Molecular Therapy
Nucleic Acids
Original Article

DeepACP: A Novel Computational Approach for Accurate Identification of Anticancer Peptides by Deep Learning Algorithm

Lezheng Yu,[1,5] Runyu Jing,[2,5] Fengjuan Liu,[3] Jiesi Luo,[4] and Yizhou Li[2]

[1]School of Chemistry and Materials Science, Guizhou Education University, Guiyang 550018, China; [2]College of Cybersecurity, Sichuan University, Chengdu 610065, China; [3]School of Geography and Resources, Guizhou Education University, Guiyang 550018, China; [4]Department of Pharmacology, School of Pharmacy, Southwest Medical University, Luzhou 646000, Sichuan, China

Cancer is one of the most dangerous diseases to human health. The accurate prediction of anticancer peptides (ACPs) would be valuable for the development and design of novel anticancer agents. Current deep neural network models have obtained state-of-the-art prediction accuracy for the ACP classification task. However, based on existing studies, it remains unclear which deep learning architecture achieves the best performance. Thus, in this study, we first present a systematic exploration of three important deep learning architectures: convolutional, recurrent, and convolutional-recurrent networks for distinguishing ACPs from non-ACPs. We find that the recurrent neural network with bidirectional long short-term memory cells is superior to other architectures. By utilizing the proposed model, we implement a sequence-based deep learning tool (DeepACP) to accurately predict the likelihood of a peptide exhibiting anticancer activity. The results indicate that DeepACP outperforms several existing methods and can be used as an effective tool for the prediction of anticancer peptides. Furthermore, we visualize and understand the deep learning model. We hope that our strategy can be extended to identify other types of peptides and may provide more assistance to the development of proteomics and new drugs.

INTRODUCTION

Cancer has become one of the biggest threats to human health and is the leading cause of death in developed countries and the second most common cause in developing countries.[1] 18.1 million new cases and 9.6 million cancer deaths were estimated to have occurred in 2018 by the International Agency for Research on Cancer (IARC).[2] Furthermore, the four most commonly diagnosed cancer types were lung, breast, prostate, and colorectal cancer, while the four leading cancer types causing mortality were lung, colorectal, stomach, and liver cancer. There are many advanced clinical methods for the treatment of cancer, such as surgery, radiotherapy, chemotherapy, and targeted therapy. Unfortunately, these traditional methods are not only expensive but also unsafe, with many serious side effects.[3–5] Moreover, it is known that cancer cells may develop resistance to traditional treatment methods and conventional drugs.[6] To combat this deadly disease and prolong the lives of patients, more and more attention is being paid to the discovery and design of novel anticancer agents in recent years.

In the past few decades, peptides have emerged as promising therapeutic agents for cancer treatment. Because they are safer, are more selective, cost less, and are more tolerable than traditional treatment methods, peptide-based therapeutics are considered to be superior treatment strategies. Therefore, anticancer peptides (ACPs) have become potential anticancer agents.[7,8] As a subset of antimicrobial peptides, ACPs have shown the potential to inactivate various cancer cells without affecting normal cells,[9] though the mechanisms by which they affect cancer cells are not entirely clear. To further understand the anticancer mechanisms of ACPs and develop new anticancer drugs, it is particularly important to rapidly and effectively identify various ACPs. There have been many experimental methods for identification and development of novel ACPs, but they are usually laborious, expensive, time consuming, and hard to achieve in a high-throughput manner.[10–12] Therefore, it is very desirable to develop computational methods based on machine learning algorithms to identify ACPs.

Over the past few years, more than a dozen computational methods have been proposed for the identification of ACPs, and more and more machine learning algorithms are used to build the predictors, such as support vector machines (SVMs),[12–19] random forest (RF),[20,21] the K-nearest neighbor (KNN), and their different ensemble methods.[22–26] Though some of these methods have offered relatively high accuracies and robustness, it is still a challenge for these conventional machine learning methods to choose the appropriate descriptors that represent the sequences of ACPs. To overcome this

Received 2 April 2020; accepted 6 October 2020;
https://doi.org/10.1016/j.omtn.2020.10.005.

[5]These authors contributed equally to this work.

Correspondence: Jiesi Luo, Department of Pharmacology, School of Pharmacy, Southwest Medical University, Luzhou 646000, Sichuan, China.
E-mail: ljs@swmu.edu.cn
Correspondence: Lezheng Yu, School of Chemistry and Materials Science, Guizhou Education University, Guiyang 550018, China.
E-mail: xinyan_scu@126.com

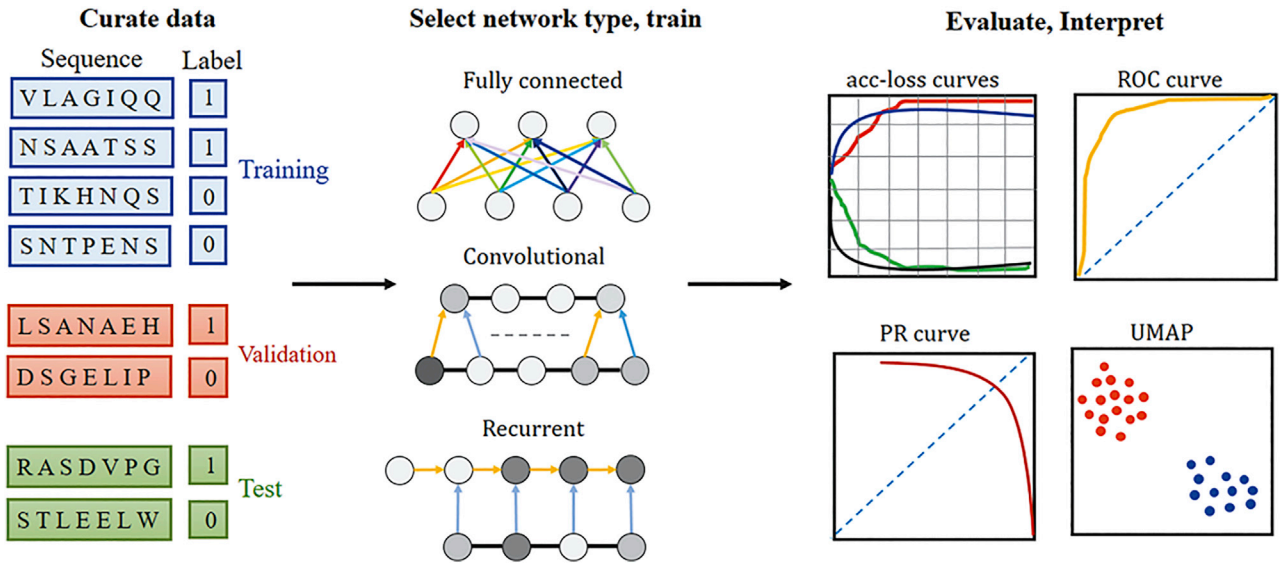

Figure 1. Deep Learning Workflow in Anticancer Peptide Prediction

limitation, deep learning algorithms are used to further improve the prediction accuracy and robustness.[27–29] Wu et al.[27] proposed a computational model based on convolutional neural networks and word2vec to predict therapeutic peptides in a highly efficient manner. Grisoni et al.[28] designed anticancer peptides by training a recurrent neural network with long short-term memory cells. Yi et al.[29] also proposed a long short-term neural network model to effectively predict novel anticancer peptides by integrating binary profile features and a *k*-mer sparse matrix of the reduced amino-acid alphabet. In addition to designing ACPs, deep learning algorithms have shown great potential in the discovery of other anticancer drugs, such as prediction of drug-target interaction[30–32] and drug-induced toxicities.[33–35] However, current deep learning-based algorithms have been limited to a particular architecture, and it is still not clear which architecture's performance is best when detecting ACPs.

In the present study, we evaluate the performance of different deep learning architectures for ACP prediction to solve the aforementioned issues with existing methods. Three deep learning architectures—a convolutional neural network (CNN), a recurrent neural network (RNN) with bidirectional long short-term memory cells (biLSTMs), and a CNN-RNN—are used to construct the predictors. Experimental results using the benchmark dataset show that the RNN architecture affords the best overall prediction performance. The RNN architecture can predict ACPs with, on average, 79.2% recall, 89.5% precision (PRE), 83.9% F value, 84.9% accuracy (ACC), and a Matthew's correlation coefficient (MCC) of 0.704 when tested on the test set 10 times. When compared with the CNN architecture, the RNN predicts ACPs with a 4.8%, 1.6%, and 2.2% improvement in PRE, F value, and ACC, respectively, and an MCC of 0.047. The results indicate that the deep RNN is better suited to recognizing sequence motifs of varying lengths, such as ACPs, than traditional feed-forward neural networks. Therefore, we propose DeepACP, a deep learning method combined with the RNN model and amino-acid character embedding for improving ACP prediction. When performed on the independent test dataset, it is found that the prediction performance of DeepACP is also comparable to those of popular existing methods. We further visualize the neurons representing each peptide in the training and test datasets to understand why the models make their predictions. DeepACP facilitates the understanding of anticancer mechanisms of ACPs and the development of novel anti-cancer drugs.

RESULTS

Overall Experimental Procedure of This Work

Because it is still not clear which deep learning architecture gives the best performance for predicting ACPs, we first used a deep learning classification tool (autoBioSeqpy) to design and evaluate three selected state-of-the-art algorithms (RNN, CNN, and their combination, CNN-RNN) on a benchmark dataset including 250 ACPs and 250 non-ACPs. To guard against overfitting, we randomly split the benchmark dataset into three subsets: training, validation, and test sets. The training set is used for learning the model parameters, the validation set is used to select the best model, and the test set is kept aside to estimate the generalization performance[36] (Figure 1). We repeated this procedure 10 times and evaluated the predictive capability of the trained model using five metrics: recall, PRE, ACC, F value, and MCC. We also designed a collection of different architecture variants by varying the number of cells in the RNN as well as the number of convolution kernels and network layers in CNNs to select the optimal architecture. We further evaluated the performance of three deep learning architectures and other existing algorithms on an independent test dataset. Finally, to make the

Table 1. Performance Comparison of RNN and CNN-RNN Architectures with Different Number of Cells

Architecture and Cell Numbers	ACC (%)	F Value (%)	Recall (%)	PRE (%)	MCC
RNN					
32	80.4	78.4	72.0	87.1	0.621
64	84.9	83.9	79.2	89.5	0.704
128	82.3	81.4	77.8	85.7	0.651
256	83.7	83.2	81.2	85.7	0.677
CNN-RNN					
32	78.8	78.5	78.2	79.7	0.583
64	81.7	80.9	78.6	84.2	0.640
128	80.5	79.5	77.0	83.0	0.616
256	79.1	78.1	74.8	82.8	0.590

ACC, accuracy; PRE, precision; MCC, Matthew's correlation coefficient; RNN, recurrent neural network; CNN-RNN, convolutional-recurrent neural network.

Table 2. Performance Comparison of CNN and CNN-RNN Architectures with Different Number of Convolution Kernels

Architecture and Kernel Numbers	ACC (%)	F Value (%)	Recall (%)	PRE (%)	MCC
CNN					
50	82.1	82.0	81.8	82.7	0.645
100	82.5	82.8	84.4	81.6	0.653
150	82.7	82.3	80.4	84.7	0.657
200	81.6	81.5	81.2	82.0	0.634
250	80.4	80.3	80.0	81.0	0.611
CNN-RNN					
50	77.9	77.1	75.0	80.6	0.566
100	81.1	80.3	77.8	83.8	0.629
150	81.7	80.9	78.6	84.2	0.640
200	81.3	80.4	76.8	85.4	0.635
250	81.2	81.0	80.8	81.8	0.628

ACC, accuracy; PRE, precision; MCC, Matthew's correlation coefficient; CNN, convolutional neural network; CNN-RNN, convolutional-recurrent neural network.

model understandable, we extracted the hidden layer representations to visualize the ACPs and non-ACPs learned by the RNN model. All detailed experimental results are shown in the following sections.

Selecting the Number of Cells in RNNs and CNN-RNNs

The number of hidden cells is one of the most important architecture variants that requires careful tuning for training a RNN.[37] In this section, we examined the effect of the number of cells on five metrics by comparing the values for 32, 64, and 128 cells to those for 256 cells (Table 1). We observed that, when the number of cells is set to 64, the RNN architecture achieves the best performance, attaining an average ACC, F value, recall, PRE, and MCC of 84.9%, 83.9%, 79.2%, 89.5%, and 0.704, respectively. The number of cells has a similar effect on the CNN-RNN. For the proposed architecture, when the cell number equals 64, it obtains the best values of all metrics. The average ACC, F value, recall, PRE, and MCC are 81.7%, 80.9%, 78.6%, 84.2%, and 0.640, respectively. Therefore, the number of cells is chosen as 64 for the RNN and for the CNN-RNN in the following experiments.

Selecting the Number of Convolution Kernels and Network Layers in CNNs and CNN-RNNs

wIn this section, we studied the effect of the numbers of convolutional kernels and network layers in CNNs and CNN-RNNs. We first evaluated the performance of two proposed architectures on the benchmark dataset by gradually varying the number of convolution kernels from 50 to 100, to 150, 200, and 250. As we can see from Table 2, when the number of convolutional kernels is specified at 150, both architectures can achieve their best results. Both deep learning architectures achieve better performance as the number of convolution kernels increases (from 50 to 150); afterwards, the performance decreases. Specifically, the CNN achieves 0.645, 0.653, 0.657, 0.634, and 0.611 for the MCC on 50, 100, 150, 200, and 250 kernels, respectively. The MCCs are 0.566, 0.629, 0.640, 0.635, and 0.628 for the CNN-RNN on 50, 100, 150, 200, and 250 kernels, respectively. With the best performing number of kernels, we further explored the effect of network depth from one to four convolutional and pooling layers on prediction performance. Table 3 shows the values of ACC, F value, recall, PRE, and MCC by specifying different numbers of network layers, and it is clear that the CNN and CNN-RNN architectures with 150 kernels and one layer of depth provide the best performance.

Performance Comparison of Different Deep Learning Architectures on the Benchmark Dataset

Based on the aforementioned findings, we evaluated the discriminative performance of the three deep learning architectures using their best parameters. All three architectures achieve better than random performance on ACP classification. In terms of model ACC, the CNN-RNN demonstrates the worst performance (average ACC = 81.7%) and the CNN follows with an ACC of 82.7%. When considering average ACC, the RNN outperforms other architectures with an ACC value of 84.9%. Given the positive cases in which ACPs are present and the negative cases in which no ACPs are present as defined in the benchmark dataset, based on predictive scores for each deep learning architectures, we used area under the curve (AUC) of receiver operating characteristic (ROC) curves to further assess the predictive performance, which is a common measurement independent from the threshold value in each algorithm (Figure 2). The RNN is the most accurate (mean AUC = 0.920), followed by the CNN (0.903). The CNN-RNN has the lowest ACC (mean AUC = 0.871). In addition to ROC, the ACC-loss curves and PRE-recall (PR) curves for these three architectures are shown in Figures S1 and S2.

Table 3. Performance Comparison of CNN and CNN-RNN Architectures with Different Number of Convolutional and Pooling Layers

Architecture and Layer Numbers	ACC (%)	F Value (%)	Recall (%)	PRE (%)	MCC
CNN					
1	82.7	82.3	80.4	84.7	0.657
2	79.6	78.6	76.8	82.3	0.601
3	76.8	76.7	76.6	77.2	0.539
4	69.1	71.9	79.2	66.6	0.396
CNN-RNN					
1	81.7	80.9	78.6	84.2	0.640
2	78.3	77.9	76.8	80.5	0.576
3	74.9	74.2	73.0	77.6	0.509
4	72.6	73.9	77.8	72.1	0.468

ACC, accuracy; PRE, precision; MCC, Matthew's correlation coefficient; CNN, convolutional neural network; CNN-RNN, convolutional-recurrent neural network.

Performance Comparison with Existing Predictors on the Independent Test Dataset

To further validate deep learning performance for predicting ACPs, we conducted experiments on an independent test dataset with 82 ACPs and 82 non-ACPs. In addition to the deep learning algorithms, four existing methods were selected for comparison with the predictive performances, including AntiCP,[13] Hajisharifi et al.'s method,[14] iACP,[16] and ACPred-FL.[12] Significantly, two models of AntiCP (AntiCP_AAC and AntiCP_DPC) were chosen for comparison in this work. For a fair comparison, the same training and independent test datasets were used to train and test all approaches. Table 4 shows the prediction results of all five methods on the independent test dataset, and the detailed results of the aforementioned four methods were directly accessed from the study describing ACPred-FL.[12] Among the deep-learning-based algorithms, the RNN architecture had the best overall prediction performance and yielded the highest scores of PRE (86.5%), ACC (82.9%), F value (82.0%), recall (78.0%), and MCC (0.662). Meanwhile, the CNN-RNN predicted the fewest numbers of ACPs (59), and the CNN identified the fewest numbers of non-ACPs (65). Because its performance was the strongest, the RNN architecture was chosen as the final predictor to further compare with other computational methods and is named DeepACP in this study.

It is observed that DeepACP correctly identifies the second largest numbers of ACPs (64) and non-ACPs (72). The overall prediction performance of DeepACP is remarkably superior to those of AntiCP, Hajisharifi et al.'s[14] method, and iACP, which provides an improvement of 2.9%–8.7%, 3.6%–10.9%, 3.6%–8.5%, 3.8%–9.3%, and 7.4%–17.1% on PRE, recall, ACC, F value, and MCC, respectively. As a result of observation, the precise projections of DeepACP are less than those of ACPred-FL (66 and 79), so the overall prediction performance of DeepACP is slightly worse than that of ACPred-FL.

Visualizing and Understanding the Deep Learning Model

We examined the internal features learned by the best performing RNN model using the Uniform Manifold Approximation and Projection (UMAP)[38] (Figure 3). Each point represents a peptide sequence projected from a 128-dimensional output of the RNN biLSTM layer into two dimensions. The points of the same peptide classes were clustered together. For the training dataset, the ACPs (Figure 3, blue dots) clustered on the right in contrast to non-ACPs (red dots), which clustered on the left. Similarly, ACPs clustered across from non-ACPs in the independent test dataset. These results indicate that the RNN architecture has learned the discriminative feature for ACP classification.

DISCUSSION

The ultimate goal of this study is to construct a deep neural network model that predicts whether each peptide is relevant for anticancer or non-anticancer activity, using only the primary sequence as an input. To achieve this goal, we first optimized some important parameters of different deep learning architectures, including the number of cells in RNNs and CNN-RNNs, and the number of convolution kernels and network layers in CNNs and CNN-RNNs. In the latter case, the experimental results show that, compared to a single convolutional and pooling architecture, adding multiple convolutional and pooling layers does not provide improved performance. While the network layer increased gradually, the MCC of the deep learning architecture decreased drastically, which indicates that the network depth has a negative effect on the model performance. We reason that it is possible that deeper architectures require larger amounts of annotated data and more information about peptide sequence to be effectively trained.

Subsequently, we used the benchmark dataset to assess the predictive performance of different deep learning architectures. Except for the 10-time test, we used 5-fold cross-validation on the benchmark dataset to further evaluate the prediction performance of the three deep learning architectures. The detailed results are shown in Table S1. As can be seen from this table, the RNN architecture yields the best overall prediction performance and achieves the highest average scores of 83.4% ± 4.7% for recall, 87.1% ± 6.2% for PRE, 84.9% ± 4.2% for F value, 85.2% ± 4.5% for ACC, and 0.708 ± 0.089 for MCC. This cross-validation result is consistent with that found by using the 10-time test, which reconfirms that the RNN architecture trained on the benchmark dataset has the best performance.

When comparing DeepACP with four other state-of-the-art methods, the overall prediction performance of our method is better than most existing methods but slightly lower than that of ACPred-FL. The most probable reason for this gap in performance is that the size of the training dataset is very small; this causes insufficient training of the deep learning model. In addition, deep learning algorithms are better at processing long sequences, while the vast majority of ACPs just contain 5–30 amino acids.[39]

We also compared DeepACP with another deep learning algorithm, ACP-DL, by using the two benchmark datasets (ACP740 and

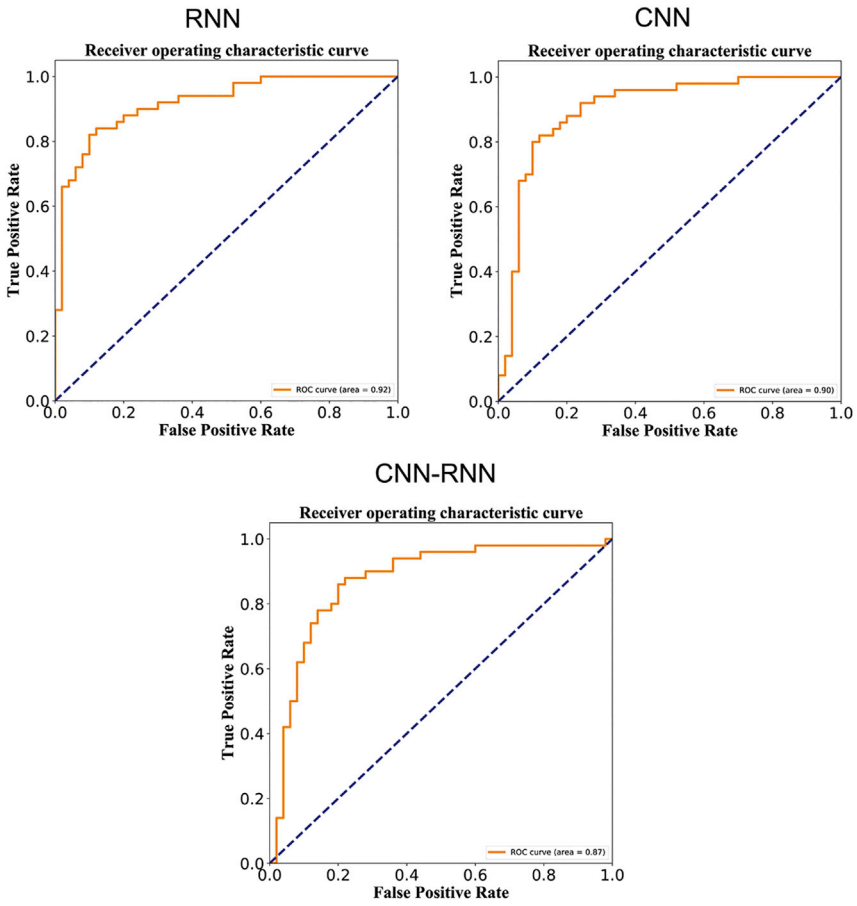

Figure 2. Receiver Operating Characteristic (ROC) Curves for Different Deep Learning Architectures

ACP240) from Yi et al.'s[29] study. The detailed prediction results of DeepACP are listed in Tables S2 and S3. For the ACP740 dataset, the overall prediction performance of DeepACP is slightly better than that of ACP-DL, which predicts ACPs with a 3.7%, 0.4%, and 0.011 improvement in mean recall, ACC, and MCC, respectively. For the ACP240 dataset, DeepACP achieves a slightly higher average recall (89.0%) and PRE (80.7%) than ACP-DL (84.6% and 80.3%, respectively) but a slightly lower average ACC (82.3%) and MCC (0.648) than ACP-DL (85.4% and 0.714, respectively). This may be caused by the small size of the ACP240 dataset, which only contains 129 experimentally validated anticancer peptide samples and 111 AMPs without anticancer functions. Overall, the prediction performance of DeepACP is comparable to that of ACP-DL but provides support for why we choose the RNN with biLSTMs to build the classifier for identification of anticancer peptides.

In this paper, we propose a novel computational approach called DeepACP to distinguish ACPs from various peptide sequences using the deep learning technique. Unlike in previous studies, three deep learning algorithms are first used to build predictors for rapidly and efficiently identifying ACPs, including the RNN, the CNN, and the CNN-RNN. Meanwhile, amino-acid character embedding is used to characterize peptide sequences. After investigating the prediction results of different deep learning architectures for the benchmark dataset by testing 10 times, it was found that the best model is constructed by the RNN. When applied to the independent test dataset, the prediction performance of the RNN model is comparable to those of other existing methods, which indicates that deep learning algorithm has great potential in peptide prediction. We believe that, by adding more ACPs into the training dataset, the prediction performance of DeepACP could be further improved. In brief, we provide a new idea for identification of ACPs and expect that DeepACP will play an important role in the functional research of ACPs and the development of novel anticancer drugs.

MATERIALS AND METHODS

Datasets

Well-established datasets are fundamental to building stable and reliable predictors, and several benchmark datasets have been constructed in previous studies.[12–14,16] To fairly compare with several existing computational methods, the benchmark dataset established by Wei et al.[12] was chosen to construct the training dataset and the independent test dataset in this study, the size of which is relatively large and balanced. After data redundancy reduction by using CD-HIT[40] with the threshold of 90%, 332 ACPs and 1,023 non-ACPs were kept in the original dataset. For machine learning methods, better performance is more readily available on a balanced training dataset. The training dataset was constructed using the same numbers of positive and negative samples, which included 250 ACPs and 250 non-ACPs randomly selected from the original dataset. For further evaluating the prediction performance of DeepACP and comparing with other methods, an independent test dataset consisting of 82 ACPs and 82 non-ACPs was generated. Note that the training and independent test datasets used in this work are the same as those for ACPred-FL.[12]

Deep Neural Networks

In recent years, deep learning has had stunning success, surpassing human-level performance on hard problems such as images, language, and speech processing.[41] Almost every discipline of science and engineering has been affected, from medicine[42] and biology[43] to high-energy physics.[44] There are two commonly used families of architectures for deep learning: CNNs and RNNs. CNNs are among the most successful deep learning architectures, owing to their outstanding capacity to analyze spatial information. The powerful

Table 4. Performance Comparisons of Our Proposed DeepACP with the Existing Methods

Model	TP	TN	FN	FP	ACC (%)	F Value (%)	Recall (%)	PRE (%)	MCC
AntiCP_AAC[a]	56	71	26	11	77.4	75.2	68.3	83.6	0.558
AntiCP_DPC[b]	61	69	21	13	79.3	78.2	74.4	82.4	0.588
Hajisharifi et al.[14]	55	71	27	11	76.8	74.3	67.1	83.3	0.547
iACP	56	66	26	16	74.4	72.7	68.3	77.8	0.491
ACPred-FL	66	79	16	3	88.4	87.4	80.5	95.7	0.778
CNN-RNN	59	67	23	15	76.8	75.6	72.0	79.7	0.539
CNN	64	65	18	17	78.6	78.5	78.0	79.0	0.573
DeepACP (RNN)	64	72	18	10	82.9	82.0	78.0	86.5	0.662

TP, true positive; TN, true negative; FN, false negative; FP, false positive; ACC, accuracy; PRE, precision; MCC, Matthew's correlation coefficient; CNN-RNN, convolutional-recurrent neural network.

property of CNNs is that they can allow computers to process hierarchical spatial representations efficiently and holistically without relying on laborious feature crafting and extraction. Therefore, CNNs are particularly relevant in fields that produce or interpret images. For example, medical imaging adopted this trend early and applied deep CNNs to complex diagnostics spanning dermatology, radiology, ophthalmology, and pathology.[41,42] Currently, CNNs are increasingly used for biological sequence analysis including genomics and other high-throughput areas of research.[36,43] RNNs are designed to process sequential inputs such as language, speech, and time-series data. Since input data are processed sequentially, recurrent computation is performed on the hidden units where cyclic connection exists.[45] The hidden units of the RNNs can be viewed as memory states that retain information from the input data previously observed and are updated at each time step. The key advantage of RNNs over CNNs is that they are able to find long-range patterns in the data, which are highly dependent on the ordering of the input data for the prediction task. Therefore, RNNs are the first choice for researchers in many areas, including machine translation, text generation, and image captioning.[42]

CNNs

A convolutional neural network is a deep learning model with the key idea of using convolutional layers to extract features from input data. In a convolutional layer, neurons are able to extract higher level abstraction features from extracted features of the previous layer. The convolution operation in CNN was inspired by the visual mechanisms of living organisms. More specifically, it could be denoted as

$$\boldsymbol{Convolution}(X)_{ik} = \boldsymbol{ReLU}\left(\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \boldsymbol{w}_{mn}^{k}\boldsymbol{x}_{i+m,n}\right), \quad \text{(Equation 1)}$$

where X represents the input matrix, i represents the index of the output position, and k represents the index of the filter. The formula $\boldsymbol{W}^{k} = (\boldsymbol{w}_{mn}^{k})_{M\times N}$ represents the weight matrix of the kth convolution kernel with size $M \times N$, where M represents window size and N represents the number of input channels. ReLU represents a nonlinear activation function applied to the convolution outputs, which sets negative values to zeros as

$$\boldsymbol{ReLU}(\boldsymbol{x}) = \boldsymbol{max}(\boldsymbol{0}, \boldsymbol{x}). \quad \text{(Equation 2)}$$

After the convolution and nonlinearity, CNNs typically use pooling, which is a dimension reduction to provide translation invariance and to extract higher level features from a wider range of the input data. The pooling operation is denoted as

$$\boldsymbol{pooling}(X)_{ik} = \boldsymbol{max}(\boldsymbol{x}_{iM,k}, \boldsymbol{x}_{iM+1,k}, \cdots, \boldsymbol{x}_{iM+M-1,k}) \quad \text{(Equation 3)}$$

where X represents the output of the convolution layer, M represents the pooling window size, i represents the index for output position, and k represents the index of the filter being pooled. In the fully connected operation, CNNs integrate high-level features of pooling outputs and transform the features into a fixed dimension space. The last operation of CNNs adopts a fully connected node to obtain a single output using a sigmoid activation function, which is defined as

$$\boldsymbol{Sigmoid}(\boldsymbol{x}) = \frac{\boldsymbol{1}}{\boldsymbol{1}+\boldsymbol{e}^{-\boldsymbol{x}}}. \quad \text{(Equation 4)}$$

RNNs

Designed to handle sequential data, RNNs have become the main neural model for tasks such as speech recognition and text generation. Although traditional RNNs have achieved significant results in natural language understanding, the "vanishing gradients" problem has made it difficult to train on long sequences. The long short-term memory (LSTM) network is a special type of RNN that can handle long-term dependencies by using gating functions. These gates can control when information is written, read from, and forgotten. Specifically, the classical structure of a LSTM cell is given in the following equations:[46]

$$\boldsymbol{i}_t = \boldsymbol{\sigma}(\boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{ci}\boldsymbol{c}_{t-1} + \boldsymbol{b}_i), \quad \text{(Equation 5)}$$

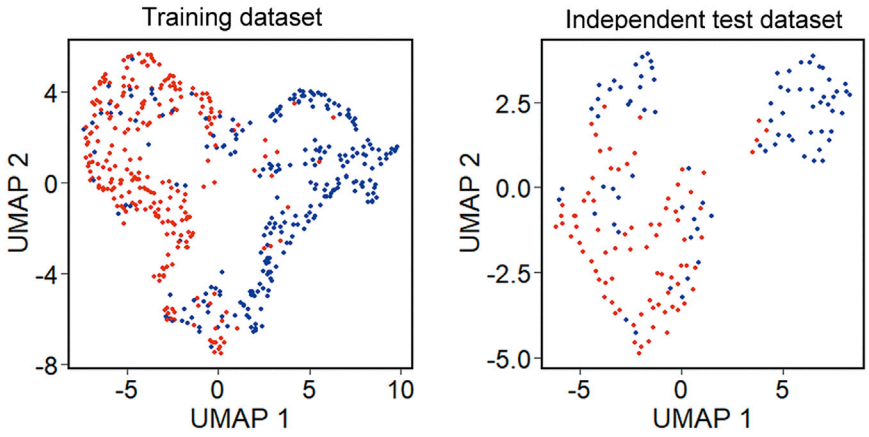

Figure 3. UMAP Visualization of the Hidden Layer Representations in the RNN Model

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right), \quad \text{(Equation 6)}$$

$$C_t = f_t c_{t-1} + i_t tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right), \quad \text{(Equation 7)}$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right), \quad \text{(Equation 8)}$$

and

$$h_t = O_t tanh\left(c_t\right), \quad \text{(Equation 9)}$$

where σ represents the logistic sigmoid function; *tanh* represents a hyperbolic tangent function that maps the real numbers to [−1, 1]; and *i*, *f*, *o*, and *c* represent, respectively, the input gate, forget gate, output gate, and cell and cell input activation vectors, which are specified to be the same value as given in the hidden vector *h*. W_{xf} represents the input-forget gate matrix, and W_{hf} represents the hidden-forget matrix. In the bidirectional LSTM network, the input sequence gets fed through two LSTM networks in both the forward and backward directions, which each produce a matrix of column vectors representing the LSTM network output. The output of the biLSTM is then computed by concatenating the output vectors of the two directions together.

Deep Learning Architectures for ACP Prediction

We trained several deep-neural-network-based models to computationally predict anticancer activity from peptide sequence. We designed three architectures: namely, CNNs, RNNs with biLSTMs, and a hybrid neural network combining a CNN and a RNN that uses peptide primary sequences as input and outputs a probability score between 0 and 1. More precisely, the input to the models is a sequence of one-letter-encoded amino acids, where each of the 20 basic amino acids are assigned a number from 1 to 20, and unknown "X" characters are assigned 0, respectively.[47] The output of the models consists of one score that maps to the [0, 1] interval; this interval corresponds to the probability of the peptide of interest being an ACP or a non-ACP. Here, we describe the overview of prediction architectures.

The RNN comprised three layers: the embedding layer, the biLSTM layer, and the output layer.[48] The embedding layer transformed the input numbers (0, 1, …, 20) into a 128-dimensional vector representation. The amino-acid character embedding can best be thought of as a one-dimensional signal (over sequence position) spanning 21 signal channels (20 common amino acids + X). In this way, each peptide sequence was represented by a (128, *L*) two-dimensional vector, where *L* is the length of the peptide. Here, the sequence length was set to 100, where a number was chosen to fit the longest ACP and non-ACP in the training set. The biLSTM layer is the basic unit of the RNN architecture, which consisted of forward and backward LSTM network layers, and each layer consisted of a basic LSTM unit (a memory cell) with a 64-dimensional hidden state vector with a dropout ratio of 0.2. The forward layer runs on the input sequences, while the backward layer runs on the reverse of input sequences. Finally, the outputs from all biLSTMs were then processed through a single, fully connected output layer with a sigmoid activation function. This yielded a single value for each peptide sequence, which represented a classification score.

The CNN was constructed from one embedding layer, one convolutional layer using the rectified linear unit (ReLU) activation function and interlaced with an AveragePooling layer.[49] The convolutional layer had 150 one-dimensional filters covering all amino-acid input channels. The filters of the convolutional layer were 5 positions wide. After convolution, the ReLU function was used to output the filter scanning results, which were above the thresholds and learned during model training. The AveragePooling layer subsampled the one-dimensional signal by a factor of 2 by averaging the filter scanning results at each position of the sequence. The flattened results from pooling were passed to a fully connected layer of 650 hidden ReLUs with 0.5 dropout, which finally connected to a dense layer with sigmoid activation and 1 output unit.

The third architecture was a hybrid CNN-RNN. Similar to the CNN architecture mentioned earlier, the convolutional stage performed one-dimensional convolution operations with 150 filters, together with a ReLU, to propagate positive outputs and eliminate negative outputs. Then, an AveragePooling layer was used to reduce dimensions and help extract higher level features by computing the average in each of the nonoverlapping windows of size 2. In this hybrid network, the features learned by the convolutional stage were followed by an RNN stage (biLSTM layer, 64 neurons) to further learn the context features from the pooled sequence patterns. The final layer was a fully connected node with a sigmoid activation function that predicted the probability of an ACP.

Model Training and Testing

Of the benchmark dataset, 70% of the data was used for training, 10% was used for validation, and 20% was used for testing. The training set was reshuffled after each training procedure. All architectures were trained for 20 epochs with a batch size of 50. Binary cross-entropy loss between the target and predicted outputs was minimized using the Adam optimizer during training. For each architecture, we repeated the training procedure 10 times, and the average of their outputs was used as the predicted output. All the deep learning architectures were trained on Graphics Processing Units (GPUs) that significantly accelerated the training process.

Predictive Performance Metric Calculation

TPs, FPs, TNs, and FNs annotate true positives, false positives, true negatives, and false negatives, respectively. Recall (also known as sensitivity) was calculated as:

$$\mathrm{Re}\,call = \frac{TP}{TP + FN}. \quad \text{(Equation 10)}$$

PRE (also known as positive predictive value) was calculated as:

$$PRE = \frac{TP}{TP + FP}. \quad \text{(Equation 11)}$$

ACC was calculated as:

$$ACC = \frac{TP + TN}{TP + FP + TN + FN}. \quad \text{(Equation 12)}$$

F value was calculated as:

$$F - \mathrm{value} = 2 \times \frac{TP}{2TP + FP + FN}. \quad \text{(Equation 13)}$$

MCC was calculated as:

$$MCC = \frac{(TP \times TN) - (FN \times FP)}{\sqrt{(TP + FN) \times (TN + FP) \times (TP + FP) \times (TN + FN)}}. \quad \text{(Equation 14)}$$

Model Visualization

To generate the model visualization, the biLSTM layer weights were first extracted from the trained RNN model. Then, the neurons representing all peptides were calculated from these weights. The UMAP[39] was applied (n_neighbors: 2, mim_dist: 0.5) to these neurons through the R uwot package[50] to reduce the original 128-dimensional vectors down to 2 dimensions. The plotting was conducted by using the R ggplot2 package.[51]

Implementation

We designed, trained, and evaluated the aforementioned deep learning models using the autoBioSeqpy tool with the Keras backend (https://keras.io). The autoBioSeqpy is an easy-to-use deep learning tool for biological sequence classification. The main advantage of this tool is its capability for easily developing and evaluating various deep learning models. Only the input datasets should be prepared by the users, as after that, the model development, training, and evaluation workflows can be run with a simple one-line command. After training, autoBioSeqpy automatically evaluates the model on the test set and generates figures to visualize the model's performance as an ACC and loss (ACC-loss) curve, ROC curve, and PR curve. In addition, this tool provides various ready-to-run applications for users, which makes the design of deep learning architectures easier. Code for model training and prediction with trained weights along with links to all data and model templates is available in a public repository at https://github.com/jingry/autoBioSeqpy/tree/master/examples/anticancer_peptide_prediction.

SUPPLEMENTAL INFORMATION

Supplemental Information can be found online at https://doi.org/10.1016/j.omtn.2020.10.005.

AUTHOR CONTRIBUTIONS

Conceptualization, J.L. and L.Y.; Methodology, R.J., J.L., and L.Y.; Investigation, L.Y., R.J., F.L., J.L., and Y.L.; Writing – Original Draft, J.L. and L.Y.; Writing – Review & Editing, J.L. and L.Y.; Funding Acquisition, L.Y.; Software, R.J. and J.L.; Supervision, J.L.

CONFLICTS OF INTEREST

The authors declare that they have no competing interests.

ACKNOWLEDGMENTS

This work was supported by grants from The Fund of Science and Technology Department of Guizhou Province ([2017]5790-07) and The Development Program for Youth Science and Technology Talents in Education Department of Guizhou Province (KY [2016]219).

REFERENCES

1. Santos, N.P., Colaço, A.A., and Oliveira, P.A. (2017). Animal models as a tool in hepatocellular carcinoma research: A Review. Tumour Biol. *39*, 1010428317695923.
2. Bray, F., Ferlay, J., Soerjomataram, I., Siegel, R.L., Torre, L.A., and Jemal, A. (2018). Global cancer statistics 2018: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries. CA Cancer J. Clin. *68*, 394–424.
3. Feng, P., and Wang, Z. (2019). Recent Advances in Computational Methods for Identifying Anticancer Peptides. Curr. Drug Targets *20*, 481–487.
4. Wu, Q., Ke, H., Li, D., Wang, Q., Fang, J., and Zhou, J. (2019). Recent Progress in Machine Learning-based Prediction of Peptide Activity for Drug Discovery. Curr. Top. Med. Chem. *19*, 4–16.
5. Song, X., Zhuang, Y., Lan, Y., Lin, Y., and Min, X. (2020). Comprehensive Review and Comparison for Anticancer Peptides Identification Models. Curr. Protein Pept. Sci. Published online January 17, 2020. https://doi.org/10.2174/1389203721666200117162958.
6. Holohan, C., Van Schaeybroeck, S., Longley, D.B., and Johnston, P.G. (2013). Cancer drug resistance: an evolving paradigm. Nat. Rev. Cancer *13*, 714–726.
7. Fosgerau, K., and Hoffmann, T. (2015). Peptide therapeutics: current status and future directions. Drug Discov. Today *20*, 122–128.
8. Lau, J.L., and Dunn, M.K. (2018). Therapeutic peptides: Historical perspectives, current development trends, and future directions. Bioorg. Med. Chem. *26*, 2700–2707.

9. Harris, F., Dennison, S.R., Singh, J., and Phoenix, D.A. (2013). On the selectivity and efficacy of defense peptides with respect to cancer cells. Med. Res. Rev. *33*, 190–234.
10. Basith Mail, S., Manavalan, B., Shin, T.H., Lee, D., and Lee, G. (2020). Evolution of Machine Learning Algorithms in the Prediction and Design of Anticancer Peptides. Curr. Protein Pept. Sci. Published online January 16, 2020. https://doi.org/10.2174/1389203721666200117171403.
11. Hu, Y., Lu, Y., Wang, S., Zhang, M., Qu, X., and Niu, B. (2019). Application of Machine Learning Approaches for the Design and Study of Anticancer Drugs. Curr. Drug Targets *20*, 488–500.
12. Wei, L., Zhou, C., Chen, H., Song, J., and Su, R. (2018). ACPred-FL: a sequence-based predictor using effective feature representation to improve the prediction of anti-cancer peptides. Bioinformatics *34*, 4007–4016.
13. Tyagi, A., Kapoor, P., Kumar, R., Chaudhary, K., Gautam, A., and Raghava, G.P. (2013). In silico models for designing and discovering novel anticancer peptides. Sci. Rep. *3*, 2984.
14. Hajisharifi, Z., Piryaiee, M., Mohammad Beigi, M., Behbahani, M., and Mohabatkar, H. (2014). Predicting anticancer peptides with Chou's pseudo amino acid composition and investigating their mutagenicity via Ames test. J. Theor. Biol. *341*, 34–40.
15. Vijayakumar, S., and Lakshmi, P. (2015). ACPP: A web server for prediction and design of anti-cancer peptides. Int. J. Pept. Res. Ther. *21*, 99–106.
16. Chen, W., Ding, H., Feng, P., Lin, H., and Chou, K.C. (2016). iACP: a sequence-based tool for identifying anticancer peptides. Oncotarget *7*, 16895–16909.
17. Li, F.M., and Wang, X.Q. (2016). Identifying anticancer peptides by using improved hybrid compositions. Sci. Rep. *6*, 33910.
18. Xu, L., Liang, G., Wang, L., and Liao, C. (2018). A Novel Hybrid Sequence-Based Model for Identifying Anticancer Peptides. Genes (Basel) *9*, 158.
19. Boopathi, V., Subramaniyam, S., Malik, A., Lee, G., Manavalan, B., and Yang, D.C. (2019). mACPpred: A Support Vector Machine-Based Meta-Predictor for Identification of Anticancer Peptides. Int. J. Mol. Sci. *20*, 1964.
20. Wei, L., Zhou, C., Su, R., and Zou, Q. (2019). PEPred-Suite: improved and robust prediction of therapeutic peptides using adaptive feature representation learning. Bioinformatics *35*, 4272–4280.
21. Rao, B., Zhou, C., Zhang, G., Su, R., and Wei, L. (2020). ACPred-Fuse: fusing multi-view information improves the prediction of anticancer peptides. Brief. Bioinform. *21*, 1846–1855.
22. Akbar, S., Hayat, M., Iqbal, M., and Jan, M.A. (2017). iACP-GAEnsC: Evolutionary genetic algorithm based ensemble classification of anticancer peptides by utilizing hybrid feature space. Artif. Intell. Med. *79*, 62–70.
23. Manavalan, B., Basith, S., Shin, T.H., Choi, S., Kim, M.O., and Lee, G. (2017). MLACP: machine-learning-based prediction of anticancer peptides. Oncotarget *8*, 77121–77136.
24. Kabir, M., Arif, M., Ahmad, S., Ali, Z., Swati, Z.N.K., and Yu, D.J. (2018). Intelligent computational method for discrimination of anticancer peptides by incorporating sequential and evolutionary profiles information. Chemometr. Intell. Lab. Syst. *182*, 158–165.
25. Schaduangrat, N., Nantasenamat, C., Prachayasittikul, V., and Shoombuatong, W. (2019). ACPred: A Computational Tool for the Prediction and Analysis of Anticancer Peptides. Molecules *24*, 1973.
26. Akbar, S., Rahman, A.U., Hayat, M., and Sohail, M. (2020). cACP: Classifying anticancer peptides using discriminative intelligent model via Chou's 5-step rules and general pseudo components. Chemometr. Intell. Lab. Syst. *196*, 103912.
27. Wu, C., Gao, R., Zhang, Y., and De Marinis, Y. (2019). PTPD: predicting therapeutic peptides by deep learning and word2vec. BMC Bioinformatics *20*, 456.
28. Grisoni, F., Neuhaus, C.S., Gabernet, G., Müller, A.T., Hiss, J.A., and Schneider, G. (2018). Designing Anticancer Peptides by Constructive Machine Learning. ChemMedChem *13*, 1300–1302.
29. Yi, H.C., You, Z.H., Zhou, X., Cheng, L., Li, X., Jiang, T.H., and Chen, Z.H. (2019). ACP-DL: A Deep Learning Long Short-Term Memory Model to Predict Anticancer Peptides Using High-Efficiency Feature Representation. Mol. Ther. Nucleic Acids *17*, 1–9.
30. Wang, L., You, Z.H., Chen, X., Xia, S.X., Liu, F., Yan, X., Zhou, Y., and Song, K.J. (2018). A computational-based method for predicting drug-target interactions by using stacked autoencoder deep neural network. J. Comput. Biol. *25*, 361–373.
31. Wan, F., Zhu, Y., Hu, H., Dai, A., Cai, X., Chen, L., Gong, H., Xia, T., Yang, D., Wang, M.-W., et al. (2019). DeepCPI: A Deep Learning-based Framework for Large-scale in silico Drug Screening. Genomics Proteomics Bioinformatics *17*, 478–495.
32. Öztürk, H., Özgür, A., and Ozkirimli, E. (2018). DeepDTA: deep drug-target binding affinity prediction. Bioinformatics *34*, i821–i829.
33. Xu, Y., Dai, Z., Chen, F., Gao, S., Pei, J., and Lai, L. (2015). Deep learning for drug-induced liver injury. J. Chem. Inf. Model. *55*, 2085–2093.
34. Feng, C., Chen, H., Yuan, X., Sun, M., Chu, K., Liu, H., and Rui, M. (2019). Gene Expression Data Based Deep Learning Model for Accurate Prediction of Drug-Induced Liver Injury in Advance. J. Chem. Inf. Model. *59*, 3240–3250.
35. Chierici, M., Francescatto, M., Bussola, N., Jurman, G., and Furlanello, C. (2020). Predictability of drug-induced liver injury by machine learning. Biol. Direct *15*, 3.
36. Zou, J., Huss, M., Abid, A., Mohammadi, P., Torkamani, A., and Telenti, A. (2019). A primer on deep learning in genomics. Nat. Genet. *51*, 12–18.
37. Zhang, Y.Q., Qiao, S.J., Ji, S.J., and Li, Y.Z. (2020). DeepSite: bidirectional LSTM and CNN models for predicting DNA–protein binding. Int. J. Mach. Learn. Cyb. *11*, 841–851.
38. McInnes, L., and Healy, J. (2018). UMAP: uniform manifold approximation and projection for dimension reduction. arXiv, arXiv:1802.03426, https://arxiv.org/abs/1802.03426.
39. Singh, M., Kumar, V., Sikka, K., Thakur, R., Harioudh, M.K., Mishra, D.P., Ghosh, J.K., and Siddiqi, M.I. (2020). Computational Design of Biologically Active Anticancer Peptides and Their Interactions with Heterogeneous POPC/POPS Lipid Membranes. J. Chem. Inf. Model. *60*, 332–341.
40. Huang, Y., Niu, B., Gao, Y., Fu, L., and Li, W. (2010). CD-HIT Suite: a web server for clustering and comparing biological sequences. Bioinformatics *26*, 680–682.
41. Belthangady, C., and Royer, L.A. (2019). Applications, promises, and pitfalls of deep learning for fluorescence image reconstruction. Nat. Methods *16*, 1215–1225.
42. Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., Cui, C., Corrado, G., Thrun, S., and Dean, J. (2019). A guide to deep learning in healthcare. Nat. Med. *25*, 24–29.
43. Eraslan, G., Avsec, Ž., Gagneur, J., and Theis, F.J. (2019). Deep learning: new computational modelling techniques for genomics. Nat. Rev. Genet. *20*, 389–403.
44. Radovic, A., Williams, M., Rousseau, D., Kagan, M., Bonacorsi, D., Himmel, A., Aurisano, A., Terao, K., and Wongjirad, T. (2018). Machine learning at the energy and intensity frontiers of particle physics. Nature *560*, 41–48.
45. Min, S., Lee, B., and Yoon, S. (2017). Deep learning in bioinformatics. Brief. Bioinform. *18*, 851–869.
46. Graves, A. (2013). Generating sequences with recurrent neural networks. arXiv, arXiv:1308.0850, https://arxiv.org/abs/1308.0850.
47. Veltri, D., Kamath, U., and Shehu, A. (2018). Deep learning improves antimicrobial peptide recognition. Bioinformatics *34*, 2740–2747.
48. Hannigan, G.D., Prihoda, D., Palicka, A., Soukup, J., Klempir, O., Rampula, L., Durcak, J., Wurst, M., Kotowski, J., Chang, D., et al. (2019). A deep learning genome-mining strategy for biosynthetic gene cluster prediction. Nucleic Acids Res. *47*, e110.
49. Bogard, N., Linder, J., Rosenberg, A.B., and Seelig, G. (2019). A Deep Neural Network for Predicting and Engineering Alternative Polyadenylation. Cell *178*, 91–106.e23.
50. Melville, J. (2019). uwot: The uniform manifold approximation and projection (UMAP) method for dimensionality reduction. https://github.com/jlmelville/uwot.
51. Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis (Springer).