Research on Bitcoin Price Prediction Model Based on Informer and CNN

Yu Shi
Sophomore, School of Management & Engineering, Nanjing University, Nanjing 210046
Junhui Zhang
Sophomore, School of Management & Engineering, Nanjing University, Nanjing 210046
Zile Zhan
Junior, School of Economics, Xiamen University, Xiamen 316005
Advisor：Gabriel Xia
Questrom School of Business, Boston University, Boston 02215

Abstract: Since the birth of BTC in 2009, digital assets have experienced a rapid development. Bitcoin, one of the cryptocurrencies with the best liquidity, has become a research hotspot for these years. As changes in the price of bitcoin have great impact on cryptocurrency trading market, how to forecast its price trends is considered a most important problem. Informer, an efficient long-term series forecasting model, was proposed in 2020 based on the Transformer model. In our paper, we devise a new time series forecasting model called Informers-CNN, which is based on Informer and Convolutional neural network. We firstly select 29 common indicators reflecting bitcoin price trends as our data set on the minute-level time unit. We then train various models with different parameter values, by adjusting the two important hyperparameters of the Informer model–the input sequence length and the forecast sequence length, thus obtaining a batch of prediction outputs. After processing these outputs, we stitch them to form a three-dimensional matrix as the input for our CNN part, which takes future time series value as the fitting object. Finally, we apply our model on the estimation of bitcoin price, and on multiple prediction sequence lengths we compare them with common forecasting models. The result shows our model has significantly better predictive effects in comparison.

Key words: Bitcoin, Price prediction, Convolutional neural network, Time series forecasting

Introduction

In recent years, the dramatic surge in Bitcoin price has aroused a lot of people's interest. Since breaking 20000 dollars in 2017, the price of Bitcoin has still kept rising drastically; in fact, it has reached approximately 59000 dollars a piece until April, 2021. The exponential growth rate has led to its huge gains that no other financial investment can attain in a short period of time. However, Bitcoin's two record-breaking falls in 2018 and 2021 have also dragged down the entire cryptocurrency market. Due to the huge volatility of the prices of cryptocurrencies, more and more people are becoming interested in studying the factors that affect Bitcoin price, thus forecasting how it will change in the future.

The prediction of the trend of Bitcoin price is a common concern for cryptocurrency traders. If a trader can know the trend of price in advance, they will be able to make long or short trades (margin trading, which is supported by many exchanges) to attain great profits, whether the price goes up or down. Holding long or short positions has become easier ever since CBOE introduced Bitcoin futures in December, 2017.

Then a question follows: can the Bitcoin price be predicted? Does the efficient market hypothesis(EMH) still hold in the Bitcoin market? As in an efficient market, all the information is reflected in the asset price[4], it is infeasible to predict how the price will change in the future. The efficiency of the Bitcoin market has been studied by many a scholar in the past few years: Urquhart(2016) gave the earliest evidence of the efficiency of the Bitcoin market, along with the conclusion that the Bitcoin market is not weakly efficient, yet its efficiency is gradually diminishing over time; Yang Xuan et al.(2019)[11] pointed out that the Bitcoin market conforms to the adaptive market hypothesis, and the market efficiency index is constantly changing; Nadarajah et al.(2017)[7] believed that the Bitcoin market is weakly efficient in most sample spaces , through multiple weak form valid tests.

Most of the studies above were based on common statistical tests, which rejected the null hypothesis of weak efficiency in the Bitcoin market through statistical tests. However, they neither provided a clear pricing model or a prediction method, nor demonstrated that excess returns may be obtained through applying certain continuous trading methods.

Researches on the prediction of Bitcoin price mainly focused on the empirical analysis of the performances of traditional time series prediction models in the Bitcoin market, and most of them only used the closing price of Bitcoin for prediction, rather than introduce other features that reflect Bitcoin price. For instance, Duan Gege(2021)[3] used the ARIMA model to predict the daily closing price of Bitcoin from 2013 to 2016; OthmanAHA et al.(2020)[9] applied the symmetrical volatility of Bitcoin price and combined the technique of artificial neural network(ANN) to predict the closing price of Bitcoin; Akyildirim et al.(2021)[2] studied the performance of a variety of common machine learning algorithms in predicting the price of cryptocurrency in the form of empirical research, and found that random forests and support vector machines have a weak advantage in price prediction. Additionally, Livieris(2020)[6] used an integrated learning method to predict the price of multiple cryptocurrencies and their ups and downs.

In this research, our goal is to establish a new Bitcoin price prediction model. Different from other studies that use daily closing prices as their forecasts, we choose price data at the minute level for forecasting. We do this in consideration that in today's cryptocurrency market, the average asset holding period is hardly more than just a few minutes when implementing the algorithms to trade, especially in high-frequency trading. Moreover, we use 29 indicators to measure the price data in a comprehensive way to improve the accuracy of the forecast results. In addition, we also attempt predict to predict the Bitcoin price of multiple time intervals in the future, rather than just the price of the next time interval, which helps us better assign trading strategies and grasp the long-term trend of Bitcoin price.

In order to build our new Bitcoin price prediction model and make it perform as well as possible, we devise a neural network model which is different from traditional ones. Recent studies show that the Transformer model[10] has the potential to improve prediction capabilities, but it has problems of secondary time complexity, high memory usage, and inherent limitations of the "encoder/decoder" structure, which makes it hard to be directly applied in time series prediction problems. Therefore, Haoyi Zhou et al.(2020)[12] devised a LSTF model on the basis of Transformer, that is, the Informer model. The Informer model achieves good results on the LSTF problems, so we employ it as the basis of our model. Convolutional neural network(CNN) is a variant of multi-layer perceptron(MLP). It reduces the number of weights by local connection and weight sharing, thus reducing the complexity of the network and making the optimization process easier.

Moreover, CNN supports the direct use of images as the input of the neural network, which avoids the process of feature extraction and data reconstruction in traditional recognition algorithms.Therefore, in order to better perform our prediction task, with the comprehensive use of the prediction results obtained from multiple training processes in determining the hyperparameters of the Informer model, we build a new time series prediction model—Informers-CNN, which is based on the Informer model and combines CNN. Then we make predictions of Bitcoin prices with our new model, and compare and analyse our estimation results with Informer and other common time series prediction models like ARMA, ARIMA and so on.

The rest of the paper is organized as follows: the first section introduces the data we use in the research and our preprocessing; the second section demonstrates our new model—Informer-CNN, which we use to predict Bitcoin price; the third section shows the prediction results of the model and the comparison between its prediction results and that of other time series forecasting models, and the fourth section is the summary.

1 Data and Preprocessing

1.1 Index Data

In our research, we collect Bitcoin transaction data in U.S. dollars from BitStamp Exchange. With different time intervals between two acquisitions—15min, 5min and 1min, we obtain three data sets. These data cover a time period from November 30th, 2020 to March 22th, 2021, and we use each price at the end of a time interval as a forecasting object. Below, we take the data set of a 15-minute time interval to clarify, as is shown in Figure 1.

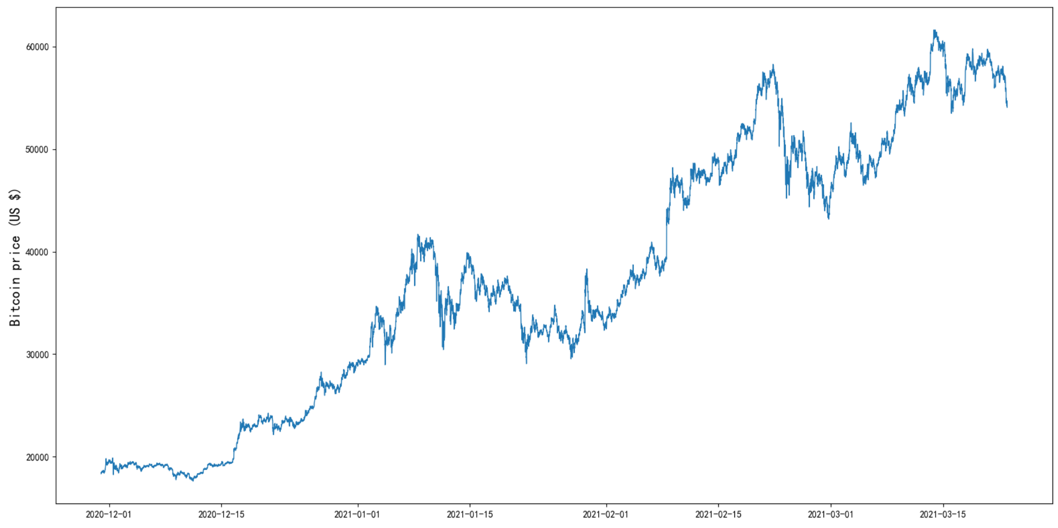


Figure 1: Bitcoin Price Curve

We use 29 indicators to predict the Bitcoin price in the next time bar. Basic indicators include 8 characteristics in total: the starting price(professionally, Open Price, which refers to the Bitcoin price at the beginning of a time period, while Closing Price stands for that at the end of a time period), the highest price, the lowest price, Closing Price, Bitcoin transaction volume (USD denominated), USD transaction volume, weighted price, difference between the highest price and the lowest price; along with 21 other indicators including five returns with different lag levels(from 1 to 5 lag lengths), weighted moving average, the correlation coefficient between weighted moving averages of different lag lengths, the sum of prices for different lengths of time, and so on. To select these indicators, we referred to the technical indexes selected by Kara(2011)[8], Guresen(2011)[5] and other researchers in conducting similar researches regarding predication of asset prices.

Explanation of some indicators are as follows:

Average true range(ATR) is calculated through the following formula:

$$ATR = \mathrm{MA}(TR, N) \tag{1}$$

where $MA(TR,N)$ refers to calculating the moving average of TR for the length of N time units, that is, the average price of the past N time periods, and TR_t in the Tth time period is computed as:

$$TR_t = \max(High_t - Low_t, High_t - Close_{t-1}, Close_{t-1} - Low_t) \tag{2}$$

where $High$, Low, $Close$ are the highest price, the lowest price and closing price respectively. Relative Strength Index(RSI) is calculated as:

$$RSI = 100 - \frac{100}{1 + \frac{\mathrm{MA}(U,N)}{\mathrm{MA}(D,N)}} \tag{3}$$

When the price goes up, we have:

$$U_t = Close_t - Close_{t-1}, \qquad D = 0 \tag{4}$$

and when it goes down, we have:

$$D_t = Close_{t-1} - Close_t, \qquad U = 0 \tag{5}$$

For price return,We have the following formula：

$$Return_{i_t} = Close_t - Close_{t-i} \tag{6}$$

For rate of change(ROC),We have：

$$ROC_{i_t} = 100 \cdot (\frac{Close_t}{Close_{t-i}} - 1) \tag{7}$$

In addition, the Stochastic oscillator can reflect the momentum effect in the Bitcoin price, and its calculation formula is as follows:

$$\begin{gathered} K = 100 \cdot \frac{Close - Low}{High - Low} \\ Stochastic\ oscillator = \mathrm{MA}(K, N) \end{gathered} \tag{8}$$

Finally, CCI(Commodity Channel Index), which is a reflection of the average price deviation, calculates the average degree of interval deviation in the condition that the price and the fixed interval are already known. It is calculated as:

$$\begin{aligned} CCI &= \frac{TP - MC}{MD \cdot 0.015} \\ TP &= \frac{High + Low + Close}{3} \\ MC &= \mathrm{MA}(Close, N) \\ MD &= \mathrm{MA}(MC - Close, N) \end{aligned} \tag{9}$$

All indicators used in our research are shown in Table1. Besides, explanations and calculation formulas of all technical indicators can also be found in Achelis (1995)[1].

Table 1: Features Utilized in Forecasting Models

	Feature name	Number of lags/window size	Number of features
1	Open, Close, High, Low	1	4
2	Weighted Price	1	1
3	Volume(BTC), Volume(Currency)	1	2
4	High - Low	1	1
5	Return	1,...,5	5
6	Correlation MA5 and MA30	1	1
7	Sum3, Sum5	5, 3	2
8	Sum5 - Sum3	1	1
9	RSI(6, 14)	6, 14	2
10	Rate of change	9, 14	2
11	Williams R	1	1
12	ATR	5, 10	2
13	CCI	1	1
14	DEMA	1	1

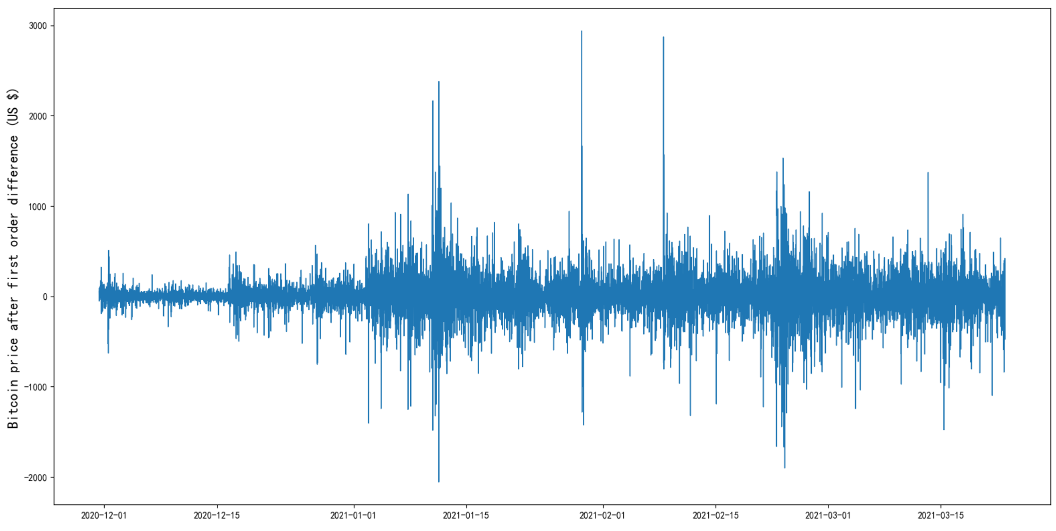
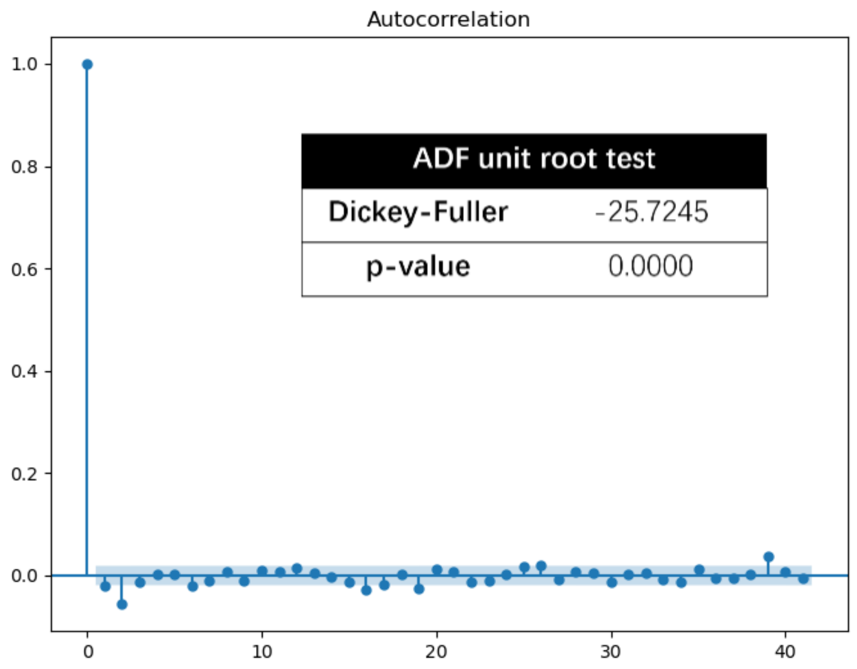


Figure 2: Bitcoin Price Curve After Difference



ADF unit root test	
Dickey-Fuller	-25.7245
p-value	0.0000

Figure 3: Autocorrelation Graph and ADF Test

1.2 Data Preprocessing

From Figure 1, we can see the Bitcoin price in this period is roughly on a rising trend and is an unstable sequence, so we consider performing a differential processing on it. The price curve after the process is shown in Figure2.

We draw an autocorrelation graph and perform the ADF unit root test, getting a $p-value$ of 0.0000. The null hypothesis of a unit root being rejected, the test is passed and the bitcoin price sequence after the differential processing is a stable one. Then we do a basic descriptive statistics on all indicators used in this study, and perform unit root tests on the original data and the first-order difference data, results of which are shown in Table 2 and Table 3.

Considering the large volatility of Bitcoin price, we take the natural logarithm of the closing price ($Close$) of each time period, and then make a first-order difference. We have:

$$new\ close_i = \log(close_i) - \log(close_{i-1}) = \log(\frac{close_i}{close_{i-1}}) \tag{10}$$

Moreover, in order to improve the performance of our neural network model, simultaneously speeding up the convergence speed and shortening the training time, We centralize all 29 indicators,We have:

$$new\ value_i = \frac{value_i - \mathrm{E}(value)}{\sqrt{\mathrm{D}(value)}} \tag{11}$$

Table 2: Descriptive Statistics of Features

Feature name	Mean	Median	Range	Variance	STD	Variation coefficient[1]
Open	37664.2	36269.08	43985.51	164022981	12807.15	2.94
High	37813.22	36443.09	44109.97	165233545.9	12854.32	2.94
Low	37509.37	36102.4	43780.72	162806276.6	12759.56	2.94
Close	37667.37	36279.81	43999.35	164027593.7	12807.33	2.94
Volume(BTC)	99.8	62.66	2177.71	15116.51	122.95	0.81
Volume(Currency)	3609200	2263022	78028857	2.04706E+13	4524449	0.8
Weighted Price	37663.33	36273.46	43987.47	164020583.2	12807.05	2.94
High Minus Low	303.86	251.1	4207.22	59960.95	244.87	1.24
Return1	3.29	1.87	4993.67	54035.74	232.46	0.01
Return2	3.32	1.87	4993.67	54017.55	232.42	0.01
Return3	3.32	1.87	4993.67	54017.4	232.42	0.01
Return4	3.33	1.87	4993.67	54013.59	232.41	0.01
Return5	3.32	1.87	4993.67	54012.96	232.41	0.01
MA5	37660.75	36279.35	43748.72	164002037.6	12806.33	2.94
Correlation MA5 and close 30	0.81	0.83	0.92	0.02	0.13	6.36
Sum3	112992.2	108808.3	131576.9	1476110881	38420.19	2.94
Sum5	188303.8	181396.7	218743.6	4100050940	64031.64	2.94
Sum5 Minus Sum3	75311.56	72528.68	87961.26	656163443.7	25615.69	2.94
RSI6	52.17	52.36	100	559.8	23.66	2.21
RSI14	52.3	51.73	92.83	261.95	16.18	3.23
MACD12_26	24.04	22.26	2455.14	53674.25	231.68	0.1
Rate of change 9	0	0	0.27	0	0.02	0.06
Rate of change 14	0	0	0.29	0	0.02	0.08
Williams R	0.44	0.41	1	0.08	0.27	1.6
ATR5	305.07	276.67	2274.9	41089.48	202.71	1.51
ATR10	304.98	284.35	1610.56	35766.64	189.12	1.61
Stochastic oscillator	0.55	0.58	1	0.13	0.36	1.51
CCI	28.42	43.95	328159.6	12401458.42	3521.57	0.01
DEMA	37667.52	36295.54	43827.36	164059577.3	12808.57	2.94

[1] where variation coefficient is calculated by $\frac{STD}{Mean}$, STD is standard deviation and $Mean$ is average value.

Table 3: Unit Root Test of Features

Feature name	Dickey-Fuller	p-value	Dickey-Fuller after 1st diff	p-value 1st diff
Open	-0.9	0.79	-21.02	0
High	-0.91	0.78	-15.93	0
Low	-0.99	0.76	-27.35	0
Close	-0.94	0.77	-25.72	0
Volume(BTC)	-8.79	0	-25.89	0
Volume(Currency)	-9.04	0	-25.61	0
Weighted Price	-0.96	0.77	-27.06	0
High Minus Low	-5.81	0	-26.32	0
Return1	-25.72	0	-29.8	0
Return2	-25.77	0	-29.81	0
Return3	-25.76	0	-29.81	0
Return4	-25.77	0	-29.81	0
Return5	-25.76	0	-29.8	0
MA5	-0.94	0.77	-16.37	0
Correlation MA5 and close 30	-14.83	0	-26.71	0
Sum3	-0.94	0.78	-15.27	0
Sum5	-0.94	0.77	-16.37	0
Sum5 Minus Sum3	-0.93	0.78	-16.28	0
RSI6	-19.69	0	-28.47	0
RSI14	-14.05	0	-28.78	0
MACD12_26	-13.46	0	-21.32	0
Rate of change 9	-16.2	0	-30.28	0
Rate of change 14	-12.6	0	-29.66	0
Williams R	-24.1	0	-28.05	0
ATR5	-5.79	0	-24.7	0
ATR10	-5.66	0	-24.79	0
Stochastic oscillator	-25.35	0	-28.51	0
CCI	-103.85	0	-28.27	0
DEMA	-0.98	0.76	-26.51	0

2 Forecasting Model

In our study, combining convolutional neural networks, we build a new time series prediction model based on the long-term series prediction model Informer. Then we make predictions of Bitcoin prices with our new model, and compare and analyse our estimation results with Informer and other common time series prediction models like ARMA, ARIMA and so on.

2.1 Informer

Recent studies show that the Transformer model[10] has the potential to improve prediction capabilities, but it has problems of secondary time complexity, high memory usage, and inherent limitations of the ”encoder/decoder” structure, which makes it hard to directly apply Transformer in time series prediction problems. Therefore, Haoyi Zhou et al. (2020)[12] devised the Informer model on the basis of Transformer and achieved good results on the long-term sequence prediction (LSTF) problems. The Informer model uses stacking self-attention computing layers as the main body, and on the whole still adopts the “encoder decoder” structure, as is shown in Figure 4.

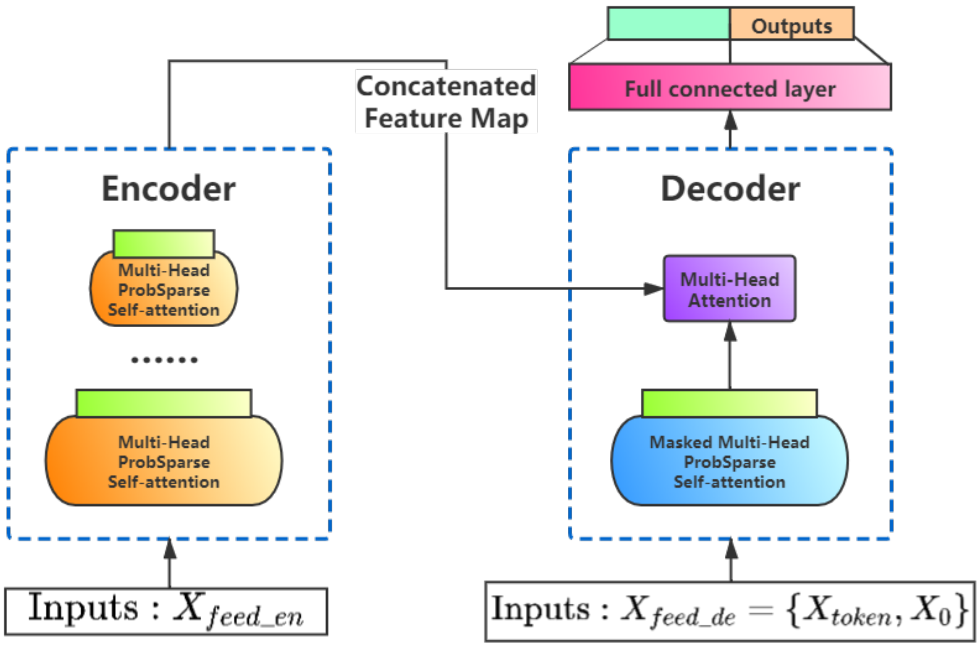

Outputs
Full connected layer
Concatenated Feature Map
Encoder
Multi-Head ProbSparse Self-attention
......
Multi-Head ProbSparse Self-attention
Decoder
Multi-Head Attention
Masked Multi-Head ProbSparse Self-attention
Inputs : X_{feed_en}
Inputs : $X_{feed\_de} = \{X_{token}, X_0\}$

Figure 4: Informer Model Structure

The whole prediction problem is defined as follows, and the input data at time *t* is:

$$x^t = \left\{x_1^t, \ldots, x_{L_x}^t \mid x_i^t \in \mathbb{R}^{d_x}\right\} \tag{12}$$

Correspondingly, the forecast output data has the following form:

$$y^t = \left\{y_1^t, \ldots, y_{L_y}^t \mid y_i^t \in \mathbb{R}^{d_y}\right\} \tag{13}$$

The input form of the traditional self-attention calculation mechanism is $(Query, Key, Value)$, and we perform a scaled dot product on the inputs, that is:

$$A(Q,K,V) = \text{Softmax}\left(\frac{QK^{\top}}{\sqrt{d}}\right)V \tag{14}$$

where $Q \in \mathbb{R}^{L_Q \times d}$, $K \in \mathbb{R}^{L_K \times d}$, $V \in \mathbb{R}^{L_V \times d}$, and the probabilistic form of the *ith Query*'s self-attention coefficient is:

$$A(q_i,K,V) = \sum_j \frac{k(q_i,k_j)}{\sum_l k(q_i,k_l)} v_j = \mathbb{E}_{p(k_j|q_i)}[v_j] \tag{15}$$

The time complexity of self-attention operation is $O(L^2)$, and research finds that the probability distribution of self-attention operation results satisfies the long-tailed distribution. By properly extracting a few of the dot product pairs that produce the main contribution, the calculation amount can be greatly reduced with a small loss of accuracy. Informer adopts the sparsity of *QueryKL* Divergence measure to design the ProbSparse self-attention operation, and the evaluation formula of the sparsity of the *ith Query* is:

$$M(q_i,K) = \ln \sum_{j=1}^{L_K} e^{\frac{q_i k_j^{\top}}{\sqrt{d}}} - \frac{1}{L_K}\sum_{j=1}^{L_K} \frac{q_i k_j^{\top}}{\sqrt{d}} \tag{16}$$

The first item of the formula is the LSE form of the dot product of q_i for all *Key*s, that is, taking the logarithm of the sum of the exponents; and the second item is their arithmetic mean value. Through this sampling method of ProbSparse, the time complexity of self-attention operation can be reduced to $O(LlogL)$.

In the encoder, for further extracting the main features of the data, we apply Self-attention Distilling to give higher weight to the dominant features, as well as creating an adjusted self-attention feature mapping in the next coding layer. The self-attention distillation process from layer j to layer $j+1$ is as follows:

$$X_{j+1}^{t} = \text{MaxPool}(\text{ELU}(\text{Conv1d}([\mathrm{X_j^t}]_{\mathrm{AB}}))) \tag{17}$$

where $[\cdot]_{AB}$ contains operations such as multi-head ProbSparse calculation.MaxPool is the maximum pooling algorithm. After performing one-dimensional convolution of $Conv1d$ on time elements, we employ ELU as the activation function.

In the decoding layer, Informer uses a standard decoder structure, which consists of two identical multi-head self-attention layers. The input vector is in the form:

$$X_{feed\_de}^{t} = \text{Concat}\left(X_{token}^{t}, X_0^{t}\right) \in \mathbb{R}^{(L_{token}+L_y)\times d_{model}} \tag{18}$$

In the decoding layer, through the application of Masked Multihead Attention, Informer avoids the occurrence of autoregressive situations. Finally, the prediction result is output through a fully connected layer. Informer takes mean square error (MSE) as the loss function.

2.2 Convolutional Neural Network(CNN)

Convolutional neural network, a variant of multi-layer perceptron (MLP), is a deep neural network with a convolutional structure. It reduces the number of weights by local connection and weight sharing, thus reducing the complexity of the network and making the optimization process easier.

This advantage is more obvious when images are used as network input. CNN supports direct use of images as input of the neural network, avoiding the process of feature extraction and data reconstruction in traditional recognition algorithms. As a multi-layer supervised learning neural network, CNN mainly extracts data features through the convolutional and the pooling layers in the hidden layers. Finally it outputs the result through the fully connected layer.

In our study, a convolutional neural network is designed as a part of the complete prediction model, whose structure is shown in Figure 5.

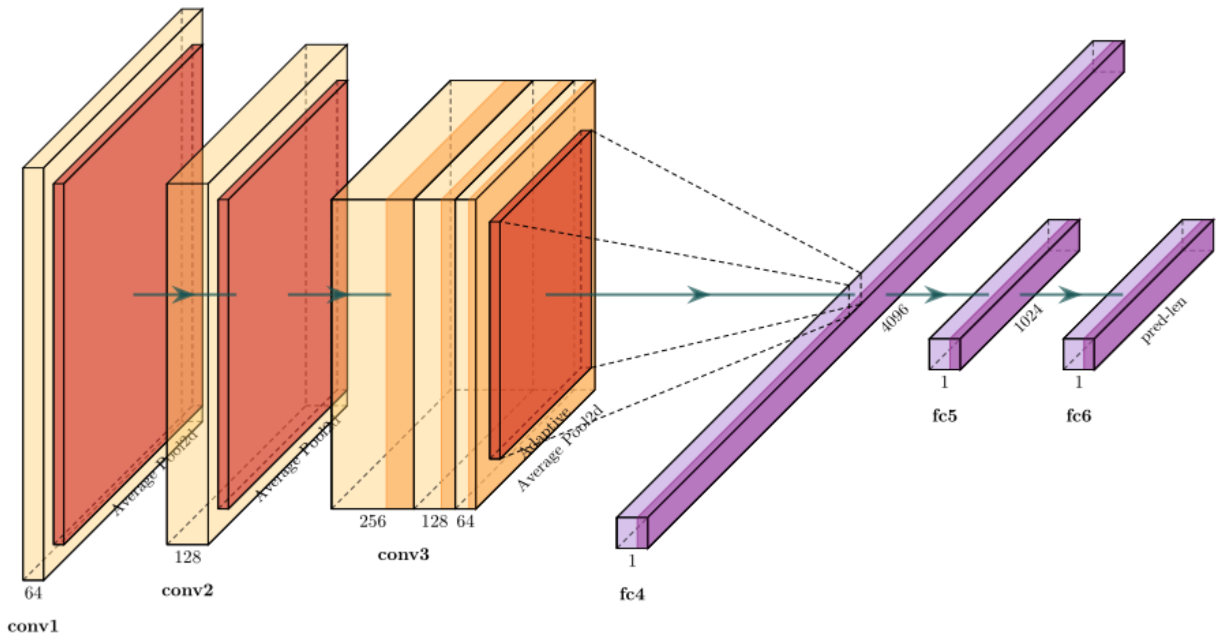


Figure 5: Convolutional Neural Network Structure

The input to this network is $X \in \mathbb{N}^*_{i\times o\times o}$.Considering that the output of the convolutional neural network in our study is required to fit a sequence satisfying the standard normal distribution, the *tanh* function is used as the activation function to enable the CNN to output a negative value, and the expression is:

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{19}$$

The *tanh* activation function takes $(0,0)$ as the symmetric center, and has a faster convergence rate than the *Sigmoid* activation function. For an input of dimension $\bar{i} \times \bar{j} \times \bar{r}$, and a filter span of $2k+1$, the kernel function of the entire convolutional neural network can be expressed as

$$\tilde{X} = K \circledast X = \sum_{r\in\mathbb{N}\cap[1,\bar{r}]} [\sum_{p,q\in\mathbb{Z}\cap[-k,k]} K_{k+1+p,k+1+q} x_{i+p+1,j+q+1}]_{i\in\mathbb{N}\cap[1,\bar{i}],j\in\mathbb{N}\cap[1,\bar{j}]} \tag{20}$$

We insert an adaptive average pool in front of the first fully connected layer of the network. As long as the input and output tensors are given, the adaptive average pooling can automatically calculate the size of the kernel and of the stride. Given the *kernel_size*, *padding*, *stride* and the *input_size*(size of the input tensor), the output tensor size can be determined by the following formula:

$$output\_size = \frac{input\_size + 2 * padding - kernel\_size}{stride + 1} \tag{21}$$

According to the formula above, we have:

$$\begin{aligned} kernel\_size &= (input\_size + 2 * paddig) - stride * (output\_size - 1) \\ stride &= \left\lfloor \frac{input\_size}{output\_size} \right\rfloor \\ kernel\_size &= input\_size - stride * (output\_size - 1) \\ padding &= 0 \end{aligned} \tag{22}$$

The whole network has four convolution layers and three fully connected layers, which ensures the network's ability to extract features of the input image and simultaneously avoids excessive parameters and a hard convergence caused by too many layers.

2.3 Informers-CNN

In practical application of Informer, we find the two hyperparameters —$input\_len$, the length of the input sequence length and $output\_len$, the length of the output (predicted) sequence, have a great impact on the prediction accuracy of the model. By adjusting the values of the two hyperparameters, we predict the Bitcoin price with this data set, in which $input\_len$ takes three values of 20, 40 and 60, and $output\_len$ takes 10 values ranging from 1 to 10. Thus we obtain 30 different Informer models through training. The average error of these models' outputs and the accuracy of their predictions for ups and downs (whether prices will rise or fall in the future) are shown in Figure 6.

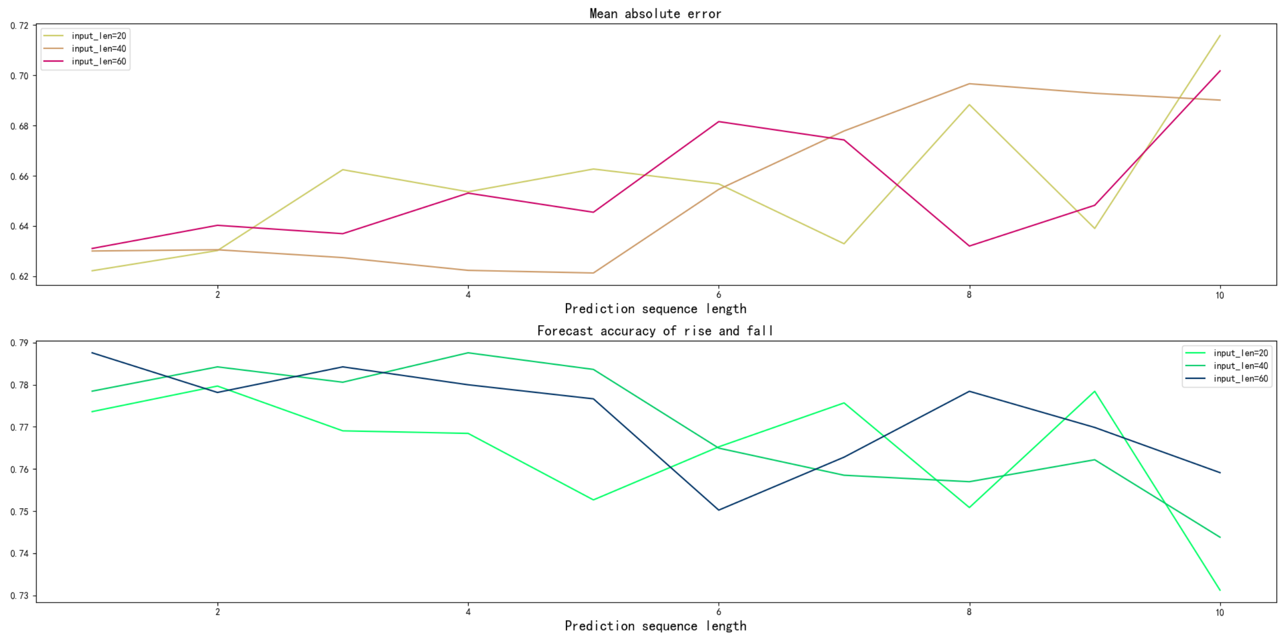


Figure 6: The Average Error and Accuracy of the Prediction Results of the Informer Model with Different Values of $input\_len$ and $output\_len$

From the figure, we can see with the lengthening of the forecast sequence, the average absolute error roughly increases, and the accuracy of the prediction for ups and downs gradually falls. However, it is hard to determine a method of quickly finding the model with the best performance in average error and prediction accuracy, in any prediction task using Informer. According to real experience, it is often necessary to train multiple models with different parameter selection and compare the prediction results so as to select the best performing one. The comparison results of the overestimated average error and the underestimated average error of the prediction results are shown in Figure 7.

It can be seen from Figure7 that there is no obvious law of overestimation and underestimation errors for the model, given different parameter values. Therefore it is not easy to choose the best parameters through these two errors, which, from another perspective, also reflects the difficulty in determining these two hyperparameters of $input\_len$ and $output\_len$.

Since several models of different parameter selection need to be trained to determine the hyperparameters of the Informer model, then, can we make comprehensive use of these prediction results to help us make better predictions? The answer is yes! In our study, we devise the Informers—CNN model. Based on multiple Informer models with different values of $input\_len$ and $output\_len$, we fully exploit the their results to further explore the characteristics of the trend of the time series, thus improving the prediction effect.

Suppose an Informer model of $input\_len = i$, $output\_len = o$, whose prediction vector is $Y_i^o = \left\{y_1,...,y_o \mid y_j \in \mathbb{R}^{d_{y_i^o}}\right\}$, shortened for $Y_i^o = [y_1,...,y_o]$. Let' s consider the set of values for $input\_len$

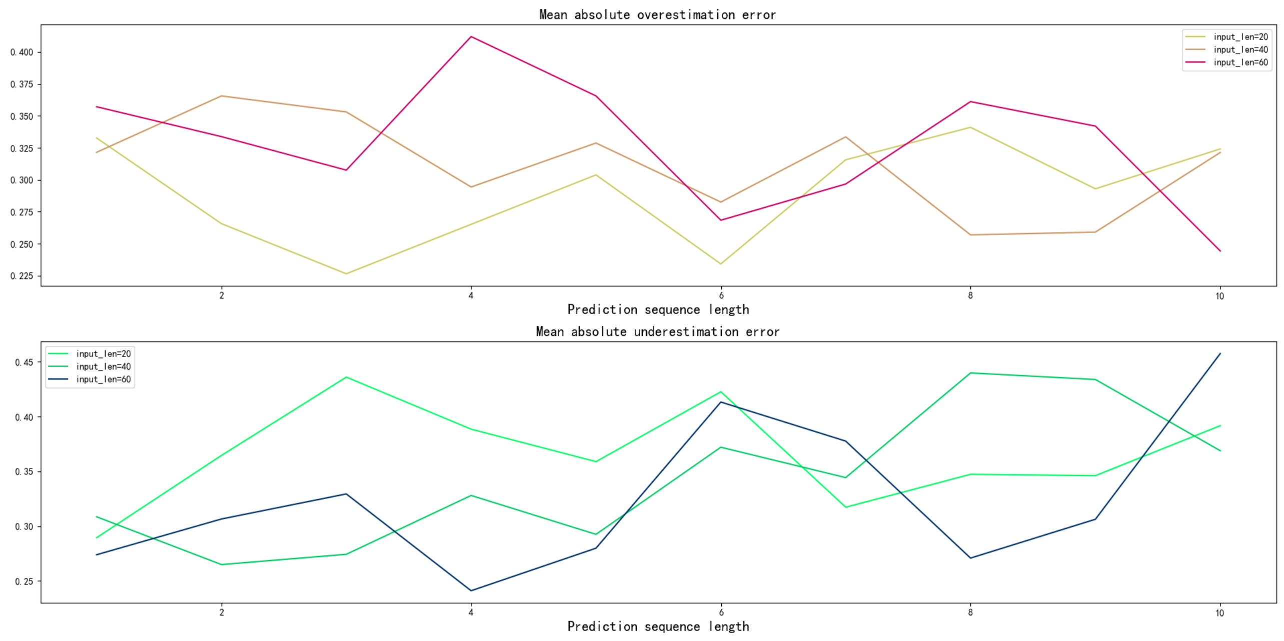


Figure 7: Overestimate and Underestimate Errors of Prediction Results of the Informer Model with Different Values of *input_len* and *output_len*

and *output_len*:

$$\begin{aligned} input\_len &\in \left\{i_1, \ldots, i_q \mid q \in \mathbb{N}^*\right\} \\ output\_len &\in \left\{o_1, \ldots, o_p \mid p \in \mathbb{N}^*\right\} \end{aligned} \tag{23}$$

When $input\_len = i_j$ and $output\_len = o_k$, our prediction result is $Y_{i_j}^{o_k} = \left[y_1, \ldots, y_{i_j}\right]$. Then we splice all the vectors of prediction results:$\left\{Y_i^{o_k} \mid i \in \left\{i_1, \ldots, i_q \mid q \in \mathbb{N}^*\right\}\right\}$ when $output\_len = o_k$, and we have:

$$Z_{o_k} = \left(Y_{i_q}^{o_k} \quad \begin{pmatrix} 0 \\ Y_{i_{q-1}}^{o_k} \end{pmatrix} \quad \cdots \quad \begin{pmatrix} 0 \\ \vdots \\ 0 \\ Y_{i_1}^{o_k} \end{pmatrix} \right) = \begin{pmatrix} y_{i_{q_q}}^{o_k} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ y_{i_{q_1}}^{o_k} & \cdots & y_{i_{1_1}}^{o_k} \end{pmatrix} \tag{24}$$

We superimpose the matrix Z_{o_k} on the third dimension, and we obtain a three-dimensional matrix M:

$$M = \begin{pmatrix} Z_{o_1} & Z_{o_2} & \cdots & Z_{o_p} \end{pmatrix}_{p \times q \times q} \tag{25}$$

Suppose the matrix composed of the prediction results at time t is M_t, then the time series prediction problem is transformed into the following problem: the input sequence is $X = \{M_1, \ldots, M_n\}$, the output sequence is $Y = \{y_1, \ldots, y_n\}$. In particular, when $input\_len = 3$ is 3 and the values in M are scaled between 0 and 255, the matrix M can be viewed as an image, as is shown in Figure8.

Different color blocks on Line i of Figure8 represent the prediction results of Bitcoin price in the i'th time interval of the future for models with different *output_len*. It can be seen that the prediction results of models with different values of *output_len* are similar, but there remains differences. These differences (between the predictions of the i'th time interval by models with different parameters and between the predictions of different time intervals by the same model) can be synthesized and extracted by the CNN to give better prediction results.

We take the matrix $M_{i \times o \times o}$ as the input of the convolutional neural network mentioned above, to output a

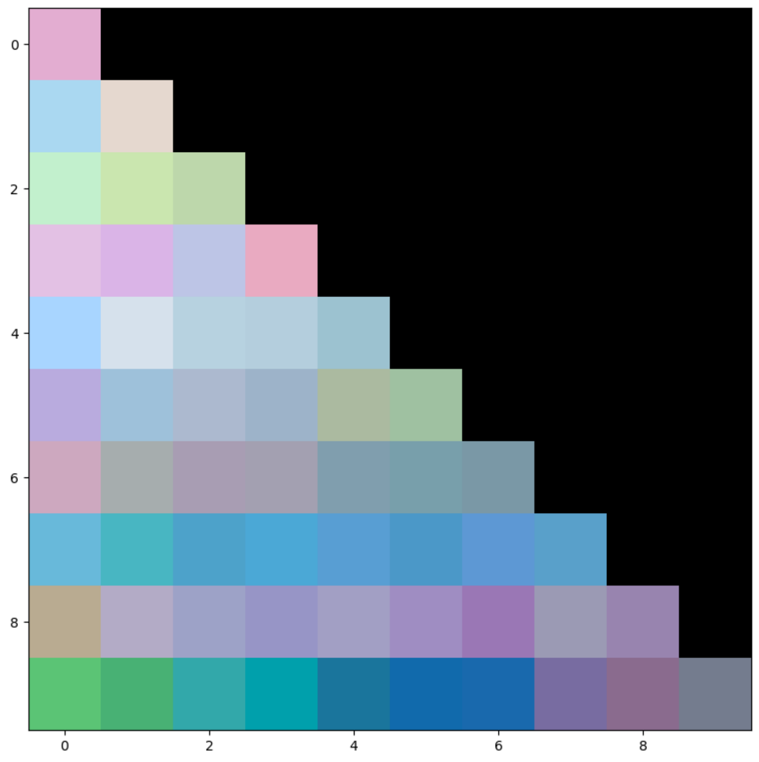


Figure 8: the Color Image Formed After Processing Matrix M When $input\_len = 3$, $output\_len = 10$

prediction sequence of length $pred\_len$, where $pred\_len \leq output\_len$.The output sequence of the CNN is the final prediction sequence of the whole model, and the overall structure of the model is shown in Figure 9.

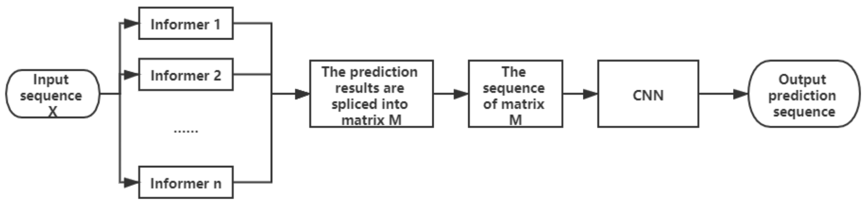


Figure 9: Informers-CNN Model Structure

For the CNN, each input matrix is a splicing of prediction results of future Bitcoin price by different Informer models at a certain point in time. We use CNN to learn the characteristics of prediction results by different Informer models at different times, rather than process the time series, so we perform a shuffling on the training set to improve the training effect and reduce the degree of overfitting.

According to the characteristics of this model, we divide the data set into different modules of training set, verification set and test set, as is shown in Figure 10.

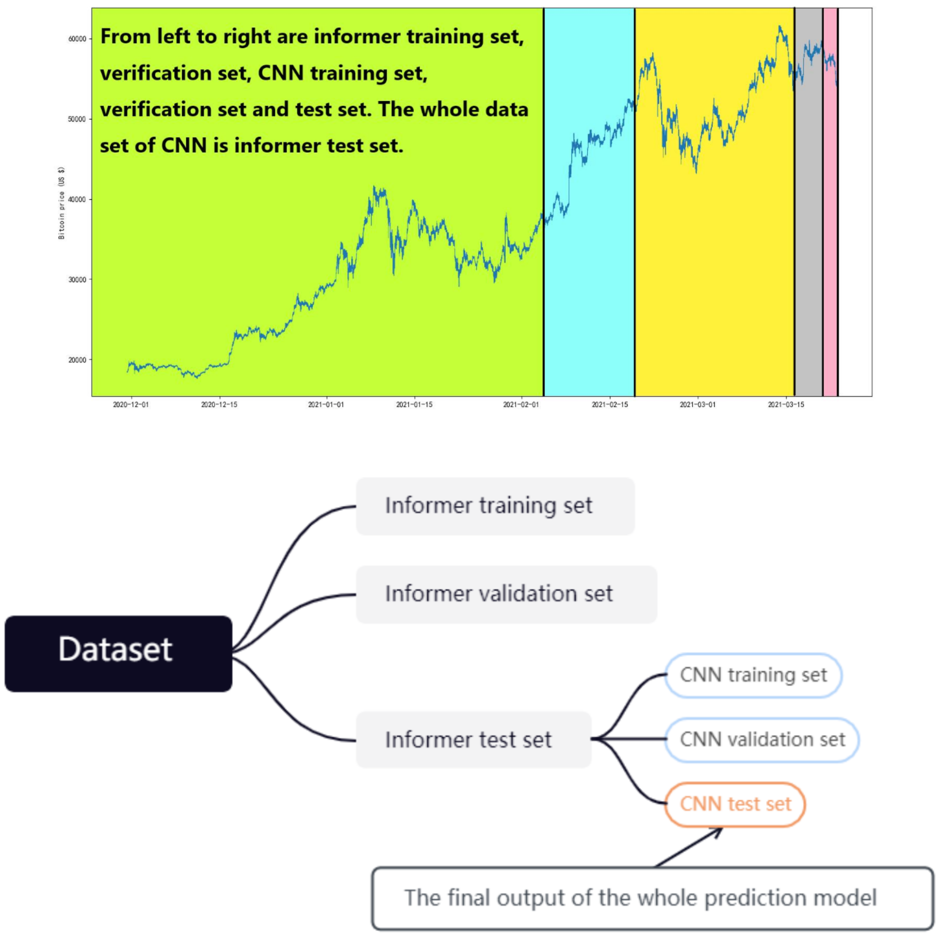

From left to right are informer training set, verification set, CNN training set, verification set and test set. The whole data set of CNN is informer test set.

Figure 10: Partitioning of Data Sets

3 Experimental Results and Comparative Analysis

3.1 Experimental Results

The entire experimental environment of our study is carried out under PyTorch, and all sets of hyperparameters in the Informer model are listed in Table4 except *input_len* and *output_len*.

parameter names	parameter values
Input vector size	29
Number of multi-head self-attention operations	8
Number of coding layers	3
Number of decoding layers	2
Probsparse Attention factor	5
Dropout Probability	0.05
Number of training batches	32
Initial learning rate	0.0001
Update of learning rate	Adam Optimizer

[1] Adam Optimizer use a combination of Adaptive Gradient and RMSProp algorithms to update the learning rate.

Table 4: Part of the Hyperparameters of the Informer Model

In order to better compare the prediction effect, we also introduce common time series prediction models, such as ARMA, ARIMA and LSTM. ARMA and ARIMA models are traditional time series prediction models based on moving average, and LSTM is an improved neural network model derived from Recursive Neural Network (RNN), which avoids the RNN's traditional problem of long dependence. If we regard the prediction sequence as a random sequence formed over time, then the interdependence of this group of random variables reflects the continuity of the original data in the time dimension and is also affected by the influencing factors and their own change laws. Suppose the influencing factors are $x_1,...,x_p$; by regression analysis, we have:

$$Y_t = \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_p x_p + \varepsilon \tag{26}$$

where Y is the observed value of the predicted object and ε is the error term. Considering the influence of its own changes on Y and the dependence relationship of error in different periods, we have:

$$\begin{aligned} Y_t &= \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots \beta_k Y_{t-p} + \varepsilon_t \\ Z_t &= \varepsilon_t + \alpha_1 \varepsilon_{t-1} + \alpha_2 \varepsilon_{t-2} + \cdots + \alpha_q \varepsilon_{t-q} \end{aligned} \tag{27}$$

Thus, we obtain the expression of the ARMA(p,q) model:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots \beta_k Y_{t-p} + \varepsilon_t + \alpha_1 \varepsilon_{t-1} + \alpha_2 \varepsilon_{t-2} + \cdots + \alpha_q \varepsilon_{t-q} \tag{28}$$

The ARIMA(p,d,q) model is an extension of the ARMA(p,q) model, and its expression is:

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) (1-L)^d X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t, \quad d \in \mathbb{N}^* \tag{29}$$

where L is the Lag Operator.

All recursive neural networks have a repeated module chain accumulation mode, and LSTM is no exception, the structure of a single module of which is shown in Figure11.

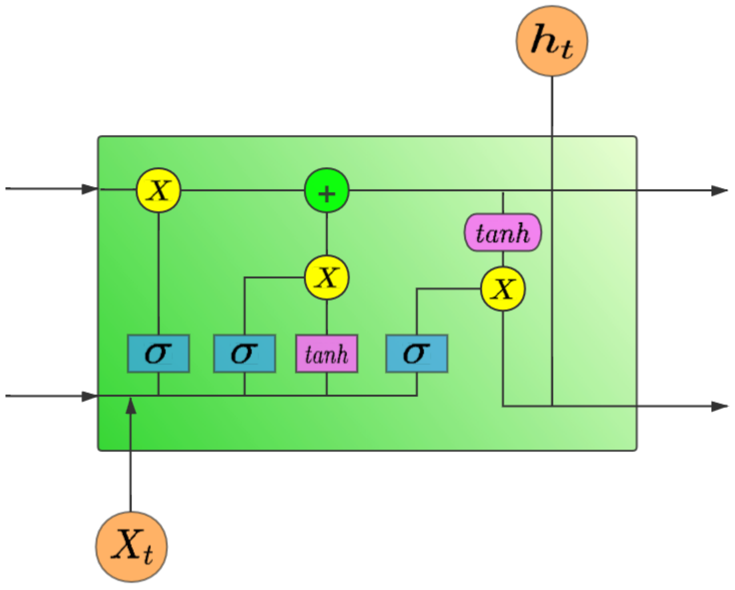


Figure 11: Structure of a Single Module of LSTM

Several modules can form an LSTM layer, and the mathematical expressions of each layer's calculation is as follows:

$$\begin{aligned}
g_t &= \phi\left(W_{gx}x_t + W_{gh}h_{t-1} + b_g\right) \\
i_t &= \sigma\left(W_{ix}x_t + W_{ih}h_{t-1} + b_i\right) \\
f_t &= \sigma\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right) \\
o_t &= \sigma\left(W_{ox}x_t + W_{oh}h_{t-1} + b_o\right) \\
s_t &= g_t \cdot i_t + s_{t-1} \cdot f_t \\
h_t &= s_t \cdot o_t
\end{aligned} \tag{30}$$

For the prediction of the value of Bitcoin price, we select mean square error (MSE) as the loss function of the Informer model, and MSE and mean absolute error (MAE) respectively as the loss functions of the Informers-CNN model and the calculation formulas are:

$$\begin{aligned}
\text{MSE} &= \frac{1}{m}\sum_{i=1}^{m}\left(y_i - \hat{y}_i\right)^2 \\
\text{MAE} &= \frac{1}{m}\sum_{i=1}^{m}\left|y_i - \hat{y}_i\right|
\end{aligned} \tag{31}$$

For the prediction of the rise and fall of prices, we treat it as a binary problem. We added Softmax function to the end of the original network structure, and used log-likehood loss function as the loss function. Under the dichotomy problem, the loss function is expressed as:

$$L(Y, P(Y \mid X)) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log p_i + (1-y_i)\log(1-p_i)) \tag{32}$$

Suppose Informer(p,q) stands for the Informer model whose $input\_len = p$ and $output\_len = q$; we train the model above and make forecasts with it, and compare the prediction results of Bitcoin prices in the first time interval. All the data of the Informer model are shown in Table5. We use the mean absolute error (MAE) as the evaluation index for the accuracy of numerical prediction of Bitcoin price.

Table 5: Predicted Effect of Informer of different *input_len* and *output_len*

Model	Loss function	Average absolute prediction error			Forecast accuracy of rise and fall		
		1min	5min	15min	1min	5min	15min
Informer(20, 1)	MSE	0.6135	0.6186	0.6222	75.48%	75.24%	75.35%
Informer(20, 2)	MSE	**0.6102**	0.6258	0.6302	76.12%	75.54%	75.94%
Informer(20, 3)	MSE	0.6123	0.6204	0.6624	75.23%	75.30%	74.91%
Informer(20, 4)	MSE	0.6125	0.6314	0.6536	75.33%	75.49%	74.84%
Informer(20, 5)	MSE	0.6389	0.6287	0.6627	74.98%	74.97%	73.27%
Informer(20, 6)	MSE	0.6358	0.6346	0.6567	74.02%	74.96%	74.52%
Informer(20, 7)	MSE	0.6410	0.6413	0.6329	74.58%	75.31%	75.57%
Informer(20, 8)	MSE	0.6541	0.6344	0.6883	73.97%	74.41%	73.08%
Informer(20, 9)	MSE	0.6398	0.6278	0.6390	75.03%	75.36%	75.85%
Informer(20, 10)	MSE	0.6356	0.6214	0.7158	73.64%	75.12%	71.12%
Informer(40, 1)	MSE	0.6243	0.6175	0.6300	75.45%	75.63%	75.85%
Informer(40, 2)	MSE	0.6201	**0.6107**	0.6305	75.01%	76.02%	76.42%
Informer(40, 3)	MSE	0.6197	0.6174	0.6274	76.00%	75.42%	76.06%
Informer(40, 4)	MSE	0.6287	0.6087	0.6223	75.86%	**75.54%**	76.75%
Informer(40, 5)	MSE	0.6312	0.6243	**0.6213**	73.39%	76.06%	76.36%
Informer(40, 6)	MSE	0.6374	0.6396	0.6545	75.09%	75.01%	74.49%
Informer(40, 7)	MSE	0.6421	0.6433	0.6778	74.20%	74.88%	73.85%
Informer(40, 8)	MSE	0.6451	0.6475	0.6966	74.54%	73.16%	73.70%
Informer(40, 9)	MSE	0.6537	0.6507	0.6928	73.28%	73.84%	74.22%
Informer(40, 10)	MSE	0.6520	0.6547	0.6901	74.30%	73.32%	72.38%
Informer(60, 1)	MSE	0.6185	0.6328	0.6310	75.88%	74.93%	**76.76%**
Informer(60, 2)	MSE	0.6140	0.6341	0.6402	75.93%	75.05%	75.81%
Informer(60, 3)	MSE	0.6277	0.6243	0.6369	76.14%	75.58%	76.42%
Informer(60, 4)	MSE	0.6289	0.6174	0.6531	**76.39%**	75.05%	76.00%
Informer(60, 5)	MSE	0.6323	0.6340	0.6454	75.13%	74.77%	75.66%
Informer(60, 6)	MSE	0.6358	0.6439	0.6815	74.84%	74.14%	73.02%
Informer(60, 7)	MSE	0.6396	0.6510	0.6742	73.36%	73.83%	74.28%
Informer(60, 8)	MSE	0.6447	0.6329	0.6320	74.54%	74.43%	75.84%
Informer(60, 9)	MSE	0.6334	0.6589	0.6482	75.09%	73.89%	74.98%
Informer(60, 10)	MSE	0.6482	0.6543	0.7017	74.30%	73.64%	73.91%

Among these models with different hyperparameters, Informer(20,2), Informer(40,2) and Informer(40,5) have the best performance in MAE, while Informer(40,4), Informer(60,1) and Informer(60,4) have the best performance in Forcast accuracy of rise and fall. Table6 shows the comparison in their prediction results of Bitcoin price in the first time interval with other models. As the ACF figure and PACF figure of the closing price (see Figure 12) all show the characteristic of non-censoring,our model takes the parameters of ARMA(1,1)and ARIMA(0,1,1)。Suppose Informers-CNN (p,q,k) stands for an Informers-CNN model with a tensor of $p*q*q$ as the input of CNN and a predicted sequence length of k as the output.

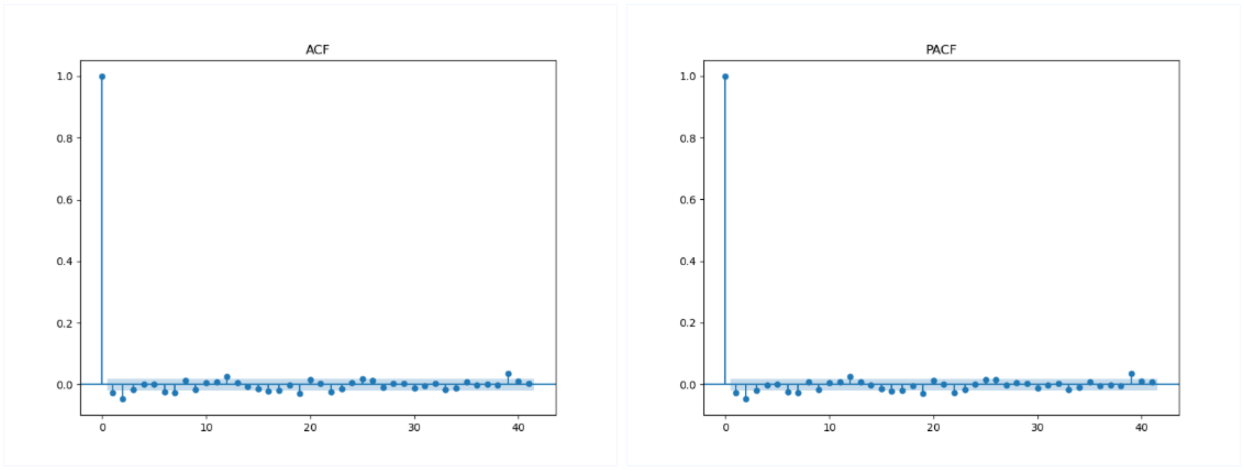


Figure 12: ACF and PACF figure

From Table6, we can draw the following conclusions:

- We employ the MAE of the prediction results of Bitcoin price in all models as our evaluation standard. As we operated differential and centralized processing on the closing price of Bitcoin, the MAE of our model's final prediction is roughly between 0.3 and 0.6, which is relatively different from that of other paper's results. In fact, the MAE obtained by other researchers is usually in the range of 0.1 and 0.2, but this is mainly due to the difference in our preprocessing of the data. Therefore, here we only consider the comparison between values of MAE, rather than the value's intrinsic meaning.
- The Informers-CNN model achieves significantly better results in mean absolute prediction error than other models.Take LSTM, in comparison, the MAE of the Informers-CNN model decreases by 56.06%, 56.14% and 55.73% in the best case respectively, in three data sets of different time intervals. Even compared with the Informer model, our model has respectively 47.33%, 48.10% and 48.27% less MAE.This, from another perspective, also verifies that the model can continue to explore the characteristics of time series in the prediction sequence, and further improve the prediction effect on the basis of this. In addition, for CNNs with inputs of small matrices like M(for our matrix $M_{i\times o\times o}$, i and o are usually small values, for instance, the values selected in our study generally are not larger than 10), how to better apply the characteristic information of the input matrix by designing the network structure specifically is also a direction of our future studies.
- Our model has a significant improvement in the accuracy of forecast accuracy of rise and fall compared with LSTM, and a relatively great progress compared with Informer. Considering that the Informer model does better in extracting the main features of time series than LSTM and ARIMA models, it has a higher accuracy rate in predicting the rise and fall of future prices. Moreover, the combination of CNN further improves the accuracy, leading to an increase of 13.07%, 13.48% and 13.50% in the three data sets respectively.
- The combination of CNN has greatly improved the prediction results, which also indicates that common

Table 6: Prediction Results of Different Models

Model	Loss function	Average absolute prediction error			Forecast accuracy of rise and fall		
		1min	5min	15min	1min	5min	15min
ARMA(1,1)	MSE	0.6147	0.6168	0.6258	52.09%	51.96%	52.42%
ARIMA(0,1,1)	MSE	0.6214	0.6178	0.6259	51.23%	51.86%	51.69%
LSTM	MSE	0.5745	0.5802	0.5984	50.48%	51.03%	51.46%
	MAE	0.5914	0.5839	0.5964	50.77%	51.35%	51.21%
Informer(20, 2)	MSE	0.6102	0.6258	0.6302	76.10%	75.54%	75.97%
Informer(40, 2)	MSE	0.6201	0.6107	0.6305	76.02%	76.02%	76.42%
Informer(40, 4)	MSE	0.6287	0.6087	0.6223	75.86%	76.54%	76.75%
Informer(40, 5)	MSE	0.6312	0.6243	0.6213	75.39%	76.06%	76.36%
Informer(60, 1)	MSE	0.6185	0.6328	0.6310	75.88%	74.93%	76.76%
Informer(60, 4)	MSE	0.6289	0.6174	0.6531	76.39%	76.05%	76.00%
Informers-CNN(3, 10, 1)	MSE	**0.3214**	**0.3159**	0.3268	85.97%	86.32%	86.97%
	MAE	0.3237	0.3169	0.3260	85.88%	**86.86%**	86.97%
Informers-CNN(3, 10, 2)	MSE	0.3888	0.3820	0.3846	**86.38%**	86.52%	86.29%
	MAE	0.3890	0.3787	0.3824	86.26%	86.63%	86.02%
Informers-CNN(3, 10, 3)	MSE	0.3849	0.3745	0.3838	85.36%	85.27%	85.56%
	MAE	0.3856	0.3719	0.3846	85.28%	85.01%	85.52%
Informers-CNN(3, 10, 4)	MSE	0.3952	0.3854	0.3949	81.88%	82.21%	81.67%
	MAE	0.3928	0.3896	0.3946	81.39%	82.37%	81.67%
Informers-CNN(3, 10, 5)	MSE	0.3898	0.3812	0.3877	85.39%	85.00%	85.84%
	MAE	0.3841	0.3814	0.3837	85.77%	85.10%	85.96%
Informers-CNN(4, 5, 1)	MSE	0.3877	0.3715	0.3784	85.03%	85.10%	84.12%
	MAE	0.3845	0.3794	0.3879	84.34%	84.74%	84.34%
Informers-CNN(4, 10, 1)	MSE	0.3364	0.3248	0.3305	85.34%	85.57%	85.69%
	MAE	0.3351	0.3254	0.3312	85.26%	85.41%	85.60%
Informers-CNN(5, 5, 1)	MSE	0.3589	0.3518	0.3556	84.31%	84.54%	84.67%
	MAE	0.3564	0.3542	0.3545	84.38%	84.37%	84.78%
Informers-CNN(5, 10, 1)	MSE	0.3286	0.3189	**0.3260**	85.67%	86.38%	**87.12%**
	MAE	0.3291	0.3214	0.3287	86.12%	86.53%	87.03%

1 The parameters of ARMA, ARIMA and LSTM models are all the values that make the prediction results best
2 In Informers-CNN(3, 10, k), $input\_len$ takes the value from 20, 40 and 60, $output\_len$ takes the value from 1 to 10
3 In Informers-CNN(4, 5, k), $input\_len$ takes the value from 10, 20, 40, 60, $output\_len$ takes the value from 1 to 5
4 In Informers-CNN(5, 10, k), $input\_len$ takes the value from 10, 20, 40, 60, 80, $output\_len$ takes the value from 1 to 10

time series prediction models are still insufficient in extracting and exploring the features of the time series with a lot of noise and increased volatility, such as the price of Bitcoin and other digital assets. Through the convolution process, we can better integrate the prediction results of multiple models and explore the hidden features of time series, so as to provide better prediction results.

Take Informer(20,1) as comparison, the prediction results of Informers-CNN (3,10,1) are shown in Figure13.

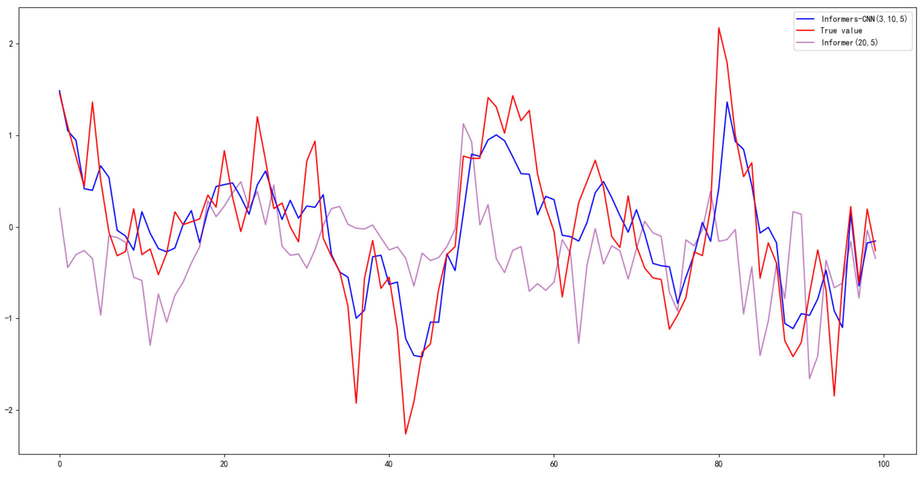


Figure 13: Comparison of the Predicted Results of Informer(20,1) and Informers-CNN(3,10,1)

We do an operation on the prediction results which is opposite of the data preprocessing, thus getting the prediction of the real Bitcoin price, see Figure 14, where the upper part is the predicted price curve and the real price curve, while the lower part is the histogram of each moment' s forecast error. From the image, we can see under the circumstance of real price, the prediction result of our model is much better than that of the Informer model.

Another advantage of the Informers-CNN model is that it can predict for multiple time intervals in the future. Take the model Informers-CNN (3, 10, *pred_len*), we set the value of *pred_len* from 1 to 10 and we consider forecasting Bitcoin's closing price for up to 10 intervals in the future. The reason why the length of the prediction sequence is set at most 10 is in consideration of Informers-CNN(3, 10, *pred_len*). In this model, the length of the longest prediction sequence in the Informer module is 10. As CNN is only used for further convolution and processing, in order to get better outcomes on the basis of the existing prediction results of the Informer model, we set the length of the longest sequence it can predict to be the length of the longest input(existing) prediction sequence, which is 10 here. Take the 15min data set as an example, the prediction results are shown in Table 7 and Table 8.

It can be seen from Table 7 and Table 8 that the Informers-CNN model still has a certain accuracy in predicting the Bitcoin price in multiple time intervals in the future, but the prediction effect gradually deteriorates when the time between the prediction and the current moment gets longer. The reason, for one part, is that with the growth of the prediction sequence, there is less information about the prediction price of this time interval in the input of CNN; for another part, as $pred_len$ increases, the overall prediction effect gets worse. When the $pred_len$ increases from 1 to 10, the predicted MAE of the price in the first time interval increases from 0.3268 to 0.3984, while the accuracy rate of judging whether price will rise or fall decreases from 86.97% to 81.27%. In addition, the model tends to have a relatively good prediction effect on the price of the last few intervals in the prediction sequence, which enlightens us to select a corresponding length of the

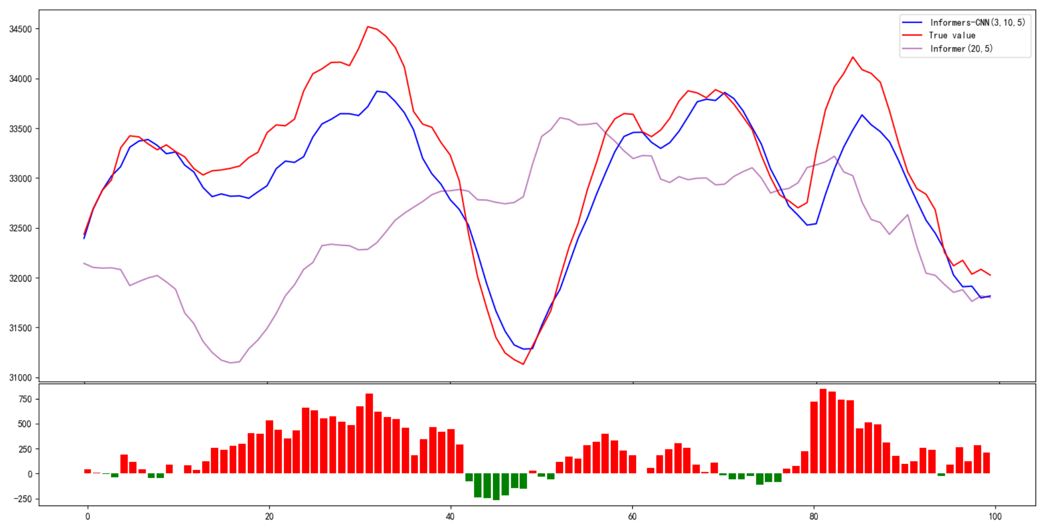


Figure 14: Comparison of forecast results on real bitcoin price

Table 7: MAE of prediction results of multiple time units in the future

pred_len	The *i*-th time unit in the future									
	1	2	3	4	5	6	7	8	9	10
1	**0.3268**									
2	0.3846	0.4806								
3	0.3838	**0.4803**	0.5394							
4	0.3949	0.4868	**0.5364**	0.5904						
5	0.3877	0.4826	0.5467	0.6100	0.6234					
6	0.3945	0.4879	0.5434	0.5949	0.6134	**0.6026**				
7	0.4042	0.4910	0.5443	**0.5899**	**0.6076**	0.6081	0.6251			
8	0.4119	0.4995	0.5524	0.5947	0.6145	0.6029	0.6250	0.6284		
9	0.4047	0.4949	0.5489	0.5963	0.6079	0.6086	0.6212	**0.6257**	**0.6185**	
10	0.3984	0.4879	0.5419	0.5924	0.6112	0.6028	**0.6176**	0.6292	0.6210	**0.6204**

Table 8: Accuracy of predicting rise and fall of multiple time units in the future

pred_len	The *i*-th time unit in the future									
	1	2	3	4	5	6	7	8	9	10
1	**86.97%**									
2	86.29%	77.78%								
3	85.56%	**80.47%**	80.10%							
4	81.67%	79.45%	78.27%	76.94%						
5	85.84%	75.54%	76.62%	74.81%	74.22%					
6	81.22%	77.43%	**80.97%**	75.81%	73.19%	73.61%				
7	82.76%	78.32%	79.33%	75.64%	**74.35%**	73.24%	**74.84%**			
8	82.21%	76.63%	78.54%	**77.89%**	73.47%	73.25%	74.00%	74.04%		
9	82.29%	76.08%	78.77%	77.31%	73.75%	73.00%	74.21%	**74.43%**	**76.19%**	
10	81.27%	77.49%	78.36%	77.41%	73.78%	**74.39%**	74.39%	74.30%	74.10%	**76.56%**

prediction sequence according to the time between the prediction and the current moment, so as to obtain the best prediction effect.

Taking Informer(20,5) as a comparison, the prediction results of Informers-CNN (3,10,5) for the next five time intervals are shown in Figure15。

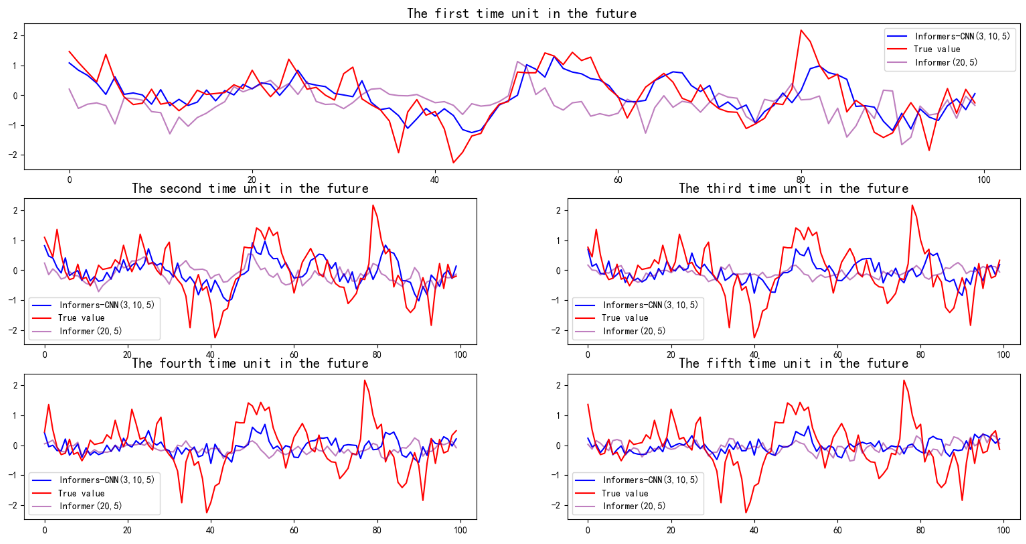


Figure 15: Comparison of the Predicted Results of Informer(20,5) and Informers-CNN(3,10,5)

In addition, compared with other models, we have a relatively longer training time. Under the configuration of the laboratory in our study, the time required to train an Informer model ranges from 12 to 20 minutes. Plus the training process of the CNN model which costs around half an hour, the total time for training the complete Informers-CNN model is approximately 8 hours, far exceeding the model's forecast time scale which is in minutes. This is a problem and unless we apply multiple GPUs to accelerate the training process and extend the time interval for the model's update, it is still hard to make true application of it. Therefore, how to quickly and dynamically update the model parameters will be a major focus of our future research.

3.2 Parameter Sensitivity Analysis

In the univariate setting, we use a 15-minute data set to carry out sensitivity analysis on some key parameters of the model. As is mentioned above,suppose Informers-CNN(p,q,k) represents the input matrix in the CNN module is $M_{p\times q\times q}$ and the output is a model with a sequence length of k. The $input\_len$ parameter in the Informer module will take corresponding subsets from $\{10,20,40,60,80\}$ according to the size of p (for instance, when $p=3$, $input\_len$ will take $\{10,20,40\}$). Similarly, the $output\_len$ of the Informer module will take corresponding subsets from $\{5,6,...,10\}$ based on the size of q. We take the MAE which predicts the Bitcoin price of the first time interval as the indicator.

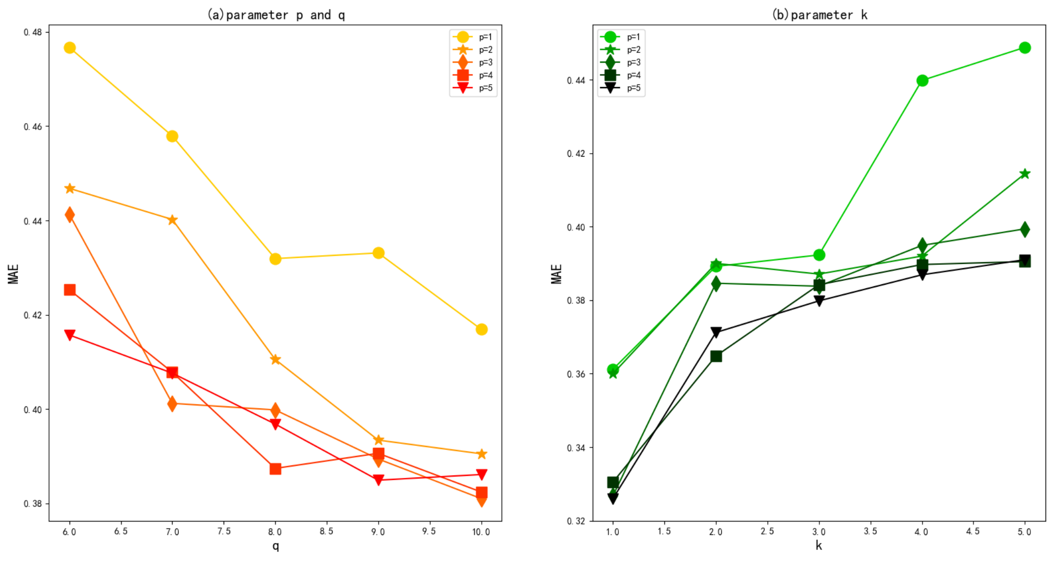


Figure 16: Sensitivity Analysis of Parameters p, q and k

Parameters p **and** q: Fix k as 5. As can be seen from Figure16a, the prediction error decreases rapidly with the increase of q, indicating that the $output\_len$ of the Informer module plays a great role in promoting the prediction effect, because the longer the prediction sequence is, the more sequence features CNN can obtain. Similarly, with the increase of p, the prediction error also decreass. However, since p reaches 3, the decrease of MAE becomes very limited, which indicates that the promoting effect of p on the prediction effect begins to weaken after p reaches a certain value.
Parameter k: Fix q as 10. As can be seen from Figure16b, the prediction effect is negatively correlated with k, but when k reaches a certain scale, the increase of MAE will not be obvious with the increase of k, . Therefore, this model can predict the value of multiple time intervals in the future and simultaneously ensure a certain prediction accuracy. Also, it can be found in the figure that when p increases to a certain extent, the prediction effect of the model will not increase significantly.

Finally, for real application of the model, how should we appropriately select values for the hyperparameters? First, in actual problems, the length of the prediction sequence that the model outputs is usually known, which is related to the specific problem. When we have determined the prediction length $pred\_len$, how do we set the values of the two key hyperparameters p and k? According to the analysis above, we believe that for the prediction task with a prediction sequence of length $pred\_len$, we set $p=2pred\_len$ and $k=pred\_len$, and set q as small as possible under the condition that the model has a good prediction effect. Under such circumstance, we may reduce the complexity and training cost of the model to the maximum extent, at the same time guaranteeing it has a good prediction effect.

4 Conclusion

Our paper studies the problem of predicting the Bitcoin price on the minute-level time intervals. We select 29 common indicators which reflect the trend of changes in Bitcoin price, construct three data sets whose time intervals are respectively 1min, 5min and 15min, and then carry out a first-order difference and a standardized processing on the data after a stationarity test of the time series.

In terms of model design, we design a new time series prediction model named Informers-CNN on the basis of the Informer model, and afterwards test its effect on prediction of Bitcoin price. By integrating multiple Informers and combining the CNN, the model is able to forecast the Bitcoin price at multiple moments in the future, meanwhile achieving a significant reduction in MAE. In addition, this model can better reflect the changing trend of the time series and has a relatively good performance in predicting the rise and fall of Bitcoin price. Considering the fact that the two key parameters $input\_len$ and $output\_len$ in the Informer module are difficult to determine, this model can make comprehensive use of the training results in the process of determining the best parameters of the Informer model, thus making a better performance.

Finally, we carry out a sensitivity analysis on the important parameters of the model, finding that the prediction effect of the model is insensitive to the parameter k, the improvement of the parameter p on the prediction effect begins to weaken after reaching a certain scale, and setting the parameter q twice or larger than twice of the length of the prediction sequence is a more stable choice.

References

[1] Achelis, S. B. (1995). Technical analysis from A to Z (2nd ed.). New York: McGraw-Hill

[2] Akyildirim, E., Goncu, A. & Sensoy, A. Prediction of cryptocurrency returns using machine learning. Ann Oper Res 297, 3–36 (2021).

[3] DuanmGege. Research on Bitcoin Price Prediction Based on ARIMA Model[J]. Modern Marketing (Next Issue) 2021(01):27 29.

[4] Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. Journal of Finance, 25,383–417.

[5] Guresen, E., Kayakutlu, G., & Daim, T. (2011). Using artificial neural network models in stock market index prediction. Expert Systems with Applications, 38, 10389–10397.

[6] Livieris I E , Pintelas E , Stavroyiannis S , et al. Ensemble Deep Learning Models for Forecasting Cryptocurrency Time-Series[J]. Algorithms, 2020, 13(5):121.

[7] Nadarajah, S., & Chu, J. (2017). On the inefficiency of Bitcoin. Economics Letters, 150, 6–9.

[8] Kara, Y., Acar, M., Boyacioglu, O., & Baykan, K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. Expert Systems with Applications, 38, 5311–5319.

[9] Othman AHA, Kassim S, Rosman RB, Redzuan NHB. Prediction accuracy improvement for Bitcoin market prices based on symmetric volatility information using artificial neural network approach. Journal of Revenue & Pricing Management. 2020;19(5):314-330. doi:10.1057/s41272-020-00229-3

[10] Vaswani A , Shazeer N , Parmar N , et al. Attention Is All You Need[J]. arXiv, 2017.

[11] Yang Xuan, Guo Sipei. Test of Adaptive Market Hypothesis in Bitcoin Market [J]. Journal of Central China Normal University (Natural Science Edition), 2019, 53(06): 864 870

[12] Zhou, H. , Zhang, S. , Peng, J. , Zhang, S. , & Zhang, W. . (2020). Informer: beyond efficient transformer for long sequence time-series forecasting.