Supplemental information

TF-BAPred: a universal bioactive peptide predictor integrating multiple feature representations

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

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

Supplemental Notes

Supplementary Note 1: Handling imbalanced datasets

In the antimicrobial peptide dataset of Ma et al., there are 1085 positive samples and 58776 negative samples. This dataset is highly imbalanced, and without intervention, the model may lean towards predicting negative samples, thereby neglecting the positive ones. Therefore, when training TF-BAPred using this dataset, adjusting class weights to assign greater importance to positive samples compensates for their scarcity, enabling the model to better learn the features of positive instances and enhance prediction performance. In our binary classification task, positive samples are represented by $y = 1$, and negative samples by $y = 0$. Without considering class weights, the cross-entropy loss function is typically used, which can be formulated as follows for computation:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i log(p_i) + (1 - y_i) log(1 - p_i)] \quad i = 1, 2, 3 \ldots N$$

Where N is the number of samples, y_i is the true label of the *i-th* sample , and p_i is the predicted probability of the *i-th* sample being positive. When incorporating class weights, the loss function can be formulated as follows:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[w_1 \cdot y_i log(p_i) + w_0 \cdot (1 - y_i) log(1 - p_i)] \quad i = 1, 2, 3 \ldots N$$

where w_1 represents the weight assigned to positive samples, and w_0 represents the weight assigned to negative samples. In Ma's dataset, the positive samples are weighted 54 times more than the negative samples, specifically $w_1 = 54 \times w_0$. Consequently, during the computation of the loss function, the model amplifies the prediction error for positive samples based on the assigned class weights, resulting in larger gradients in the loss function. This approach directs the model to pay closer attention to learning from positive samples when updating parameters.

Supplementary Note 2: Verify overfitting

TF-BAPred employs a five-fold cross-validation technique to assess the potential for overfitting within the data. First, the original dataset is randomly divided into five equally sized subsets. Four of these subsets are then used as the training set, while the remaining subset serves as the validation set. This procedure is repeated five times, each time utilizing a different validation set. During the five-fold cross-validation, a larger performance gap between the training and validation sets indicates a higher risk of overfitting. The training set accuracy, validation set accuracy, training set loss, and validation set loss obtained from this process are visually represented in the supplemental figures. This visualization facilitates the identification of significant discrepancies between the performances of the training and validation sets, which may suggest overfitting. By carefully examining these cross-validation results, researchers can evaluate the generalization capability of the TF-BAPred model and make informed decisions regarding potential model refinements or the need for additional training data.

Supplementary Note 3: Model Training and Hyperparameter Configuration

Model Training: During the model training process, we divided the dataset into training, validation, and test sets with a ratio of 7:1:2. We chose the Adam optimizer due to its adaptive learning rate properties, which perform well in handling complex neural networks. The initial learning rate was set to 0.001, with a decay parameter of 1e-6. For the loss function, we used binary cross-entropy. We set the training process to run for a maximum of 100 epochs and employed an early stopping strategy to prevent overfitting. Specifically, training is stopped early if there is no significant improvement in validation loss over 10 consecutive epochs, saving computational resources and preventing model performance degradation.

Hyperparameter Configuration: In the TF-BAPred model, we fine-tuned the hyperparameters through grid search. The learning rate was initially set to 0.001 and adjusted dynamically based on the validation loss during training. For the dropout rate, we tested values between 0.2 and 0.5 and ultimately chose 0.3 to balance model fitting and generalization. The batch size was chosen to be 32 based on memory constraints and computational efficiency to ensure stable training and faster convergence. Regarding the specific configuration of TCN layers, we also used grid search for hyperparameter optimization. The number of convolutional filters was set to 128, with each filter having a size of 3. Larger kernels can capture broader patterns but may increase computational complexity, while smaller kernels are better suited for capturing finer-grained temporal relationships. We use multiple dilation rates to help the network capture temporal dependencies at different scales. Dilated convolutions enable the network to obtain a larger receptive field without increasing the number of parameters, which is particularly effective for handling sequence data with varying temporal dependencies. We use ReLU as the activation function. ReLU helps mitigate the vanishing gradient problem, accelerates training convergence, and enhances the network's ability to learn nonlinear features.

Supplementary Note 4: Assessment metrics

To assess the performance of the model, we employed commonly used evaluation metrics such as accuracy (ACC), 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:

$$ACC = \frac{TP+TN}{TP+FP+TN+FN}, \quad (1)$$

$$PREC = \frac{TP}{TP+FP}, \quad (2)$$

$$SE = \frac{TP}{TP+FP}, \quad (3)$$

$$SP = \frac{TN}{TN+FP}, \quad (4)$$

$$F1-score = \frac{2 \cdot PREC \cdot SE}{PREC+SE}, \quad (5)$$

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

where TP represents true positives, TN represents true negatives, FP represents false positives, FN represents false negatives, and PREC represents the precision of prediction. In addition, our study employed the construction of ROC curves and analysis of AUC values for evaluating model performance.

These metrics were selected because they allow for a comprehensive assessment of model performance from various perspectives, particularly in tasks involving imbalanced classes or higher requirements for predicting specific classes.

Supplementary Note 5: Residue Sparse Matrix

We will explain the construction process of RSM using a simple example.

Suppose we have a sequence DVADVMYYV, and the amino acid alphabet is {A: 0, D: 1, M: 2, V: 3, Y: 4}. Since the alphabet length is 5 and the sequence length is 9, we initialize a 5×9 matrix filled with zeros A.

Next, we begin to iterate through the peptide sequence. The integer value obtained from the amino acid mapping in the alphabet corresponds to the row position in the matrix, while the index value of the amino acid in the sequence corresponds to the column position. Ultimately, by using the mapping and index values, we can determine the position of the amino acid in the matrix and replace the 0 at that position with 1. For example, the first element of the sequence is D, which has a mapping value of 1 in the alphabet and an index value of 0 in the sequence. Therefore, its position in the matrix is (1, 0), resulting in $A_{1,0}$= 1. Following this process, when the peptide sequence has been fully traversed, the values in matrix A will be updated as follows:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Next, we perform singular value decomposition on the matrix, We multiply the left singular vectors obtained from the decomposition by their corresponding singular values and perform normalization to generate a new feature vector. The generated feature vector contains both the positional information of the amino acids in the sequence and the types of amino acids.

Supplementary Note 6: Experimental setup

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. However, when dealing with large datasets, we still faced some computational challenges, particularly in terms of memory consumption and training time. To address these issues, we optimized the batch size and applied techniques such as model parameter sharing and gradient accumulation to minimize memory usage and improve training efficiency. Nonetheless, when handling very large datasets, the training time remains relatively long, which may become a bottleneck for further optimization in certain cases.

Supplemental Figures

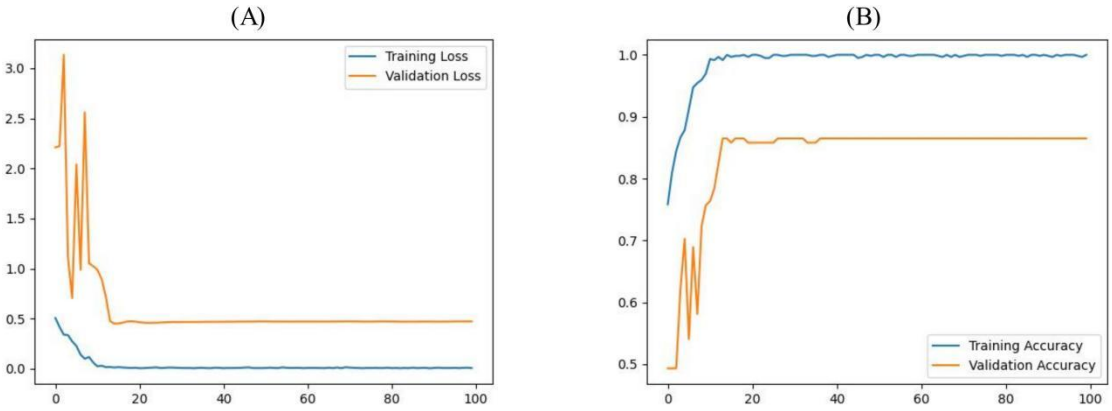

Figure S1. Learning curve of TF-BAPred on the ACP740. (A) The training loss and validation loss. (B) The training accuracy and validation accuracy.

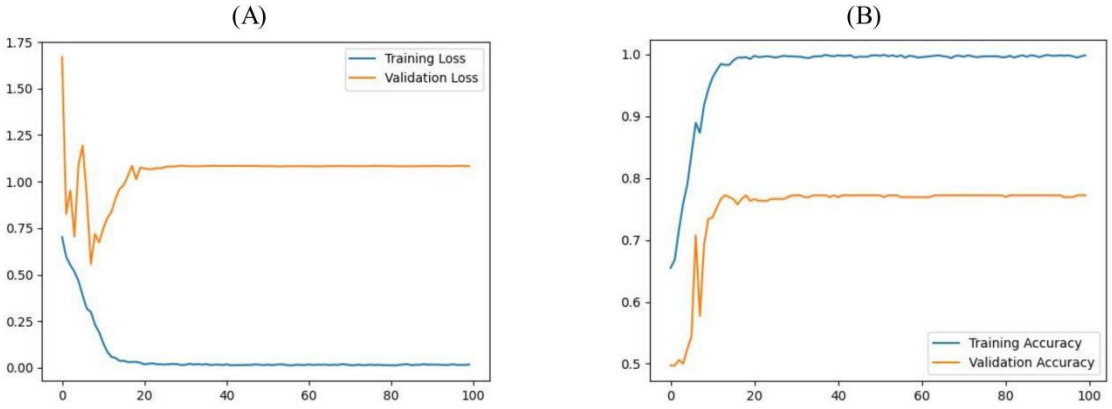

Figure S2. Learning curve of TF-BAPred on the ACPmain. (A) The training loss and validation loss. (B) The training accuracy and validation accuracy.

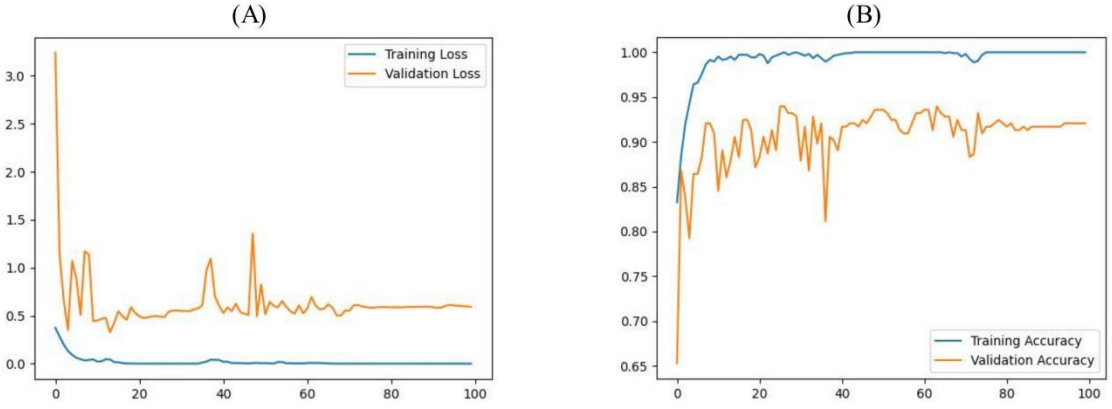

Figure S3. Learning curve of TF-BAPred on the Veltri's dataset. (A) The training loss and validation loss. (B) The training accuracy and validation accuracy.

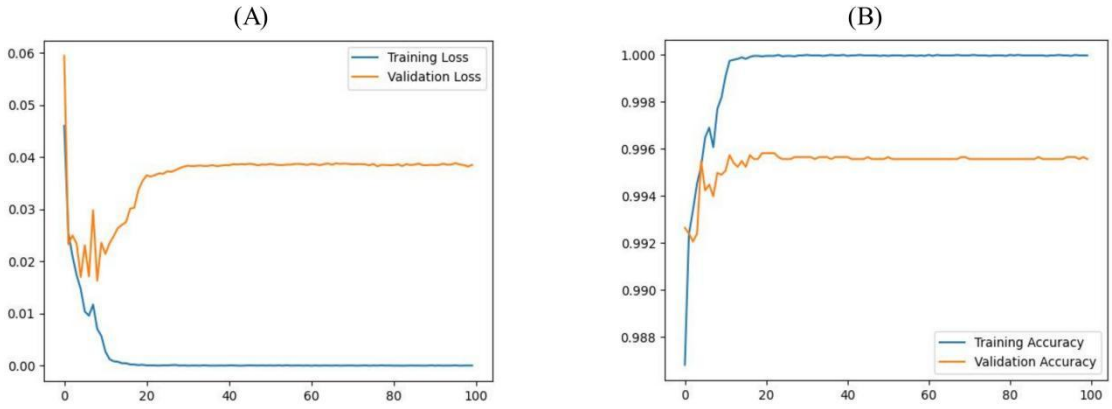

Figure S4. Learning curve of TF-BAPred on the Ma's dataset. (A) The training loss and validation loss. (B) The training accuracy and validation accuracy.

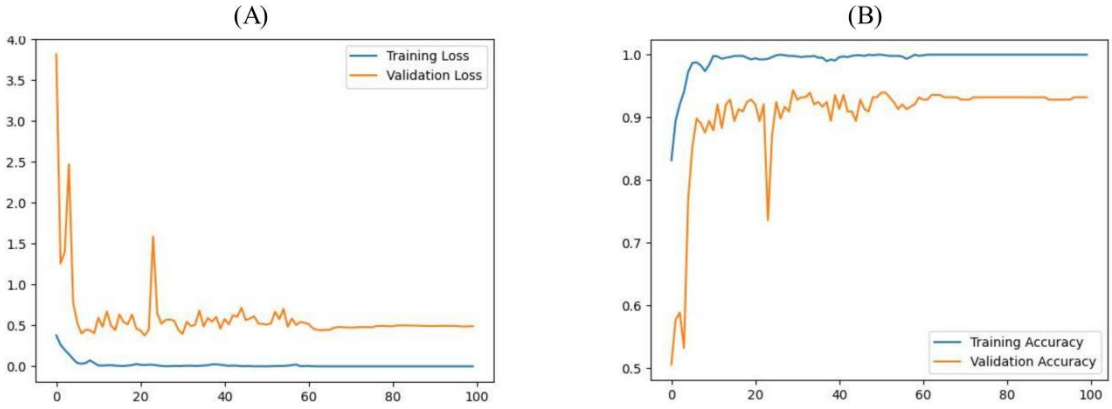

Figure S5. Learning curve of TF-BAPred on the CPP924 dataset. (A) The training loss and validation loss. (B) The training accuracy and validation accuracy.

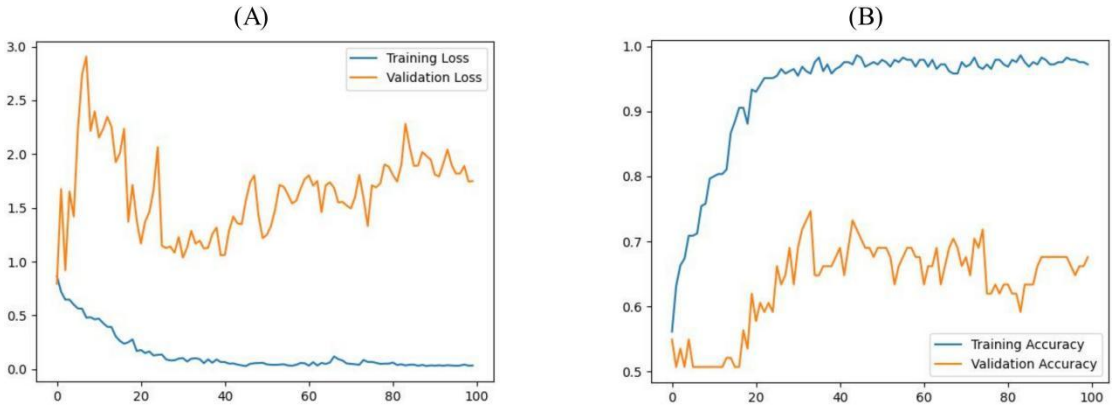

Figure S6. Learning curve of TF-BAPred on the CPPsite3 dataset. (A) The training loss and validation loss. (B) The training accuracy and validation accuracy.

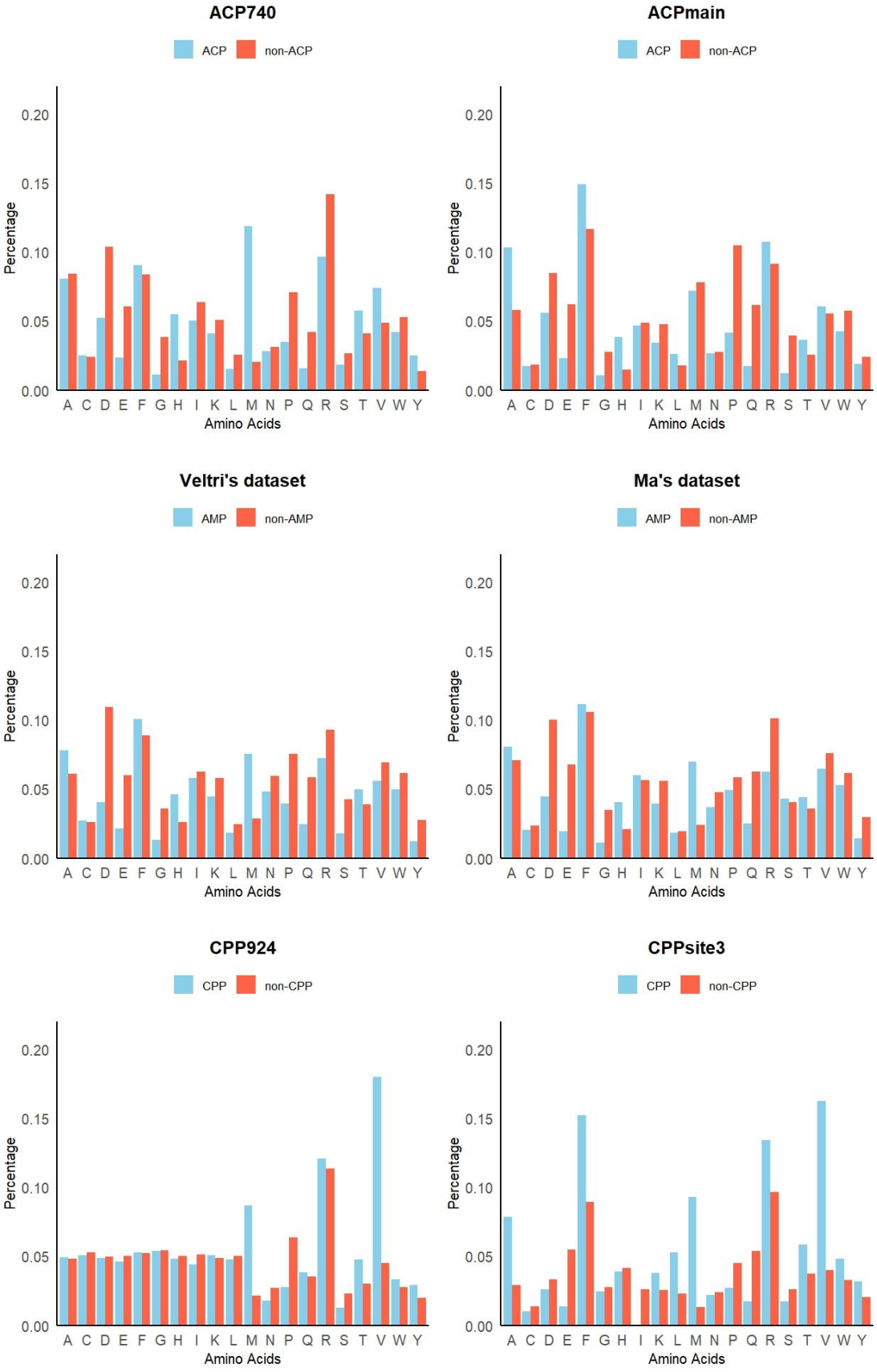

Figure S7. Composition of amino acids in positive and negative samples on six datasets.

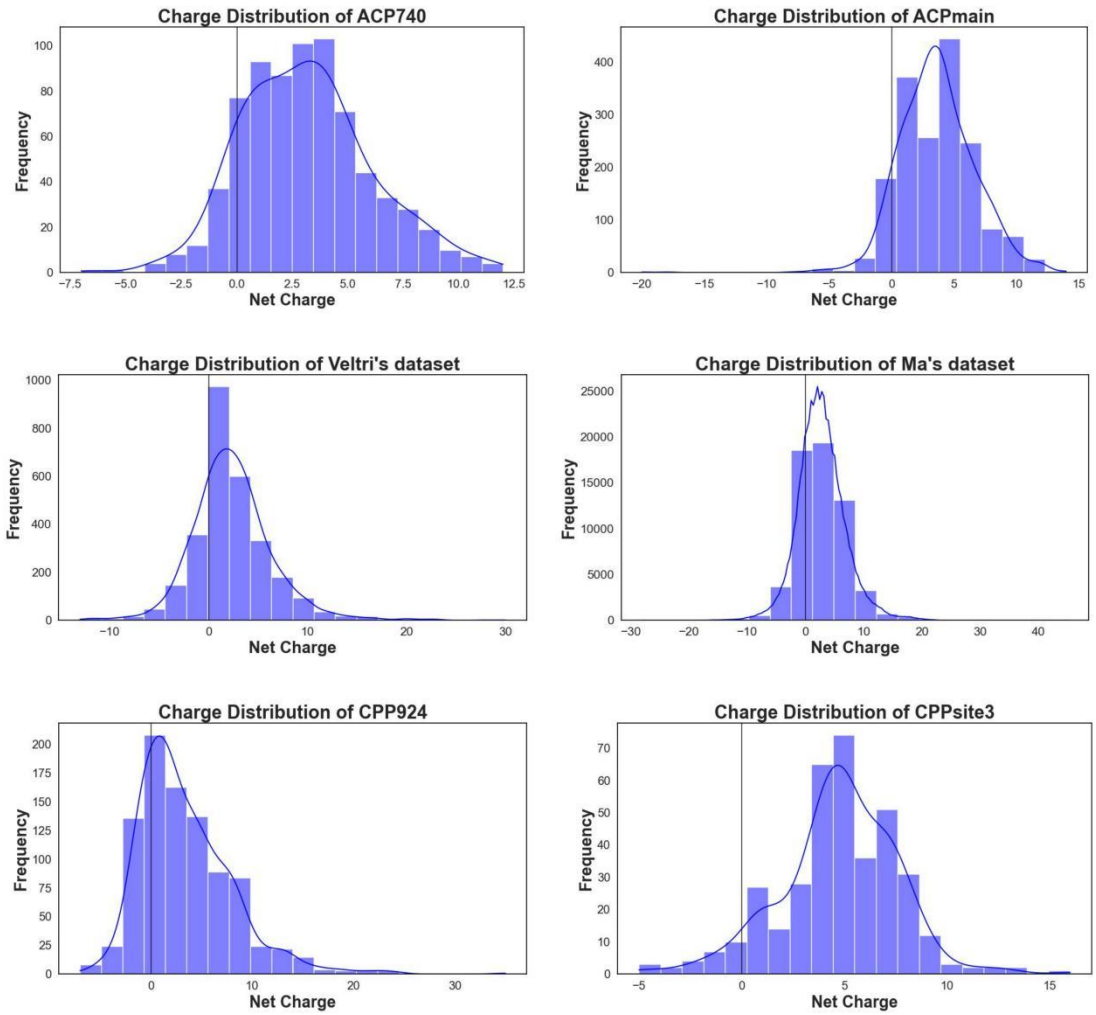

Figure S8. Charge distribution of amino acids on six datasets.

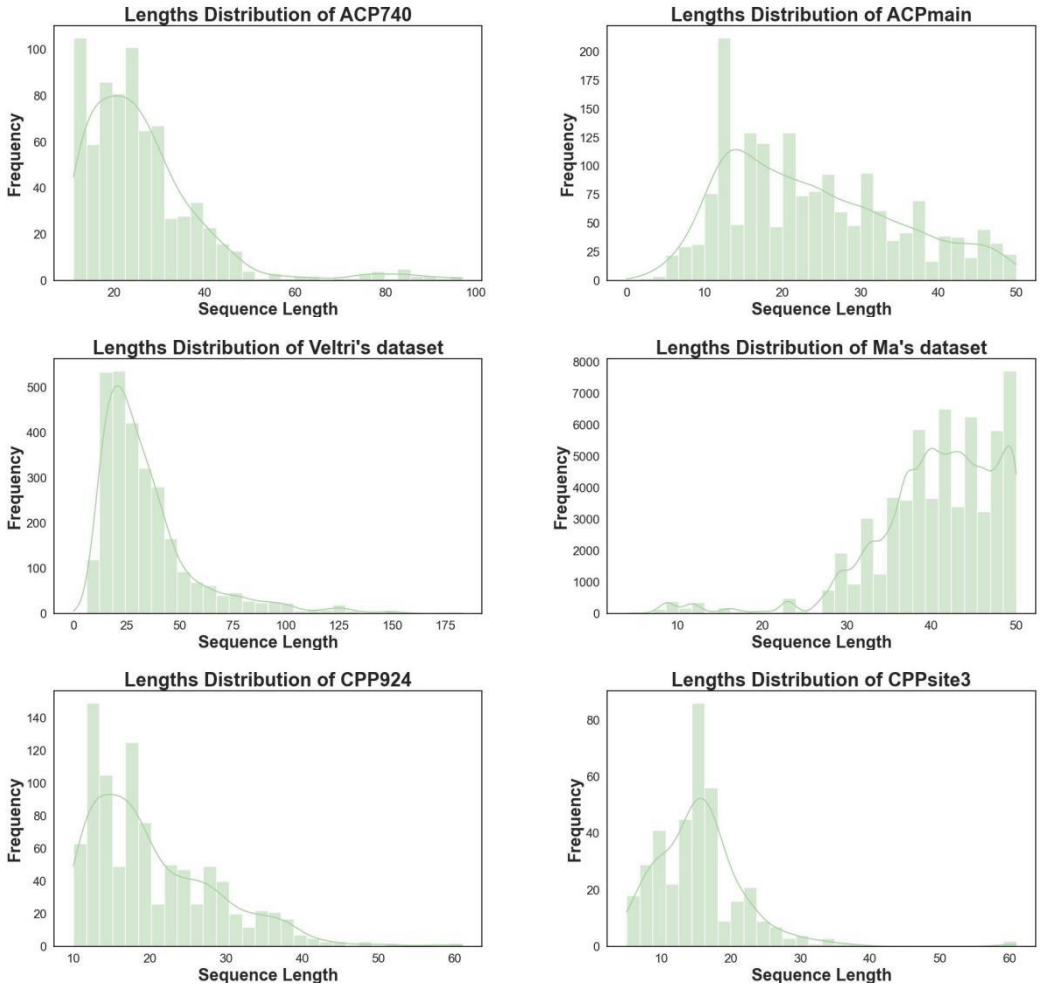

Figure S9. Distribution of Peptide Sequence Lengths on six datasets.

Supplemental Tables

Table S1. Evaluation of the effectiveness of TCN and FVG on ACP740.

Method	ACC	SE	SP	F1-score	MCC
CNN	0.755	0.693	0.626	0.722	0.532
LSTM	0.741	0.826	0.653	0.766	0.488
TCN	0.789	0.693	0.888	0.769	0.592
CNN+FVG	0.810	0.813	0.805	0.813	0.618
LSTM+FVG	0.789	0.653	**0.930**	0.789	0.605
TCN+FVG	**0.889**	**0.907**	0.871	**0.881**	**0.779**

Table S2. Evaluation of the effectiveness of TCN and FVG on ACPmain.

Method	ACC	SE	SP	F1-score	MCC
CNN	0.680	0.686	0.674	0.682	0.360
LSTM	0.680	0.494	0.866	0.607	0.388
TCN	0.715	**0.790**	0.839	0.769	0.435
CNN+FVG	0.706	0.720	0.691	0.710	0.412
LSTM+FVG	0.683	0.682	**0.883**	**0.806**	0.399
TCN+FVG	**0.760**	0.721	0.797	0.782	**0.521**

Table S3. Evaluation of the effectiveness of TCN and FVG on Veltri's dataset.

Method	ACC	SE	SP	F1-score	MCC
CNN	0.860	0.888	0.831	0.875	0.720
LSTM	0.857	0.854	0.891	0.861	0.722
TCN	0.865	0.897	0.872	0.893	0.770
CNN+FVG	0.884	0.880	0.889	0.877	0.769
LSTM+FVG	0.893	0.903	0.880	0.886	0.787
TCN+FVG	**0.901**	**0.906**	**0.896**	**0.905**	**0.802**

Table S4. Evaluation of the effectiveness of TCN and FVG on Ma's dataset.

Method	ACC	SE	SP	F1-score	MCC
CNN	0.992	0.672	0.993	0.849	0.752
LSTM	0.993	0.765	0.990	0.856	0.806
TCN	0.994	0.779	0.995	0.871	0.821
CNN+FVG	0.993	0.746	0.994	0.880	0.807
LSTM+FVG	0.994	0.788	0.995	0.864	0.822
TCN+FVG	**0.995**	**0.836**	**0.996**	**0.886**	**0.859**

Table S5. Evaluation of the effectiveness of TCN and FVG on CPP924.

Method	ACC	SE	SP	F1-score	MCC
CNN	0.818	0.954	0.681	0.750	0.661
LSTM	0.810	0.893	0.727	0.787	0.681
TCN	0.868	0.909	0.807	0.810	0.702
CNN+FVG	0.864	0.909	0.818	0.833	0.730
LSTM+FVG	0.825	**0.955**	0.666	0.754	0.663
TCN+FVG	**0.902**	0.906	**0.905**	**0.916**	**0.841**

Table S6. Evaluation of the effectiveness of TCN and FVG on CPPcite3.

Method	ACC	SE	SP	F1-score	MCC
CNN	0.657	0.552	0.763	0.700	0.323
LSTM	0.579	0.552	0.605	0.583	0.358
TCN	0.684	0.605	0.763	0.679	0.373
CNN+FVG	0.758	0.553	0.763	0.700	0.523
LSTM+FVG	0.744	0.595	0.694	0.628	0.521
TCN+FVG	**0.766**	**0.731**	**0.800**	**0.755**	**0.533**

Note: Ex represents except, and the best values are shown in the boldface font.

Table S7. Performance comparison of different feature combinations on six datasets.

Dataset	Component	ACC	SE	SE	F1-score	MCC
ACP740	-Ex ACC	**0.877**	0.906	0.847	**0.895**	0.755
	-Ex DPC	0.878	0.880	0.877	0.886	**0.757**
	-Ex BPF	0.876	0.851	**0.881**	0.863	0.742
	-Ex CKSAAGP	0.856	**0.909**	0.803	0.863	0.716
	-Ex RSM	0.845	0.822	0.867	0.841	0.690
	-Ex FVG	0.811	0.759	0.871	0.812	0.629

Table S8. Performance comparison of different feature combinations on six datasets.

Dataset	Component	ACC	SE	SE	F1-score	MCC
ACPmain	-Ex ACC	0.769	0.757	0.780	0.766	0.540
	-Ex DPC	**0.784**	**0.773**	0.794	**0.781**	**0.568**
	-Ex BPF	0.772	0.738	**0.806**	0.763	0.545
	-Ex CKSAAGP	0.760	0.721	0.797	0.750	0.520
	-Ex RSM	0.756	0.727	0.785	0.747	0.513
	-Ex FVG	0.736	0.672	0.762	0.717	0.476

Table S9. Performance comparison of different feature combinations on six datasets.

Dataset	Component	ACC	SE	SE	F1-score	MCC
Veltri's dataset	-Ex ACC	0.910	**0.920**	0.901	0.911	0.821
	-Ex DPC	**0.917**	0.917	**0.917**	**0.919**	**0.825**
	-Ex BPF	0.908	0.914	0.915	0.908	0.817
	-Ex CKSAAGP	0.905	0.901	0.908	0.904	0.810
	-Ex RSM	0.884	0.871	0.899	0.883	0.770
	-Ex FVG	0.877	0.896	0.860	0.873	0.754

Table S10. Performance comparison of different feature combinations on six datasets.

Dataset	Component	ACC	SE	SE	F1-score	MCC
Ma's dataset	-Ex ACC	0.992	0.860	0.993	0.806	0.811
	-Ex DPC	0.995	0.865	0.995	0.822	0.818
	-Ex BPF	**0.996**	**0.868**	**0.996**	**0.825**	**0.821**
	-Ex CKSAAGP	0.993	0.862	0.994	0.809	0.814
	-Ex RSM	0.990	0.853	0.991	0.812	0.810
	-Ex FVG	0.987	0.844	0.988	0.798	0.796

Table S11. Performance comparison of different feature combinations on six datasets.

Dataset	Component	ACC	SE	SE	F1-score	MCC
CPP924	-Ex ACC	0.918	0.917	0.920	0.918	0.837
	-Ex DPC	**0.928**	**0.920**	**0.937**	**0.928**	**0.857**
	-Ex BPF	0.920	0.904	0.935	0.918	0.840
	-Ex CKSAAGP	0.917	0.911	0.924	0.917	0.835
	-Ex RSM	0.910	0.908	0.911	0.909	0.828
	-Ex FVG	0.894	0.892	0.906	0.893	0.825

Table S12. Performance comparison of different feature combinations on six datasets.

Dataset	Component	ACC	SE	SE	F1-score	MCC
CPPcite3	-Ex ACC	**0.768**	0.729	0.695	0.711	0.511
	-Ex DPC	0.758	0.716	**0.741**	**0.728**	**0.535**
	-Ex BPF	0.762	0.749	0.703	0.725	0.531
	-Ex CKSAAGP	0.741	0.722	0.718	0.689	0.503
	-Ex RSM	0.721	**0.758**	0.709	0.683	0.496
	-Ex FVG	0.684	0.743	0.728	0.667	0.473

Table S13. Five-Fold Cross-Validation Results of TF-BAPred on ACP740.

Dataset	Fold set	ACC	SE	SP	F1-score	MCC
ACP740	1	0.885	0.906	0.863	0.890	0.771
	2	0.872	0.920	0.822	0.880	0.746
	3	0.885	0.893	0.877	0.889	0.770
	4	0.905	0.906	0.904	0.906	0.811
	5	0.878	0.901	0.876	0.892	0.757

Table S14. Five-Fold Cross-Validation Results of TF-BAPred on Veltri's dataset.

Dataset	Fold set	ACC	SE	SP	F1-score	MCC
Veltri's dataset	1	0.932	0.918	0.947	0.931	0.865
	2	0.936	0.925	0.946	0.933	0.872
	3	0.940	0.932	0.946	0.937	0.879
	4	0.940	0.925	0.954	0.953	0.880
	5	0.954	0.940	0.969	0.969	0.910

Table S15. Five-Fold Cross-Validation Results of TF-BAPred on CPP924.

Dataset	Fold set	ACC	SE	SP	F1-score	MCC
CPP924	1	0.929	0.946	0.913	0.931	0.859
	2	0.923	0.924	0.924	0.924	0.848
	3	0.929	0.923	0.934	0.927	0.859
	4	0.921	0.926	0.915	0.922	0.841
	5	0.918	0.917	0.919	0.918	0.837

Table S16. GPU Time for Running 100 Epochs Across Different Datasets for Various Methods (Unit of Measurement: Seconds).

	ACP740	ACPmain	Veltri's dataset	Ma's dataset	CPP924	CPPsite3
ACP-DL	70	54	46	7500	57	50
ACP-check	415	554	939	26900	171	246
pLMFPPred	983	2465	3700	88320	1375	498
AMP-EBiLSTM	198	188	696	54100	300	145
DNN	184	927	392	44500	168	194
Ma’s method	160	498	531	45200	250	92
CPPred-RF	261	92	271	16800	178	56
TF-BAPred	229	667	891	35786	314	123

Table S17. Computational Memory Requirements of Different Methods on Various Datasets(Unit of Measurement: MegaByte).

	ACP740	ACPmain	Veltri's dataset	Ma's dataset	CPP924	CPPsite3
ACP-DL	345.29	353.21	337.29	357.50	347.49	341.30
ACP-check	415.63	444.44	477.61	448.96	449.62	400.61
pLMFPPred	559.88	553.37	551.55	552.52	543.32	550.87
AMP-EBiLSTM	377.17	359.88	422.52	392.22	375.02	349.88
DNN	349.86	349.50	353.44	349.27	343.50	349.40
Ma’s method	352.86	356.67	357.07	357.67	353.26	351.77
CPPred-RF	318.30	334.00	320.78	339.60	322.10	313.44
TF-BAPred	432.61	497.86	535.31	506.92	466.87	421.75