TIME-SERIES PRICE FORECASTING MODELS FOR CRYPTOCURRENCY

20190876 - Bibek KC **20170783 - Donggyu Lee** **20160452 - Minuk Lee**

20195319 - Jawook Huh

ABSTRACT

Recent advances in machine learning have brought much attentions in the fields of finance as its advanced technologies are expected to provide practical supports in risk management, price forecasting, and portfolio optimization for professionals in finance (i.e., insurers and brokers, investment bankers, or economists). Among many machine learning problems in finance, we focus on addressing time-series forecasting problem, which is crucial in that it requires multivariate temporal feature modelling over time-wisely incoming data streams. To tackle this problem, we develop multiple deep learning-based forecasting models which are expected to efficiently learn temporal representations from time-series data. We validate our models on Bitcoin datasets (hourly and daily arranged) and the results shows that convolutional neural network-based forecasting models outperforms other forecasting models (e.g., recurrent neural network and Bayesian model) on the datasets in forecasting accuracy.

1 INTRODUCTION

Recently advances in machine learning has brought lots of attentions in the field of finance as deep neural networks provides efficient tolls to predict and analyze market prices and trends. In the major fields of finance, price forecasting in stock markets has been always a main interests for many professionals in finance. However, time-series forecasting in finance is a extremely difficult task due to lots of identified/unidentified variables which affect price, unexpected volatility over time, and high level of noise.

Among many finance markets, cryptocurrency markets is one of the popular markets, in which most of cryptocurrencies are secured by cryptography that can not be counterfeited and they are not issued by a central banks or any authority. Although cryptocurrency was introduced more than 10 years ago, the market is still in the initial phase and it's uncertain that its existence and usage continue in the future. For now, there are more than five thousand cryptocurrencies with approximately six million users in the industry Hamayel & Owda (2021). Considering its recent popularity, we focus on forecasting price in the bitcoin (Böhme et al., 2015) market that is the most popular digital asset and has brought much attentions from news and social media.

However, due to its volatility and high liquidity, the bitcoin price prediction is extremely difficult. In general, price is formed in the complicated dynamic between demand and supply and its trend and volatility is affected by lots of social and behavioral variables including economic situations, politics, international events, natural events, and human psychology. Therefore, predicting the price of cryptocurrency in the financial market has long been a challenge.

In the project, we 1) develop multiple deep learning-based forecasting models (i.e., convolutional neural network-based LeCun et al. (1995), recurrent neural network-based models, and attentional forecasting model Vaswani et al. (2017)) to predict the future bitcoin price and 2) validate our models on the bitcoin subsets which we crawled from Yahoo finance websites with two multiple tasks (One-day forecasting and long sequence forecasting). Furthermore, we analyze the experimental results with baselines we additionally built for comparison in performance.

2 RELATED WORK

Time-series forecasting As many researchers realize the importance of time-series forecasting problem, there recently have been many methods that tackle time-series forecasting task. ARIMA (Zhang, 2003) is proposed to solve the problem with differencing techqniues that enable to convert non-stationary to stationary process. Also, as deep learning techniques have developed, convolutional neural networks-based methods (LeCun et al., 1995) are proposed. LSTNet (Lai et al., 2018) introduces a CNN-based forecasting model with skip connections, which successfully captures global and local temporal representations. Among many recurrent neural network-based models proposed for forecasting problem, most models are designed to model the temporal dependencies over time. Transformer-based models (Vaswani et al., 2017) are also introduced with self-attention mechanism which effectively handles sequentially arranged data points.

Attention Mechanism Attention mechanism Bahdanau et al. (2015) is an effective approach to adaptively select a subset of features (or inputs) in an input-dependent manner, such that the model dynamically focuses on more relevant features for prediction. This mechanism works by input-adaptively generating coefficients for the input features to locate more weights to the features that are more relevant for the prediction on the given input. Attention mechanisms have achieved success with various applications, including image translation (Xu et al., 2015), machine translation (Bahdanau et al., 2015), memory-augmented networks (Sukhbaatar et al., 2015), and visual question answering (Das et al., 2017).

3 METHODOLOGY

We develop seven forecasting models with two targeted forecasting tasks: 1) One-day forecasting and 2) long-sequence forecasting task. We first describe our problem definition and describe our models in detail. This section composed of problem definition and model descriptions and we describe our models with three subsections: 1) deep learning-based forecasting model, 2) ARIMA, and 3) transformer-based sequence-to-sequence model. We select two representative models and show its architecture, depicted in Figure 1.

3.1 PROBLEM DEFINITION

First, we describe the time series problem definition. In the rolling forecasting scenario with a fixed window, we denote input as $\mathcal{X}^t = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x^t_{L_x}} | \mathbf{x^t_i} \in \mathbb{R}^{\mathbf{d_x}}\}$ at time t and the output as $\mathcal{Y}^t = \{\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y^t_{L_y}} | \mathbf{y^t_i} \in \mathbb{R}^{\mathbf{d_y}}\}$, which represents that a learning model predicts output sequence $\mathcal{Y}^t$ given $\mathcal{X}^t$. In the setting, a model enables to predict long output sequence L_y and the feature dimension d_y can be univariate or multivariate.

3.2 DEEP NEURAL NETWORK-BASED FORECASTING MODELS

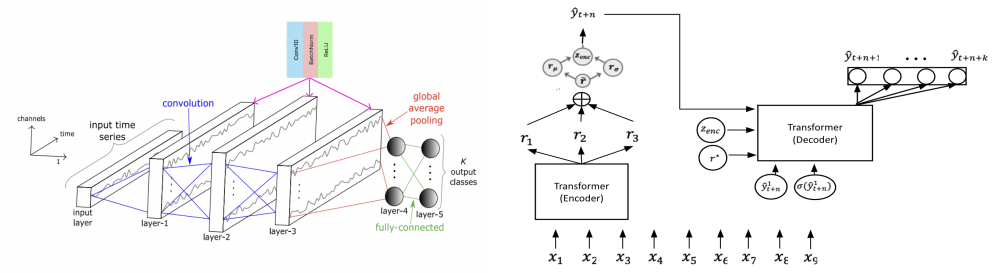
Naïve model As baseline model, we used naïve forecast for prediction. It doesn't require training and all the predicted values are equal to the value of last observation. $\hat{y}_t = y_{t-1}$. The prediction at timestamp t, $\hat{y}_t$ is equal to the value of previous timestamp $t-1$, y_{t-1}.

RNN model In our RNN model (Mikolov et al., 2010), we used a bidirectional LSTM (Long Short Term Memory) layer followed by a FC dense layer. The RNN extends the conventional Feed-Forward neural networks adding the ability of managing the variable-length sequence inputs. These RNN models can handle sequential inputs and has gates to store the information of previous inputs to handle the case of dependency between inputs and outputs.

For a formal definition of RNNs, lets assume $x = (x_1, x_2, x_3, ..., x_T)$ represents a sequence of length T, and h_t represents RNN memory at time step t, an RNN model updates its memory information using:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b_t)$$

where σ is a nonlinear function (rectified linear unit (ReLU) in our case), W_x and W_h are weight matrices that are used in deep learning model, and b_t is a constant bias.



(a) CNN-based forecasting model (b) Transformer-based encoder & decoder model

Figure 1: We present two representative model architectures: 1) CNN-based forecasting model and 2) Transformer-based sequence-to-sequence model. (a) CNN model composed of convolutional layers and feed-forward layers in which global & local pooling layers aggregate sequential data points in a sliding-window manner. (b) Self-attention mechanism in encoder-decoder structure captures temporal patterns for long sequential data points performs long sequence forecasting.

Since RNNs has difficulty in learning long term dependencies, we used LSTM (an extension for RNNs) to learn important part of the sequence and forget others. LSTM solves the gradient vanishing and long-term dependencies issues. In general, an LSTM model consists of three gates: forget, input and output gates. The forget gate makes the decision of preserving/removing the existing information, the input gate specifies the extent to which the new information will be added into the memory, and the output gate controls whether the existing value in the cell contributes to the output. As an extension to the traditional LSTM, we used bidirectional LSTM for better performance as it trains on two sequences i.e first on the input sequence and second on the reversed copy of the input sequence.

CNN model We used 1D convolutions (LeCun et al., 1995) to extract meaningful information along the time dimension. As shown in our model architecture above, the length of the time is kept constant in each layers as we are not using local pooling layers. Here, the convolution with filter n means same as applying a moving average with a sliding window of length n. The convolution for the centered timestamp t can be generalized by the following equation:

$$C_t = f(w * X_{t-l/2:t+l/2} + b) | \forall t \in [1, T]$$

where C denotes the result of convolution applied on a univariate time series X of length T with a filter w of length l, a bias parameter b and a final non-linear function f such as the Rectified Linear Unit (ReLU). The result of convolution using one filter on the input time series X can be considered as another univariate time series that underwent a filtering process. Likewise, applying several filters on a time series will result in a multivariate time series having dimensions equal to the number of filters used. Different filters on input time series are used to learn multiple discriminative features that are useful for prediction. We are using the same convolution with the same filter values, to find the result for all timestamps t [1, T]. This property called weight sharing of CNNs enables to learn the filters that are invariant across the time dimension.

Local pooling is not used as it reduces its length T by aggregating over a sliding window of the time series. But we used global average pooling layer followed by the FC layer. The global pooling operation is used to aggregate the time series over the whole-time dimension resulting in a single value. Also, the global pooling operation reduces the number of parameters in a model thus decreasing the chance of overfitting the model. Finally parameters of this CNN model are trained and the process is identical to training an MLP: a feed-forward pass followed by backpropagation.

3.3 AUTOREGRESSIVE INTEGRATED MOVING AVERAGE (ARIMA)

We develop an Auto-regressive Integrated Moving Average model (ARIMA) for ONE -day forecasting task Zhang (2003). ARIMA is designed to deal with time-series forecasting model which incorporates benefits of auto-regression (AR), moving average (MA), and integration technique for efficient temporal representation learning.

Auto-regression Auto regression model uses linear combination of previous data to predict the current data.

$$\mathbf{y_t} = \mathbf{c} + \phi_\mathbf{1}\mathbf{x_{t-1}} + \phi_\mathbf{2}\mathbf{x_{t-2}} + \cdots + \phi_\mathbf{p}\mathbf{x_{t-p}} + \mathbf{Z_t}$$

Z_t is a white noise that follows $\mathcal{N}(0, \sigma^2)$

Moving Average Moving average model uses linear combination of past forecast error to predict the current data.

$$\mathbf{y_t} = \mathbf{c} + \theta_\mathbf{1}\mathbf{Z_{t-1}} + \theta_\mathbf{2}\mathbf{Z_{t-2}} + \cdots + \theta_\mathbf{q}\mathbf{Z_{t-q}} + \mathbf{Z_t}$$

Integration Time series data should have stationarity for effective prediction, which means data's distribution should be indifferent for all time step. Integration makes non-stationary data to stationary data by levels of differencing. Differenced data are represented as the set of the differences between each time step. Also, differencing could be used multiple times. First order differencing and second order differencing are described as follows:

$$\mathbf{y'_t} = \mathbf{y_t} - \mathbf{y_{t-1}} \qquad \mathbf{y''_t} = \mathbf{y'_t} - \mathbf{y'_{t-1}}$$

Then, we incorporating these methods and ARIMA model is described as follows:

$$\mathbf{y'_t} = \mathbf{c} + \phi_\mathbf{1}\mathbf{x'_{t-1}} + \cdots + \phi_\mathbf{p}\mathbf{x'_{t-p}} + \theta_\mathbf{1}\mathbf{Z_{t-1}} + \cdots + \theta_\mathbf{q}\mathbf{Z_{t-q}} + \mathbf{Z_t}$$

ADF test We used ADF(Augmented Dickey-Fuller) test to determine d(the order of differencing). ADF hypothesis calculates the p-value with the null hypothesis "D*ata has stationarity*".

AIC We used AIC(Akaike Information Criterion) to determine the parameters which are p for auto-regression and q for moving average. L is likelihood, and if constant term c used in AR or MA is not 0, k = 1, else k = 0. The lower is better.

$$\mathbf{AIC} = -\mathbf{2}\log(\mathbf{L}) + \mathbf{2}(\mathbf{p} + \mathbf{q} + \mathbf{k} + \mathbf{1})$$

PACF We used PACF(partial Auto Correlation function) to measure the correlation between the time steps, and help to choose parameter p and q.

$$\mathbf{PACF}(\mathbf{k}) = \mathbf{Corr}(\mathbf{e_t}, \mathbf{e_{t-k}})$$
$$\mathbf{e_t} = \mathbf{y_t} - (\beta_\mathbf{t-1}\mathbf{y_{t-1}} + \cdots + \beta_\mathbf{t-k-1}\mathbf{y_{t-k-1}})$$

3.4 Transformer-based Encoder-Decoder Forecasting Model

Using deep learning techniques from an encoder-decoder prediction paradigm, we develop the transformer-based encoder-decoder architecture which targets the long-term sequence time-series forecasting problem. The overall learning pipeline is described in Figure 1-(b).

Self-attention We first briefly explain the mechanism of self-attention (Vaswani et al., 2017) and describe the model structure in details. Given the tuple inputs, query, key, and value, denoted as $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ respectively, self-attention performs the scaled dot-product as follows:

$$\mathcal{A}(Q, K, V) = \textit{Softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}})\mathbf{V}$$

where $\mathbf{Q} \in \mathbb{R}^{\mathbf{L_Q} \times \mathbf{d}}, \mathbf{K} \in \mathbb{R}^{\mathbf{L_K} \times \mathbf{d}}, \mathbf{V} \in \mathbb{R}^{\mathbf{L_V} \times \mathbf{d}}$. Considering q_i, k_i, v_i represents the i-th row in $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$, we define the attention for i-th query in a probability form as follows:

$$\mathcal{A}(\mathbf{q}_i, \mathbf{K}, \mathbf{V}) = \sum_j \frac{k(q_i, k_i)}{\sum_l k(q_i, k_l)} v_j = \mathbb{E}_{p(\mathbf{k}_j|\mathbf{q}_i)}[v_j]$$

where $p(\mathbf{k}_j|\mathbf{q}_i) = \frac{k(\mathbf{q}_i, \mathbf{k}_j)}{\sum_l k(\mathbf{q}_i, \mathbf{k}_l)}$. $k(\mathbf{q}_i, \mathbf{k}_j)$ selects the asymmetric exponential kernel $\exp(\mathbf{q}_i\mathbf{k}_j^\top/\sqrt{d})$. Self-attention incorporate these values and compute the probability $p(\mathbf{k}_j|\mathbf{q}_i)$.

		MAE	MAPE	MSE	RMSE
Task	Model				
One-day Prediction	Naïve	0.2866	3.3284	0.1554	0.3942
One-day Prediction	ARIMA	0.2949	3.4247	0.1650	0.4062
One-day Prediction	CNN (Uni)	0.2891	3.3437	0.1582	0.3977
One-day Prediction	RNN (Uni)	0.3192	3.6564	0.1825	0.4272
One-day Prediction	CNN (Multi)	**0.1482**	**1.7264**	**0.0425**	**0.2062**
One-day Prediction	RNN (Multi)	0.1679	1.9441	0.0491	0.2215
Long-sequence Forecasting	Linear Regression	0.3755	3.3681	0.1552	0.4112
Long-sequence Forecasting	Bayesian Network	0.3942	3.3112	0.1442	0.3991
Long-sequence Forecasting	Transformer E&D	0.2582	3.3493	0.1314	0.3772

Table 1: The reported numbers indicate MAE (Mean Absolute Error), MAPE (Mean Absolute Percentage Error), MSE(Mean Square Error), and RMSE(Root Mean Square Error) on the bitcoin historical dataset.

Encoder The model consists of two components: 1) Encoder and 2) Decoder, in which the encoder extract feature information from input data $\mathcal{X}^t$ and generate hidden representations $\mathcal{H}^t = \{\mathbf{h_1^t}, \ldots, \mathbf{h_{L_h}^t}\}$ and decoder generates output representations $\mathcal{Y}^t$ from $\mathcal{H}^t$. The encoder extract the long temporal representation from the long input sequences. After the extraction, $\mathcal{X}_e^t$ at t-th sequence is shaped into a matrix $\mathbf{X}_e^t \in \mathbb{R}^{L_x \times d_{model}}$. We have a similar approach with, in that we use the distilling operation to selectively choose the more meaningful features out of redundant combinations of value V and generate feature map in the further layers.The distilling procedure from j-th layer into $(j+1)$-th layer as follows:

$$\mathbf{X}_{j+1}^t = \text{MaxPool}(\text{ELU}(\text{Conv1d}([\mathbf{X}_j^t]_{\text{AB}})))$$

where $[\cdot]_{\text{AB}}$ stands for the attention block.

Decoder The decoder consists of two identical multi-head attention layers, which performs the generative inference in long sequence forecasting. We feed decoder input $\mathbf{X}_d^t$ to the decoder, which is formed as $\mathbf{X}_d^t = \text{Concat}(\mathbf{X}_{token}^t, \mathbf{X}_0^t) \in \mathbb{R}^{(L_{token}+L_y)\times d_{\text{model}}}$, where $\mathbf{X}_{token}^t \in \mathbb{R}^{L_{\text{token}} \times d_{\text{model}}}$ is the start token $\mathbf{X}_0^t \in \mathbb{R}^{L_d \times d_{\text{model}}}$ is a placeholder for the target sequence. To extract rich feature representation, we sample a L_{token} long sequence from the input before the output sequence as start token $\mathbf{X}_{\text{token}^t}$. Then the decoder predicts long sequence outputs just with one forward passing. When computing the loss, we use the *mean squared error* loss function on prediction with regards to the target sequences.

4 EXPERIMENT

4.1 DATASETS

Historical bitcoin datasets Using historical bitcoin data (Böhme et al., 2015), we preprocess the dataset and generate three subsets with different intervals: 1) every minute, 2) hourly, and 3) daily. The daily historical data of bitcoin has been collected from Yahoo. The dataset contains financial data such as high, low, open, close prices, and volume of bitcoin from 09/17/2014 to 06/05/2022. Also, single variable(close price) data was generated by removing rest of the columns in the dataset to compare the performance with multi variate data by models.

4.2 BASELINES

TASK 1: One-day forecasting
Naïve model
1) ARIMA ARIMA model used first order differenced univariate daily closing price of bitcoin.
2) CNN (Univariate) With WINDOW-SIZE=6 and HORIZON=1, CNN model uses Conv1D with Adam optimizer and MAE loss function to extract information along time axis. We are using only one feature i.e Closing price, varying over time to forecast the Closing price of one day in future.
3) CNN (Multivariate) The model architecture of multivariate CNN is same as univariate but we are using more layers of Conv1D-BatchNorm-Relu. In this case, we are using 5 features varying over time i.e Opening price, High price, Low price, Closing price and Volume to forecast the Closing price of one day in future.
4) RNN (Univariate) With WINDOW-SIZE=6 and HORIZON=1, RNN model uses bidirectional LSTM with relu activation followed by a Fully-Connected layer. We are using only one feature i.e Closing price, varying over time to forecast the Closing price of one day in future.
5) RNN (Multivariate) The model architecture of multivariate RNN is same as univariate but we are using more FC dense layers. In this case, we are using 5 features varying over time i.e Opening price, High price, Low price, Closing price and Volume to forecast the Closing price of one day in future.

TASK 2: Long-horizon forecasting
6) Linear regression model Simple linear regression model that desinged for performing time-series forecasting.
7) Bayesian network-based model Bayesian model-based global seasonality forecasting model, which is trained under a variational inference framework.
8) Transformer Encoder & **Decoder** Our model which consists of transformer-based encoder and decoder architecture. Training procedure is the same as 7).

4.3 EXPERIMENTAL RESULTS

4.3.1 NAÏVE MODEL

As shown above, the naïve forecast is lagging the test data i.e. the predicted values come slightly after the actual values, which makes sense as this forecast just use the previous timestamp value to predict the next timestamp.

4.3.2 RESULT OF ARIMA

	p-value
Original dataset	0.877978
First order differenced dataset	1.373919e-13
Second order differenced dataset	3.776752e-13

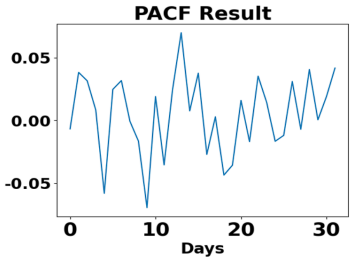
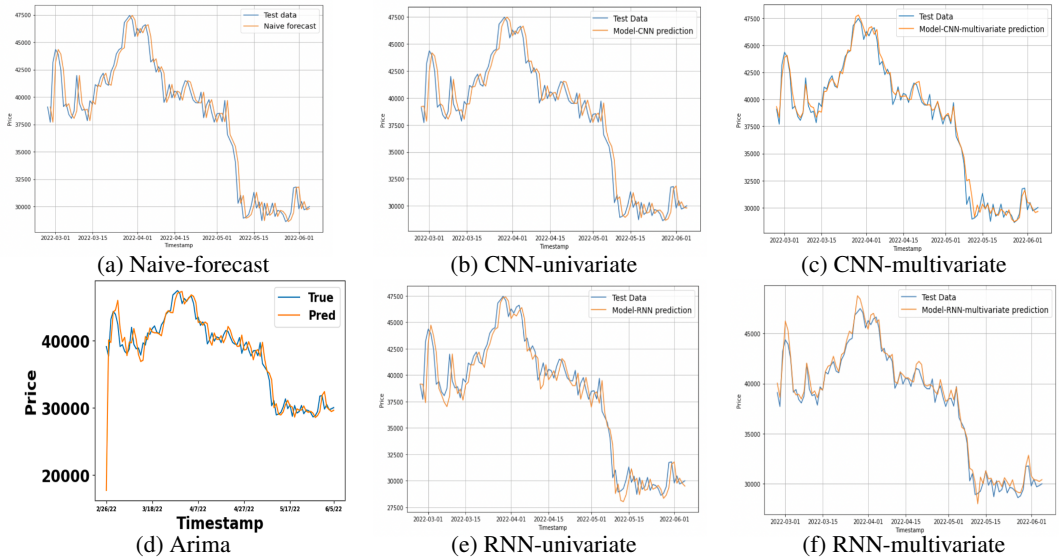


Figure 2: (a): result of ADF Tests (b): result of PACF

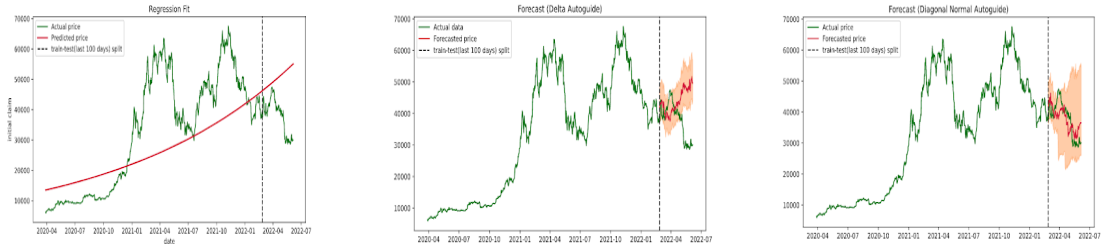
We conducted ADF test on original data, first-order differenced data, second-order differenced data, and first-order data had enough low p-value(0.05) for showing that null hypothesis(Data has a stationarity) is true. After choosing order of difference, we conducted PACF to decide p. It showed that correlation between 9 days had explainable power. We also measured the AIC value for different combination of parameters (p, d, q) for range of p = [0, 20], d = 1, q = [0, 20]. It showed that combination of (9, 1, 5) had lowest AIC value which is -5792.4577. The result shows that mae value is 0.2949, worse than 0.2866 of naive model. We think this consequence derived from the large variance of the bitcoin dataset. Model used 9 days of previous previous data for regression, but



(a) Naive-forecast (b) CNN-univariate (c) CNN-multivariate

(d) Arima (e) RNN-univariate (f) RNN-multivariate

Figure 3: The graphs represent forecasted price trajectory from 2020 (Mar) to current time point, 2022 (June) with a actual price trajectory of bitcoin price.



(a) Linear regression (b) Bayesian network (Delta auto-guide) (c) Transformer E & D

Figure 4: The graphs represent forecasted price trajectory from 2020 (Jan) to current time point, 2022 (June) with a actual price trajectory of bitcoin price. Our model (c) show that it provides more accurate price predictions (red) compared to other models and it also provides its uncertainty (orange).

volatility was so big in single days and the significance of previous price data did not have enough explanation power on prediction.

4.3.3 CNN Model

From the diagram above, it can be noticed that the CNN-multivariate model performs better than the CNN-univariate model as both line charts overlap in most of the regions. The univariate model is good at learning the correct pattern of but its forecast lags the test data plot. Its mae value is 0.2891, which is more than the baseline (0.2866) suggesting that this model doesn't beat the baseline model. The mae of multivariate model is 0.1482, which beats the baseline model by a good margin.

4.3.4 RNN Model

From above charts, it can be noticed that the RNN-univariate model is performing bad as we don't find any region where the two line plots overlap with each other. Its mae value is 0.3192, which not only is beaten by the baseline (0.2866) but is the worst model among all other models. For the multivariate case, its mae value is 0.1679, which easily beats the baseline model, but cannot beat the CNN multivariate model. It can be noticed that during sharp peaks, this model predicts little bad (predicting higher values than the actual value).

4.4 Results of long sequence forecasting task

A long sequence forecasting model enables to cover an extended period than the short sequence predictions (e.g., one day), which allows a great benefits in many practical applications such as opti-

mizing portfolio or investment protections. Table 1(bottom) summarizes the results of long sequence forecasting experiments in which we validate our transformer encoder & decoder model with two baselines: 1) linear regression- and 2) Bayesian network-based forecasting model. Under a long sequence forecasting setting, the result show that our proposed model (transformer E&D) outperforms its baseline in that the predicted trajectory is closer to the label trajectory, compared to other models' trajectories, which shows the effectiveness of the generative-style inference in transformer E&D. Beyesian network which is trained under a variational inference framework (Blei et al., 2017) hardly fit the label trajectory, which is cased by a difficulty of setting a prior for the network: we utilized delta auto-guide as a prior. Furthermore, our model provides additional information about the confidence level given predicted price such that it helps a real-world practitioners in finance when making a decision.

5 CONCLUSION

We developed seven forecasting models from deep neural networks- to Bayesian network-based model with two forecasting tasks: 1) one-day and 2) long-sequence (up to 100 days) forecasting. We validate our models on the bitcoin datasets with multiple metrics (mae, maps, mse and rmse) and perform quantitative analysis based on the results. Experimental results show that CNN-based forecasting model outperforms other models which proves that convolutional networks effectively learn multivariate features and predict the best accurate price. We found that increasing model complexity to extract rich representations doesn't perform well on our current time-series experiment setting.

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Rainer Böhme, Nicolas Christin, Benjamin Edelman, and Tyler Moore. Bitcoin: Economics, technology, and governance. *Journal of economic Perspectives*, 29(2):213–38, 2015.

Abhishek Das, Harsh Agrawal, Larry Zitnick, Devi Parikh, and Dhruv Batra. Human attention in visual question answering: Do humans and deep networks look at the same regions? *Computer Vision and Image Understanding*, 163:90–100, 2017.

Mohammad J Hamayel and Amani Yousef Owda. A novel cryptocurrency price prediction model using gru, lstm and bi-lstm machine learning algorithms. *AI*, 2(4):477–496, 2021.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 95–104, 2018.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pp. 1045–1048. Makuhari, 2010.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *NIPS*, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.