Supplemental information

TheraPepNet: Enhancing Therapeutic Peptide Recognition via Temporal Sequence Prediction Networks and Preprocessed Features

Zhenming Wu, Yangyang Sun, Xiaoquan Su, and Jin Zhao

School of Computer Science and Technology, Qingdao University, Ningxia Road, 266071, Shandong, China

Contents

Supplemental Notes

Supplementary Note 1: Pre-Trained Embedding Vector from ProtBERT

To capture the diverse properties of amino acids, we use pre-trained embedding vectors from ProtBERT, a model designed specifically for protein sequences. ProtBERT is based on the Bidirectional Encoder Representations from Transformers architecture. It learns rich representations through unsupervised learning from large-scale protein sequence data. During pre-training, ProtBERT employs a Masked Language Modeling approach, which helps it capture syntactic and semantic features of sequences.

We obtain the ProtBERT model via the Hugging Face transformers library and use a tokenizer to encode raw peptide sequences into token IDs. To standardize sequence lengths, we apply truncation and padding, while attention masks ensure that the model focuses on real data rather than padding. Finally, we input the processed sequences into the ProtBERT model to extract fixed-dimensional embedding vectors for downstream classification tasks.

Supplementary Note 2: Feed-Forward Network

The Feed-Forward Network (FFN) is typically used to apply non-linear transformations to the representation at each position. It is a fully connected network consisting of two linear transformations and an activation function, with the ReLU function commonly used as the activation. The mathematical expression for the FFN is as follows:

$$Y = W_2 \cdot ReLU(W_1 X + b_1) + b_2$$

where Y is the output vector, X is the input vector; W_1 and W_2 are the weight matrices for the first and second layers, respectively; b_1 and b_2 are the corresponding bias vectors. These variables work together on the input data to generate the final output Y through linear transformations and the ReLU activation function.

Supplementary Note 3: Gating Mechanism

To dynamically adjust the contribution of each feature output, we introduce a gating mechanism that uses a Sigmoid function to generate gating values between 0 and 1, which are then used to weight the outputs from different modules. The formula for the gating mechanism is presented as follows:

$$\begin{cases} gate = \sigma(w_{gate} \cdot input) \\ output = gate \times input \end{cases} \tag{7}$$

Here, σ denotes the Sigmoid activation function, w_{gate} represents the learnable gating weight matrix, and input refers to the corresponding output of each network module. The gating value for each module controls the extent to which its output is retained. Specifically, when the gating value is close to 1, the output's influence is significant, whereas when the gating value approaches 0, the module's contribution is suppressed. This gating mechanism effectively modulates the outputs of the temporal network, enhancing the model's expressive power and providing greater control and flexibility for fine-tuning.

Supplementary Note 4: Assessment Metrics

To assess the performance of the model, we employed commonly used evaluation metrics such as accuracy (ACC), precision (PREC), sensitivity (SEN), specificity (SPE), F1-score, and Matthews correlation coefficient (MCC). These metrics serve as universal indicators for gauging the effectiveness of the model. The detailed description of these six indicators is as follows:

$$ACC = \frac{TP+TN}{TP+TN+FP+FN}, \tag{1}$$

$$PREC = \frac{TP}{TP+FP}, \tag{2}$$

$$SEN = \frac{TP}{TP+FN}, \tag{3}$$

$$SPE = \frac{TN}{TN+FP}, \tag{4}$$

$$F1-score = \frac{2 \cdot PREC \cdot SEN}{PREC+SEN}, \tag{5}$$

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (6)$$

where TP represents true positives, TN represents true negatives, FP represents false positives, and FN represents false negatives.

Supplementary Note 5: Hyperparameter Setting

In this study, the model hyperparameters are set as follows: a batch size of 64, 300 training epochs, a dropout rate of 0.2, and a learning rate of 0.001. For the Global Feature Learning Network, we employed 8 attention heads and a 3-layer network structure to effectively capture global features. The Local Feature Learning Network progressively increases the number of channels ([8, 16, 32, 64, 128]) and uses convolutional kernel with the size of 3 to extract local features. These hyperparameters were carefully tuned to optimize the overall training performance of the model.

Supplementary Note 6: Experimental Setup

TheraPepNet is built on the PyTorch 2.4.1 deep learning framework and developed and tested in a Python 3.8 environment. We utilized four NVIDIA RTX 3090 GPUs for parallel computation to enhance training speed and computational efficiency. Through multi-GPU parallel processing, we were able to significantly accelerate the training process on large-scale datasets. Even without GPU acceleration, the experiments can still run on a CPU.

Supplemental Figures

1. Lengths Distribution

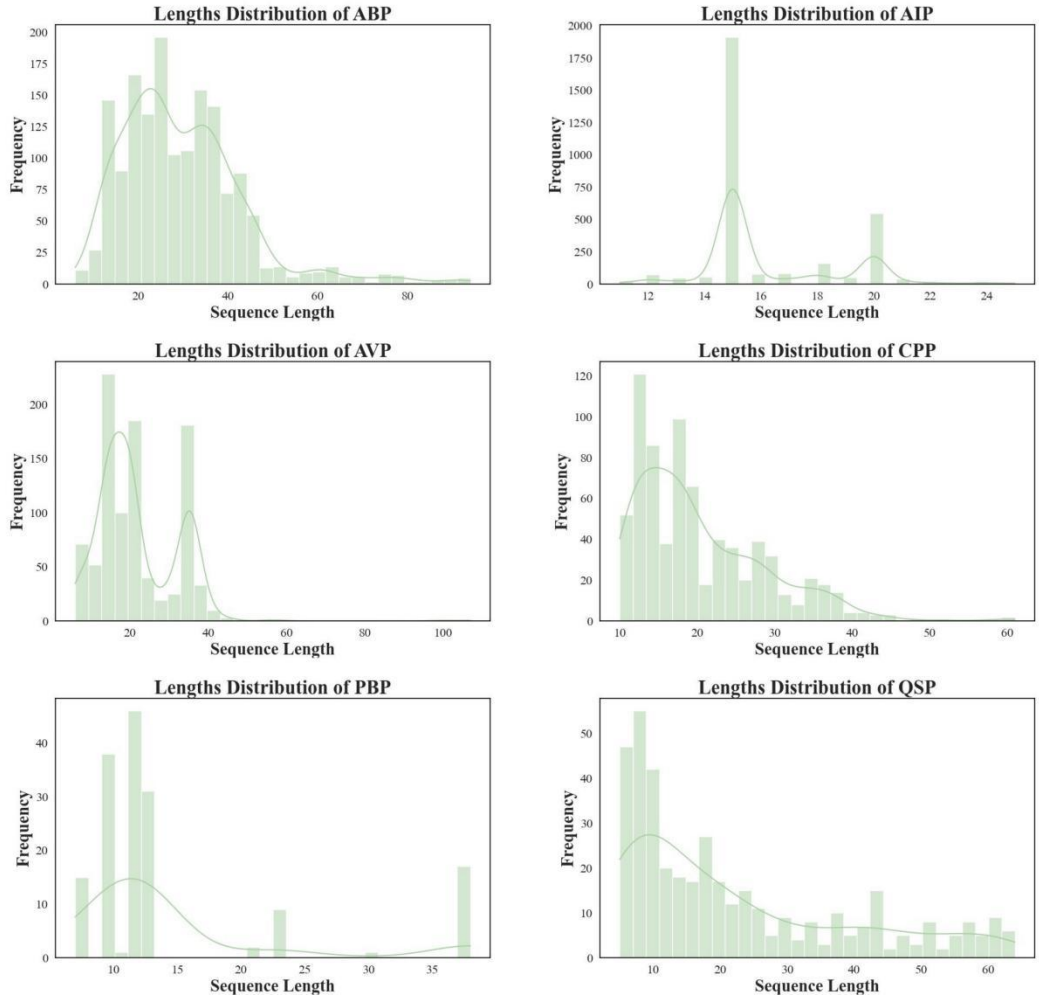

Figure S1. Distribution of peptide sequence lengths across six datasets: ABP, AIP, AVP, CPP, PBP, and QSP.

2. Hydrophobicity and Charge Distribution

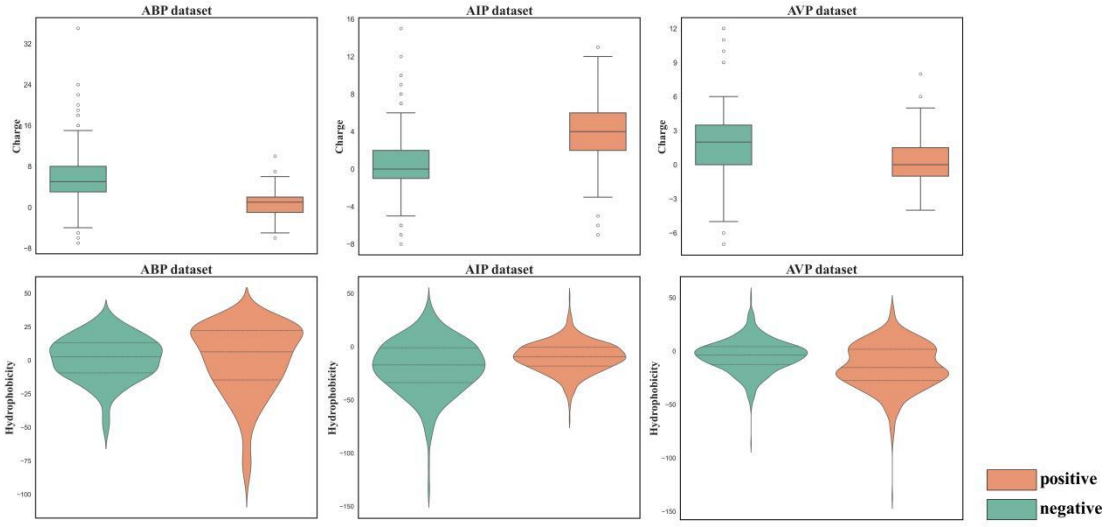

Figure S2. The hydrophobicity and charge distribution characteristics of positive and negative samples in the ABP, AIP, and AVP datasets.

3. Learning Curve

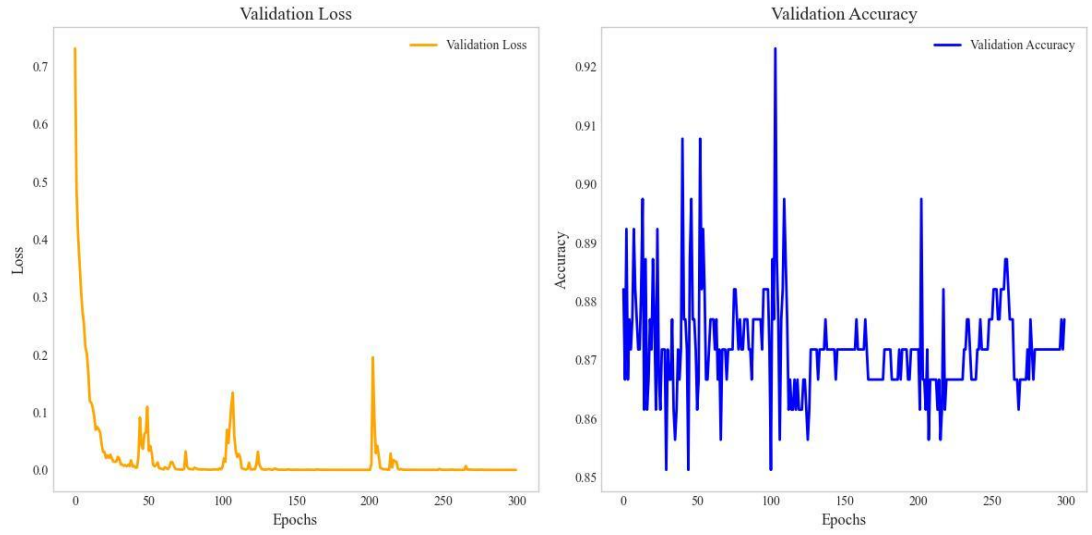

Figure S3. Learning curve of TheraPepNet on the mixed dataset composed of CPP, PBP, and QSP. The changes in the validation loss and accuracy are shown as the number of epochs increases.

Supplemental Tables

1. Experimental Results

Table S1. The Performance of TheraPepNet and Four Other Methods on the ABP.

Method	ACC	SE	SP	F1-score	MCC
ETFC	0.894	0.929	0.873	0.879	0.803
PrMFTP	0.903	0.910	0.895	0.899	0.806
TPpred-ATMV	0.911	0.852	0.873	0.866	0.826
DeepTPpred	0.915	0.904	0.876	0.886	0.822
TheraPepNet	0.929	0.932	0.927	0.930	0.860

Table S2. The Performance of TheraPepNet and Four Other Methods on the AIP.

Method	ACC	SE	SP	F1-score	MCC
ETFC	0.613	0.631	0.592	0.687	0.220
PrMFTP	0.675	0.610	0.661	0.705	0.290
TPpred-ATMV	0.639	0.690	0.662	0.703	0.251
DeepTPpred	0.648	0.682	0.586	0.710	0.260
TheraPepNet	0.707	0.681	0.716	0.698	0.313

Table S3. The Performance of TheraPepNet and Four Other Methods on the AVP.

Method	ACC	SE	SP	F1-score	MCC
ETFC	0.800	0.790	0.808	0.768	0.595
PrMFTP	0.837	0.763	0.800	0.766	0.675
TPpred-ATMV	0.767	0.719	0.762	0.725	0.536
DeepTPpred	0.821	0.775	0.857	0.805	0.635
TheraPepNet	0.854	0.809	0.891	0.861	0.705

2. Consuming Time and Memory

Table S4. GPU Time for Running 300 Epochs Across Different Datasets for Various Methods (Unit of Measurement: Seconds).

method \ dataset	ABP	AIP	AVP
ETFC	516.0s	879.2s	203.4s
PrMFTP	447.0s	719.2s	274.6s
TPpred-ATMV	106.5s	256.9s	65.2s
DeepTPpred	75.3s	97.8s	76.4s
TheraPepNet	281.1s	428.3s	183.4s

Table S5. Computational Memory Requirements of Different Methods on Various Datasets (Unit of Measurement: MegaByte).

method \ dataset	ABP	AIP	AVP
ETFC	553.37 MB	551.55 MB	552.52 MB
PrMFTP	359.88 MB	422.52 MB	392.22 MB
TPpred-ATMV	349.50 MB	353.44 MB	349.27 MB
DeepTPpred	356.67 MB	357.07 MB	357.67 MB
TheraPepNet	334.00 MB	320.78 MB	339.60 MB