machine learning algorithms and navigating in their chemical space. Sci Rep 11:7628. https://doi.org/10.1038/s41598-021-87134-w
15. Manavalan B, Patra MC (2022) MLCPP 2.0: an updated cell-penetrating peptides and their uptake efficiency predictor. J Mol Biol 434:167604. https://doi.org/10.1016/j.jmb.2022.167604
16. Park H, Park J-H, Kim MS et al (2023) In silico screening and optimization of cell-penetrating peptides using deep learning methods. Biomolecules 13:522. https://doi.org/10.3390/biom13030522
17. Zhang X, Wei L, Ye X et al (2023) SiameseCPP: a sequence-based Siamese network to predict cell-penetrating peptides by contrastive learning. Brief Bioinform 24:1–12. https://doi.org/10.1093/bib/bbac545
18. Wei L, Tang J, Zou Q (2017) SkipCPP-Pred: an improved and promising sequence-based predictor for predicting cell-penetrating peptides. BMC Genomics 18(Suppl 7):742. https://doi.org/10.1186/s12864-017-4128-1
19. Kardani K, Bolhassani A (2021) Cppsite 2.0: an available database of experimentally validated cell-penetrating peptides predicting their secondary and tertiary structures. J Mol Biol 433:166703. https://doi.org/10.1016/j.jmb.2020.11.002
20. Fu L, Niu B, Zhu Z et al (2012) CD-HIT: accelerated for clustering the next-generation sequencing data. Bioinformatics 28:3150–3152. https://doi.org/10.1093/bioinformatics/bts565
21. Kabir M, Arif M, Ahmad S et al (2018) Intelligent computational method for discrimination of anticancer peptides by incorporating sequential and evolutionary profiles information. Chemom Intell Lab Syst 182:158–165. https://doi.org/10.1016/j.chemolab.2018.09.007
22. Rao R, Bhattacharya N, Thomas N et al (2019) Evaluating protein transfer learning with TAPE. arXiv. https://arxiv.org/abs/1906.08230
23. Brandes N, Ofer D, Peleg Y et al (2022) ProteinBERT: a universal deep-learning model of protein sequence and function. Bioinformatics 38:2102–2110. https://doi.org/10.1093/bioinformatics/btac020
24. Liu B, Heinzinger M, Dallago C et al (2021) ProtTrans: towards cracking the language of life's code through self-supervised learning. https://arxiv.org/abs/2007.06225
25. Pavlyshenko B (2018) Using stacking approaches for machine learning models. In: 2018 IEEE second international conference on data stream mining & processing (DSMP), pp 255–258. https://doi.org/10.1109/DSMP.2018.8478522
26. Qiang X, Zhou C, Ye X et al (2018) CPPred-FL: a sequence-based predictor for large-scale identification of cell-penetrating

peptides by feature representation learning. Brief Bioinform. https://doi.org/10.1093/bib/bby091
27. Wei L, Xing P, Su R et al (2017) CPPred-RF: a sequence-based predictor for identifying cell-penetrating peptides and their uptake efficiency. J Proteome Res 16:2044–2053. https://doi.org/10.1021/acs.jproteome.7b00019
28. Hayat M, Khan A (2011) Predicting membrane protein types by fusing composite protein sequence features into pseudo amino acid composition. J Theor Biol 271:10–17. https://doi.org/10.1016/j.jtbi.2010.11.017
29. Hayat M, Khan A, Yeasin M (2012) Prediction of membrane proteins using split amino acid and ensemble classification. Amino Acids 42:2447–2460. https://doi.org/10.1007/s00726-011-1053-5
30. Georgiev AG (2009) Interpretable numerical descriptors of amino acid space. J Comput Biol 16:703–723. https://doi.org/10.1089/cmb.2008.0173
31. Liang G, Chen G, Niu W et al (2008) Factor analysis scales of generalized amino acid information as applied in predicting interactions between the human amphiphysin-1 SH3 domains and their peptide ligands. Chem Biol Drug Des 71:345–351. https://doi.org/10.1111/j.1747-0285.2008.00641.x
32. Kidera A, Konishi Y, Oka M et al (1985) Statistical analysis of the physical properties of the 20 naturally occurring amino acids. J Protein Chem 4:23–55. https://doi.org/10.1007/BF01025492
33. Van Westen GJ, Swier RF, Wegner JK et al (2013) Benchmarking of protein descriptor sets in proteochemometric modeling (part 1): comparative study of 13 amino acid descriptor sets. J Cheminform 5:41. https://doi.org/10.1186/1758-2946-5-41
34. Eisenberg D, Schwarz E, Komaromy M et al (1984) Analysis of membrane and surface protein sequences with the hydrophobic moment plot. J Mol Biol 179:125–142. https://doi.org/10.1016/0022-2836(84)90309-7
35. Karshikoff A, Nilsson L, Ladenstein R (2015) Rigidity versus flexibility: the dilemma of understanding protein thermal stability. FEBS J 282:3899–3917. https://doi.org/10.1111/febs.13343
36. Huber R (1987) Flexibility and rigidity, requirements for the function of proteins and protein pigment complexes. Eleventh Keilin Memorial Lecture. Biochem Soc Trans 15:1009–1020. https://doi.org/10.1042/bst0151009
37. Tang H, Chen W, Lin H (2016) Identification of immunoglobulins using Chou's pseudo amino acid composition with feature selection technique. Mol Biosyst 12:1269–1275. https://doi.org/10.1039/C5MB00883B
38. Govindan G, Nair AS (2011) Composition, transition and distribution (CTD)—a dynamic feature for predictions based on hierarchical structure of cellular sorting. In: 2011 Annual IEEE India conference, pp 1–6. https://doi.org/10.1109/INDCON.2011.6139332
39. Chen Z, Zhao P, Li F et al (2018) iFeature: a Python package and web server for features extraction and selection from protein and peptide sequences. Bioinformatics 34:2499–2502. https://doi.org/10.1093/bioinformatics/bty140
40. Chou K-C (2009) Pseudo amino acid composition and its applications in bioinformatics, proteomics and system biology. Curr Proteomics 6:262–274. https://doi.org/10.2174/157016409789973707
41. Li F-M, Wang X-Q (2016) Identifying anticancer peptides by using improved hybrid compositions. Sci Rep 6:33910. https://doi.org/10.1038/srep33910
42. Kopasov AA, Melnikov AS (2020) Estimation of genomic characteristics by analyzing k-mer frequency in de novo genome projects. arXiv. https://doi.org/10.48550/arXiv.2007.11339
43. Ju Z, Cao J-Z (2017) Prediction of protein N-formylation using the composition of k-spaced amino acid pairs. Anal Biochem 534:40–45. https://doi.org/10.1016/j.ab.2017.07.011
44. Lin Z, Akin H, Rao R et al (2023) Evolutionary-scale prediction of atomic-level protein structure with a language model. Science 379:1123–1130. https://doi.org/10.1126/science.ade2574
45. Noble WS (2006) What is a support vector machine? Nat Biotechnol 24:1565–1567. https://doi.org/10.1038/nbt1206-1565
46. Ali L, Niamat A, Khan J et al (2019) An optimized stacked support vector machines based expert system for the effective prediction of heart failure. IEEE Access 7:54007–54014. https://doi.org/10.1109/ACCESS.2019.2909969
47. Dorogush AV, Ershov V, Gulin A (2023) CatBoost: gradient boosting with categorical features support. arXiv. https://doi.org/10.48550/arXiv.1810.11363
48. Boateng EY, Abaye DA (2019) A review of the logistic regression model with emphasis on medical research. J Data Anal Inf Process 7:190–207. https://doi.org/10.4236/jdaip.2019.74012
49. Wainberg M, Merico D, Delong A et al (2018) Deep learning in biomedicine. Nat Biotechnol 36:829–838. https://doi.org/10.1038/nbt.4233
50. Alzubaidi L, Zhang J, Humaidi AJ et al (2021) Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8:44. https://doi.org/10.1186/s40537-021-00444-8
51. Guo B, Zhang C, Liu J et al (2019) Improving text classification with weighted word embeddings via a multi-channel TextCNN model. Neurocomputing 363:366–374. https://doi.org/10.1016/j.neucom.2019.07.052
52. De Angeli K, Gao S, Danciu I et al (2022) Class imbalance in out-of-distribution datasets: improving the robustness of the TextCNN for the classification of rare cancer types. J Biomed Inform 125:103957. https://doi.org/10.1016/j.jbi.2021.103957
53. Luo Z, Wang R, Sun Y et al (2024) Interpretable feature extraction and dimensionality reduction in ESM2 for protein localization prediction. Brief Bioinform 25:1–16. https://doi.org/10.1093/bib/bbad534