5.1 Data sources

So far, there is no public website to integrate off-target data, and most studies still use various detection methods to obtain the data of potential off-target sites and off-target propensity of specific sgRNA at specific site. We used the off-target data that has been published in the DeepCrispr article as our training data. The off-target data set contains a total of 29 sgRNA from two different cell types: 293-related cell lines and K562t. For all 29 sgRNAs, a total of more than 650 positive data have been identified as off-target sites, and Guohui C *et al*.[37] obtained more than 160,000 possible loci across the whole genome similar to the corresponding sgRNA using *bowtie2*. The whole dataset was highly unbalanced, it was likely to affect model fitting precision in the process of training, we will further describe the concrete solution in the "Sampling for Training data" section. For the classification model, the labels of off-target sites were set to "1", and the labels of other sites were set to "0".For the regression model, the labels of off-target site were set to the targeting cleavage frequency detected by different detection assays and other sites were set to “0”.

5.1.1 Data Preprocessing and Encoding

The sgRNA sequence and its corresponding DNA sequence are each composed of 23 bases. Considering that the combination of the two bases at the same site is a feature, since the DNA sequence is composed of four types of bases: A, C, G and T, there are altogether 4×4=16 possible situations for this feature. Therefore, we set a unique index value for each combination, and encoded the original sgRNA-DNA sequence with an index vector of 23 for subsequent GloVe model embedding (showed in **Fig. 1a**).

5.2 GloVe model for Data Embedding

As an unsupervised word representation method, the GloVe[48] model enables vectors to contain as many hidden features of input data as possible through vectorization of the original data, which can eliminate the disadvantages of the one-hot coding method. In this model, the input vector was obtained by multiplying the one-hot coding of the original "vocabulary" by a trained weight matrix. When training the GloVe model, the co-occurrence matrix X should firstly be calculated according to the original "corpus" (here it refers to the original data set composed of our original sgRNA and corresponding off-target sequences). An element $x_{i,j}$ in matrix X is the sum of the times that the word w_j appears in the context box of the word w_i.

$X_i = \sum_{j=1}^{N} x_{i,j}$, represents the sum of the times that appear in the context box of word w_i for all words in the word list.

$P_{i,k} = \frac{x_{i,k}}{X_i}$, represents the probability that word w_k appears in the context box of word w_i.

$ratio_{i,j,k} = \frac{P_{i,k}}{P_{j,k}}$, represents the correlation of words w_i, w_j, w_k (**Table 4**). We can notice that the value of $ratio_{i,j,k}$ is calculated according to the co-occurrence times in the co-occurrence matrix $X = (x_{i,j})$. Now we hope to construct a word vector for each word and reproduce the value by using the word vector v_i, v_j, v_k and a specific function $g(\bullet)$. If such a word vector v_i can be found, it indicates that the word vector will definitely contain the information in the co-occurrence matrix.

In order to make the value of $g(v_i, v_j, v_k)$ as close as possible to $ratio_{i,j,k} = \frac{P_{i,k}}{P_{j,k}}$, we considered to build a cost function:

$$J = \sum_{i,j,k}^{N} \left(\frac{P_{i,k}}{P_{j,k}} - g\left(v_i, v_j, v_k\right) \right)^2,$$

and obtained the final cost function expression:

$$J = \sum_{i,j}^{N} f\left(x_{i,j}\right)\left(v_i^{\mathrm{T}} v_j + b_i + b_j - \log\left(x_{i,j}\right)\right)^2$$

through a series of hypothesis derivation.

In order to implement the GloVe embedding model, we called the *Python* extension package *mittens*, and used the GloVe model to train the preprocessed co-occurrence matrix. Finally, the embedded word vector representation was obtained. Global Vector integrated the global statistics of Latent Semantic Analysis (LSA) with the advantages of local context window. By integrating the aforementioned statistical information in its entirety, the training speed of the model can be accelerated and the relative weight of words can be controlled.

5.3 Recurrent Neural Network and LSTM Variant

As a special neural network model, the recurrent neural network (RNN) adds the horizontal connection of each neuron node in the same layer on the basis of the multi-layer feedforward network.

$$O_t = g(VS_t)$$
$$S_t = f(Ux_t + WS_{t-1})$$

RNN is mainly used to process time series. The output layer performs a full connection operation with the adjacent hidden layer. V is the connection matrix of the output layer, and g is used as an activation function to obtain the final output result. For the hidden layer at time t, neuron node first receives the network input from that moment by weight U and receives the hidden layer output from time $t-1$ by weight W. And further operations on the sum of the two are taken. It can be seen that RNN has only one hidden layer, and the hidden layer is called multiple times during the training process to realize the information extraction function of the input information context. RNN implements the memory function but this memory function is limited because there will be a phenomenon of gradient disappearance or gradient explosion. Therefore, *Hochreiter* and *Schmidhuber*[49] proposed the LSTM network to solve the above mentioned problems by introducing a memory cell, the key to which is the cell state, which receives or rejects input information by a well-designed structure called a "gate". Although many LSTM variants have been proposed, we have only described the forward recursion of a most basic LSTM model. The basic formulas were given below:

Input gate: $f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right)$

Forgetting gate: $i_t = \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right)$

$$C_t = \tanh\left(W_C \cdot \left[h_{t-1}, x_t\right] + b_C\right)$$

Output gate: $o_t = \sigma\left(W_o \cdot \left[h_{t-1}, x_t\right] + b_o\right)$

For the model input at time t, the input gate controls the selection of information and adds it to next step. The value of the input gate is usually between $\left[0,1\right]$ as the degree of information selection. The purpose of forgetting gate is mainly to prevent the introduction of too much information to relieve the burden of memory C. Therefore, some useless information is discarded by setting forgetting gate. As a deformed structure of RNN, LSTM adds memory units to each neuron in the hidden layer. Through the setting of several controllable gates on the neuron, it can control the memory and forgetting degree of current information and previous information, thus achieving the function of long-term memory.

In order to get the characteristic representation of forward and backward information of RNA sequences, the bidirectional LSTM (showed in **Fig. 1c**), a variant of LSTM, was used, which consists of two parallel LSTM: one input forward sequence and one input reverse sequence. Here's how it works:

$$y_2 = g\left(VA_2 + V^{'}A_2^{'}\right)$$

And,

$$A_2 = f\left(WA_1 + Ux_2\right)$$
$$A_2^{'} = f\left(W^{'}A_3^{'} + U^{'}x_2\right)$$

For output y_2, its input mainly comes from the weighted sum of two parts, A_2 and $A_2^{'}$.

Therefore, two values should be stored in the hidden layer of LSTM, which are respectively A participating in forward calculation and $A^{'}$ participating in reverse calculation.

5.4 Convolution Neural Network and Batch Normalization

CNN (showed in **Fig. 1d**) mainly uses image data as network input, which avoids the complex process of feature extraction and data reconstruction that often occurs in traditional recognition algorithms. Therefore, it has great advantages in 2D image processing.

In the training of CnnCrispr, due to the small number of samples available for training and testing, the samples in the training set will be over-trained, which will lead to overfitting problem and the insufficient generalization ability of the model in the test and other related data sets. Therefore, we need to reduce the over-fitting phenomenon of the model as much as possible by adjusting the model structure. Batch Normalization is a good tool for solving this problem. With the deepening of network depth or in the training process, the distribution of the activated input value before the nonlinear transformation will gradually shift or change the deep neural network, and the gradient of the low layer neural network will disappear in the backward propagation process, hence the convergence speed of the deep layer neural network will decrease. Batch Normalization forcibly transforms the input value distribution of any neuron into the standard normal distribution with a mean value equaling to 0 and variance equaling to 1, ensuring that the input value of each layer of network is uniformly distributed in the training process of the model, so as to avoid the gradient disappearance problem and further accelerate the training speed. In addition, the Batch Normalization layer, as an alternative operation to Dropout, avoids the inactivation of some input nodes and thus reduces the loss of effective information.

5.5 Sampling for Training data

The samples in the entire data set is extremely unbalanced: the number of negative samples is about 250 times that of positive samples. A highly unbalanced data set may make the gradient update process unstable, increase the difficulty of training, and even lead to the failure of model training. In order to solve this problem, we designed a data sampling method for model training: we set the batch size as m for model training, and divided the negative sample set into N subsets according to this value (if the negative set has M samples, batch size is m, then N =[M / m]), and N random oversampling operations are further performed on the positive training set, therefore generating N training batches. In the data sampling process, almost all negative data are traversed, and the potential information of the negative data can be searched more reliably. However, one thing to note is that oversampling may over-emphasize the effects of some positive data on the training results, resulting in the model overfitting. So, the choice of batch size is very crucial.

5.6 Model Evaluation and Performance Measures

In this paper, we investigated the classification and regression models for prediction of off-target propensity. For the classification model, the confusion matrix M of the predicted results can be obtained.

$$M = \begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$$

Among them, TP (True Positive) and TN (True Negative) were expected prediction results, while FP (False Positive) and FN (False Negative) would reduce the prediction accuracy of our model.

For the prediction of off-target propensity, we should pay more attention to whether the actual off-target site is correctly predicted as "off-target" -- that is, whether the label value is 1. Therefore, ROC curve, PRC curve and Recall value were selected as the performance measures. The Recall value is defined as follows:

$$Recall = \frac{TP}{TP + FN}$$

The recall value can well describe the proportion of the actual off-target sites that are correctly classified, that is, the larger the recall value is, the better the prediction ability of the model will be.

ROC and PRC curves[50] are widely used in classification model, it is important to note that since the prediction problem of category has highly unbalanced characteristics, compared with the AUC value under the ROC curve, the area under the PRC and its curve is more worthy of attention, the higher the value, proves the model has better performance in class imbalance problems.

For the regression model, the Pearson and Spearman correlation coefficients are mainly considered as metrics. The two measures are continuous variable correlation evaluation indicators, between that two, the Pearson value pay more attention to whether it has a linear relationship between two variables, while Spearman value is a nonparametric statistical method to do the linear correlation analysis by using the rank of each variable.