


# LSTM and GRU Scalability: A Comparative Study in Bitcoin Price Prediction

Mahdi Hallal <sup>1</sup> 

<sup>1</sup> School of Arts and Sciences, Lebanese American University, Byblos, Lebanon; mahdi.hallal@lau.edu

\* Correspondence: mahdi.hallal@lau.edu

**Abstract:** Bitcoin has received a global attention from investors, researchers, and media for being a valuable financial asset as a cryptocurrency. An unfavourable feature of Bitcoin is the high volatility of its price which causes risks for investors. To reduce the risk, researchers employed various machine learning models to predict the fluctuating price of Bitcoin. Two deep learning models, long short-term memory (LSTM) and gated recurrent unit (GRU), has shown great and better performance in Bitcoin price prediction than traditional time-series prediction models and statistical techniques. In this paper, we compare the performance of LSTM and GRU in Bitcoin price prediction. Some previous research tackled the comparison of the performance of LSTM and GRU using either a single data set or multiple data sets derived from a single data set through different partitioning intervals (e.g. 1-min, 1-hour, or 1-day) and a constant time-interval (e.g. 3-years data set). However, in this research we focus on evaluating the scalability of performance of LSTM and GRU in predicting Bitcoin price by using multiple data sets of different and increasing time intervals (e.g. a 3-years data set and a 6-years data set) that simulate the behavior of Bitcoin historical data. The features we used in this study to predict the price of Bitcoin are OHLC (open, high, low, and close) prices, Bitcoin volume (BTC volume), and Tether USD (USDT). We employed three data sets of variant and increasing time-intervals to predict the price of Bitcoin: 1-year data set, 2-years data set, and 4-years data set. The performance of LSTM and GRU was measured using mean absolute percentage error (MAPE) and root mean square error (RMSE). Experimental results show that LSTM has a more scalable and stable performance than GRU in Bitcoin price prediction as the time-interval of the data set increases.

**Keywords:** Bitcoin; price prediction; deep learning; long short-term memory; gated recurrent unit; time-series data; time-interval; time-step; scalability

---

## 1. Introduction

The concept of virtual currency has become a sought-after and accepted means of financial transactions worldwide. A major subcategory of virtual currencies that has been receiving a lot of attention in terms of investors and media is cryptocurrencies [31]. A recent study revealed that over 18,000 cryptocurrencies are being traded by institutions and individuals on a global scale [29]. The characteristics of cryptocurrencies such as security, speed of transfer, and transparency are the reasons behind their immense popularity [32]. Bitcoin is the first and most popular cryptocurrency that was launched in 2008 by Nakamoto and is legal for trading in more than thirty different currencies [30]. Bitcoin is not controlled by any central government or bank authority that manages the currency supply, thus making Bitcoin a decentralized cryptocurrency. Two entities can directly exchange Bitcoin provided by public and private keys each. The technology behind Bitcoin is called blockchain technology where every transaction is recorded and encrypted [33]. The username of Bitcoin holders is not included during transactions; only the wallet ID is made public. These properties led Bitcoin and other cryptocurrencies towards being valuable assets in the world. An unconventional feature of Bitcoin is that its price fluctuates immensely during short periods of time unlike traditional financial assets (such as commodities, stocks, and gold). Figure 1 shows clearly that Bitcoin volatility is much higher than that of Gold, H5300, and S&P500 [3]. It is normal for the fluctuation rate to be between 5% and 10%, and sometimes between 20% and 30% in very short intervals of few hours. Figure 2 gives more insight into the fluctuation rate of Bitcoin as the price (in USD) on 10 May 2020, dropped approximately from 9,500 to 8,200 in a few hours [28]. The leading factors of Bitcoin price continuous fluctuations are the lack of government regulations, fake news that manipulate investors to make unreasonable decisions, and major global events [34]. This high volatility of Bitcoin is the main threat facing its users and investors.

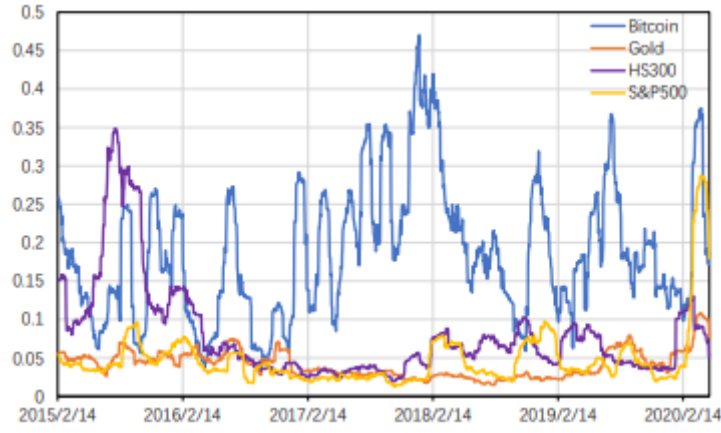


Figure 1. Historical volatility of various financial assets over 30 days [3]

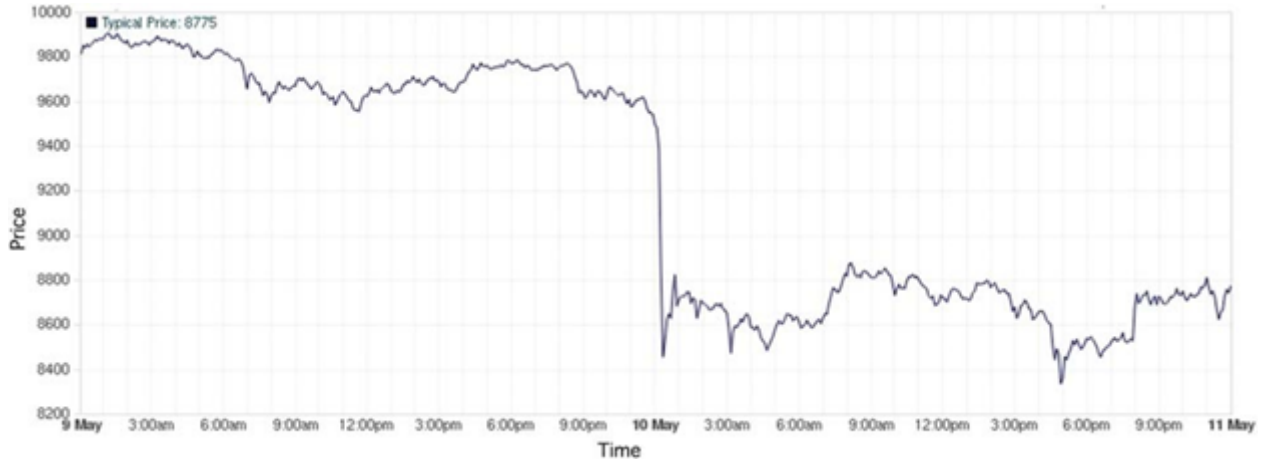


Figure 2. Historical volatility of Bitcoin in 2020 between May 9 and May 11 [28]

### 1.1. Motivation

Although the price fluctuation problem causes concerns for Bitcoin investors, the same fluctuation has the potential to make users gain huge capital if a mechanism to predict the drastic changes in the price is present [35]. Technical solutions for price prediction have been heavily relied on in stocks market price prediction as they were accurate and profitable [36]. A significant amount of research is invested in finding algorithms to predict the price of Bitcoin. There exist numerous amount of endogenous and exogenous factors that affect the price of Bitcoin, thus making price prediction a complex process. This same complexity results in price fluctuating behavior that, if predicted with a decent accuracy, will carry great financial gains that are worth the trial of finding price prediction models to handle the sophisticated behavior of Bitcoin price volatility. The Machine Learning field has shown great promise in providing reliable predictions for the Bitcoin price [37]. Most of the research focusing on Bitcoin price prediction found that Deep Learning models are exceptionally effective in dealing with the nature of this problem [37]. This nature is based on time-series historical data that creates a dependency between all the previous prices of Bitcoin and the price to be predicted. Short-term prediction models that capture short-term dependencies are not as accurate as prediction models that are capable of capturing patterns between long sequences of Bitcoin historical data [6]. Previous Bitcoin price prediction research emphasized the remarkable ability of two Deep Learning algorithms in dealing with time-series data to produce very accurate results. These two models are Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) that have been heavily relied in previous research to perform Bitcoin price prediction [2,3,6,10,13,16,27,28]. Both LSTM and GRU are built upon the principle of storing relevant and useful information from a whole sequence of time-series data and getting rid of irrelevant information in the time series. The architecture design of LSTM and GRU is one of the best in terms of predicting the price of Bitcoin relying on time-series historical data. It is no doubt that both LSTM and GRU displayed superiority

in predicting the price of Bitcoin when compared with other models, but it is still unclear which model of the two is more worthy of investing in. The comparison between LSTM and GRU in previous literature has been based on either data sets of fixed size, like the data set used in [6], or data sets of variant sizes that are derived from the fixed-size data sets through interval-partitioning, similar to the comparison technique used in [3,28]. A fixed-size data set for Bitcoin is a time-series data set that have a fixed-size of time-steps in the time-series. For example, a daily time-series data set of 3000 days where each day represents a time-step. There is yet to exist previous literature that uses Bitcoin data sets of varying sizes through variations in time-series intervals (e.g. 3-years data set and 6-years data set) and not through variations of partitioning intervals over the same time-series interval (e.g. a 1-year data set partitioned by 1-hour interval and 4-hours interval). In this paper, we introduce the comparison between LSTM and GRU based on data sets of different sizes through the usage of different time-series intervals.

### 1.2. Contributions

A majority of previous research has focused on comparing the performance of LSTM and GRU with other Deep Learning, Machine Learning, and statistical models in Bitcoin price prediction using a data set of fixed size. Research of such nature strengthened the notion that LSTM and GRU are one of the most optimal and practical solutions for Bitcoin price prediction. Other previous literature attempted to highlight the differences between LSTM and GRU by doing price predictions on Bitcoin data of varying sizes. This technique is implemented by taking data of fixed size and partitioning it according to different partitioning time-intervals (for example 1-min, 1-hour, and 1-day) thus deriving new data sets of larger size. Although this methodology is helpful in figuring out and comparing the behavior of both LSTM and GRU when exposed to an increasing size of data, it leaves out one critical parameter that is essential for comparing the two models, behavior when exposed to a new and increasing number of patterns. Through partitioning, LSTM and GRU performances are compared based on a data set of increasing size but with the same patterns, no new data or information is added to the time-series sequence. All investors of Bitcoin would definitely prefer a prediction model that is accurate and reliable for a long-term period rather than a short-term period. A model worth investing in should be capable of maintaining excellent price prediction results when exposed to new data patterns carrying information that is accumulated to the information gained from previous data patterns. In real-world predictions, Bitcoin time-series data increases as more time passes by, so it creates new patterns of data since the increase in size is due to the legitimate addition of new unseen data and not the partitioning of existing data. This research paper contributes to the comparison of LSTM and GRU to give investors a better idea of which model would be more beneficial in their financial business. We compare the performance of LSTM and GRU using three data sets: 1-year data set, 2-years data set, and 4-years data set. Each of the data sets is partitioned by hour creating 1-hour interval data sets for each of the three data sets. The features we used for Bitcoin price prediction in this research are OHLC features (opening price, high price, low price, and closing price) in addition to Bitcoin volume (BTC volume) and USDT volume. We chose these features as they are direct and pivotal endogenous determinants of the Bitcoin price [8,9,11,13,16]. The hourly closing price is the value we predict in this research following a regression prediction type. The varying and increasing size of the datasets will lead to results that reflect which model's performance is more scalable with the increase in data frequency. The 1-year data set is a subset of the 2-years data set and the 2-years data set is a subset of the 4-years data set. This ensures that the comparison of LSTM and GRU will not only include an increased frequency of data, but also the introduction of new patterns to existing old ones. The design of the data set allows experimenting with the scalability of LSTM and GRU with both increasing data frequency and increasing and new data patterns. The superior model is the model whose performance will be more scalable after training on the 1-year data set, then the 2-years data set, and finally, the 4-years data set. Scalability is a critical factor in the comparison process since the better model is not the one with higher accuracy after training on the 4-years data set, but the model that will maintain its performance as the frequency and patterns of data will increase from 1-year to 2-years to 4-years. A scalable performance (maintaining steady performance or producing better performance) indicates that the given model will eventually achieve higher accuracy than the other model as long as the size of data and patterns continue on increasing, which is a definite thing for Bitcoin time-series data.

## 2. Related Work

Bitcoin price prediction can be approached in two ways. The first way is to predict whether the price is going to rise or fall which is called classification. The second way is to predict the exact price at a fixed decided

interval of time called regression. In [1], the work focuses on comparing the performance of multiple Deep Learning and Machine Learning models in both regression and classification prediction. The results of the study show that LSTM has the best performance among the Deep Learning models used (LSTM, DNN, CNN, ResNet, CRNN, and Ensemble) in both regression and classification. An improved version of the architecture of LSTM is proposed in [17,23]. In [17], LSTM is combined with Convolutional Neural Networks (CNN) to form the CNN-LSTM model in which CNN filters out the best features to predict Bitcoin price and an LSTM layer uses these selected features to predict Bitcoin price. Support Vector Regression (SVR) is combined with LSTM in [23] to produce SVR-LSTM, a two-stage framework where LSTM takes as an input the features that are chosen as an output of SVR. Another attempt to gain insight into the optimal features that can be chosen for Bitcoin price prediction is proposed in [19] using Bayesian Regression. In [1], the authors used a data set of 7 years to exhibit that classification models outperformed regression models for algorithmic Bitcoin trading. The outcome of [25] opposes that of [1] by conveying that regression models generally outperformed classification models in Bitcoin price prediction problems. The usage of hybrid models in financial predictions is getting a lot of attention lately due to their phenomenal results. Hybrid models are models that combine different properties of different models to create a new model that is specifically optimized for a specific problem. [3] uses a hybrid model called Attentive LSTM and Embedding Network (ALEN) in Bitcoin price prediction. This model is compared with other highly effective models such as LSTM and GRU. The methodology of the research considers a data set of 3-years that is partitioned into 5-min, 15-min, 1-hr, and 4-hr datasets. Results reflected the superiority of the hybrid ALEN model over the rest of the models in all the different-size datasets present. In [7] the authors also introduced a hybrid model combining Recurrent Neural Networks (RNN) and CNN called DRCNN. DRCNN outperformed other machine learning models thus validating the effectiveness of hybrid models. Support for the notion that Deep Learning techniques are the best in Bitcoin price prediction was established in [5]. In this study, the Bayesian Neural Networks (BNN) architecture outperformed both Linear Regression and Support Vector Regression models. Moreover, [4,11,12] showcase that LSTM has better accuracy in Bitcoin price prediction than the Autoregressive Integrated Moving Average (ARIMA) model and Linear Regression. On the other hand, [8] argues that Bitcoin price prediction can be performed efficiently with the usage of classical machine learning and statistical techniques such as Linear Regression and Darvas Box. Some data manipulations needed to be done to resolve the Random Walk problem that makes the patterns underlying the Bitcoin time series seem random and complicated. The authors in [12] manipulated the data through partitioning to avoid the random walk problem of the Bitcoin time series. This process led linear machine learning models to perform better than neural network models. Furthermore, the work in [9] attempts to predict the price of Bitcoin using Decision Trees and Linear Regression using a huge daily data set that contains 8-years Bitcoin data. There is a positive correlation between the size of the data set used in training the model and the performance of the model in the testing phase. The decision of the enormous data set attempts to optimize the chances of the machine learning models of efficiently learning the patterns of the Bitcoin time-series data. The results of this work displayed an exceptionally high accuracy for both Linear Regression and Decision Trees in Bitcoin price prediction. According to [11], the design of both LSTM and GRU was inspired by the design of RNN. RNN is punctual in the usage of time-series data, however, the model suffers from a short-term memory problem meaning that the model is ineffective in learning from long sequences of data. This flaw is caused by the vanishing gradient, where the weights of the neural network stop updating causing the model to stop learning. The work in [6,21] highlights the difference between LSTM, GRU, and RNN in addition to the ARIMA model over a data set of a fixed size. Results reflect the vanishing gradient problem of RNN as both LSTM and GRU outperformed the RNN model in Bitcoin price prediction with GRU achieving the highest accuracy. Enhancements to LSTM and GRU models were established in [2]. The authors in [2] investigate the performance of LSTM and GRU with the presence and absence of recurrent dropout, a model-design decision used to optimize the learning of the models, using a fixed-size daily Bitcoin data set. GRU with the presence of recurrent dropout outperforms all other models used in this study for Bitcoin price prediction. Research in [13,16,27] supports the results attained in [2,6] where a GRU model achieves higher accuracy in Bitcoin price prediction than LSTM. Although in studies [13,27] the GRU model outperformed other statistical techniques such as Theil-Sen Regression and Huber Regression by using a huge data set of 6-years that is partitioned into 1-minute intervals, the GRU model used in study [16] had a lower accuracy than the statistical models used, Hidden Markov Models, on the problem using a data set also of 6-years with 15-min interval partition. The shift of performance between Machine Learning models and statistical models is further investigated in [20]. In this study, Machine Learning models outperformed statistical methods when high-frequency data was used while



statistical methods outperformed Machine Learning models with the usage of low-frequency data with a large number of Bitcoin features (high-dimensional features). Another study that compares the performance of RNN, LSTM, and GRU is [10], but the study also attempts to compare the usage of different currencies (AUD, CAD, CNY, EUR, GBP, and JPY). Unlike [6], [10] showcases that LSTM outperforms both GRU and RNN over a data set of 3-years which is similar to the data set used in study [6]. The variation of the results between studies [6] and [10] is a clear indication of the unreliability of using a fixed size data set to determine which of LSTM and GRU is superior to the other. There exist many external factors that affect the volatility of Bitcoin price, some of these are highlighted in [14,15,17,18]. In [14,18] Google trends and VIX index are taken as features for Bitcoin price along outputs of the Garch statistical models used explicitly in study [14], while in [15,18] Twitter sentiments relating to Bitcoin news carried out a correlation with the fluctuating price of Bitcoin. [17] took the consideration of external factors even further by including crude oil future prices, gold price, VIX index, NASDAQ index, federal funds rate, and Yuan-dollar exchange rate. The authors in [22] introduce the usage of Stacked Denoising Autoencoders (SDAE) Deep Learning algorithm in Bitcoin price prediction. Results clarified that SDAE has a better prediction performance than SVR, BPNN, and PCA-SVR. [24] focuses more on the comparison of diverse Deep Learning models in Bitcoin price prediction. SVM, DNN, and Random Forest (RF) performance were compared, and the results showed RF has the highest accuracy. Some research such as [26] investigates the relation between Bitcoin prices and other cryptocurrencies' prices. The work consists of predicting the price of three different cryptocurrencies (Bitcoin, Ethereum, and Ripple) where each one of the virtual currencies has the other two as features to experiment with the association between the cryptocurrencies in the price prediction of any of them. The models used in this experiment are two hybrid models: Multiple-Input Deep Neural Network, MICDL (CNN layer, LSTM layer, Dense layer, Pooling layer, Dropout layer, and Batch Normalization layer), and a CNN-LSTM model similar to the one used in [17]. MICDL outperformed CNN-LSTM in both regression and classification. A study of the difference in the behavior of LSTM and GRU when exposed to both historical and real-time data was done by [28]. In this study, the data set of almost 4-years is partitioned into 4-hr interval, 8-hr interval, and 12-hr interval. The results show that LSTM outperformed GRU in the 4-hr interval data set, but GRU outperformed LSTM in the 8-hr interval and the 12-hr interval datasets.

In this paper, we focus on comparing the performance of LSTM and GRU models in Bitcoin price prediction based on three datasets that are of varying sizes through adding more time-interval to a data set with a constant partitioning interval. This research takes into consideration a 1-year data set that increases into a 2-years data set and finally becomes a 4-years data set, all of which have a constant 1-hr partitioning interval. We used datasets to cover not only the variation in data size but also the variation in data patterns. The approach taken in this study has not been adopted by any of the previous research comparing LSTM and GRU. Although some previous research focused on highlighting the differences between the two models by having a data set of varying size through partitioning [3,28], the usage of a data set of increasing time-interval with a constant partitioning has not been focused on by any previous research. The data set design of this research paper allows the experimentation of the effects of the scalability of both data size and data patterns on the accuracy of prediction of LSTM and GRU. The results will reflect the model that is better employed for long-term investments that rely heavily on an adequate prediction accuracy for data of increasing size and patterns.

### 3. The Prediction Models

Recurrent Neural Network (RNN) is a type of neural network that has a good performance in modeling sequence data for predictions [21]. A RNN has an input layer, a hidden layer, an output layer, and a hidden state that acts as a loop passing information from one iteration to the next iteration as shown in Figure 3. The hidden state plays the role of memory in the RNN since the hidden state accumulates information from previous inputs until the model finishes learning. The result is a model that learns the associations in a sequence of data. However, RNN suffers from a short-term memory problem [6]. If the model is exposed to a long sequence of data, it will have a problem preserving information from earlier steps. The short-term memory problem is induced by the vanishing gradient problem; the model's inability to learn [2,28]. Time-series modeling (such as Bitcoin time-series) is characterized by long sequences of data, thus the usage of RNN in problems that rely on time-series modeling will result in a vanishing gradient due to the nature of backpropagation; the algorithm used for the training and optimization of neural networks. During training, the RNN model undergoes a forward pass to make predictions, then a loss function is used to compute the difference between the predicted value and the actual value. This difference is the error value which indicates the performance of the network. The error value is heavily relied on during backpropagation to calculate the gradients. Gradients are used to

adjust the weights of the nodes in the network thus allowing the network to learn (the smaller the gradient, the smaller the adjustments, and vice versa). The design of RNN is based on layers, so when backpropagation occurs, all nodes in a given layer calculate their gradients with respect to the changes of the gradients that happened in the previous layer. If the previous layer has small adjustments in the weights, the current layer will have even smaller adjustments, an example of that is shown in Figure 4. As the gradient propagates down it will shrink exponentially causing the early layers not to adjust weights and stop learning, and this is the vanishing gradient problem. The incapability of learning on earlier time-steps in the sequence of time-series data leads to the short-term memory problem in RNN.

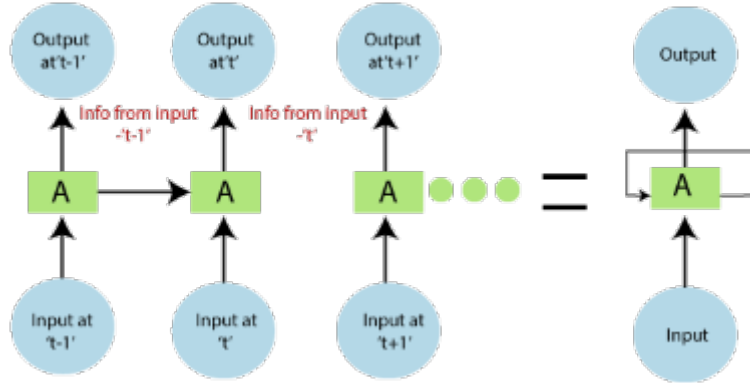


Figure 3. RNN internal structure. A: hidden layer,  $\rightarrow$  : passing hidden state from one time-step to the next time-step [40]

$$\text{new weight} = \text{weight} - \text{learning rate} * \text{gradient}$$

$$\boxed{2.0999} = \boxed{2.1} - \boxed{0.001}$$

Not much of a difference                      update value

Figure 4. An example of RNN vanishing gradient problem [41]

### 3.1. LSTM

Long short-term memory (LSTM) was first introduced in [38] as a special type of RNN. The LSTM model was derived from the RNN model with the purpose of solving the short-term memory problem that RNN suffers from. LSTM works similarly to RNN when processing data, the model passes on data as it propagates forward to learn the patterns in an entire sequence. Nevertheless, the differences between LSTM and RNN are found in the internal structure of a LSTM cell shown in Figure 5 and the operations taking place within the internal structure shown in Figure 6. The operations present in Figure 6 are based on mathematical notations that are introduced and described in Table 1. The internal mechanism of LSTM is based on gates. Gates regulate the flow of information by learning which information in the sequence is important to keep and which can be neglected. This ensures the passage of relevant information all the way through long sequences to make reliable predictions. The internal architecture of LSTM consists of a forget gate, an input gate, an output gate, and a cell state (Figure 5). The cell state is extremely important in LSTM as it takes the role of the memory of the network. The cell state is responsible for the transfer of relative information throughout the entirety of the data sequence. This reduces the effects of the short-term memory problem since data from early steps can be present in later time steps. The other three gates are responsible for learning which information is relevant to keep or forget, and controlling which information is permitted in the cell state. Gates add information to the cell state if it is relevant and remove information from the cell state if it is irrelevant.

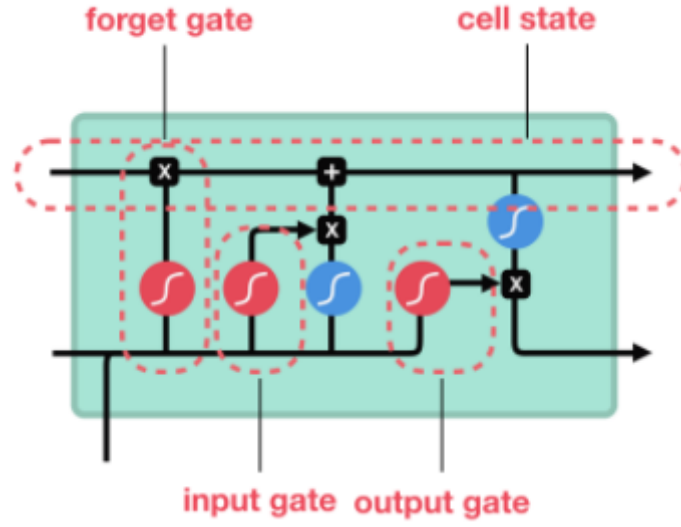


Figure 5. The architecture of a long short-term memory (LSTM) cell-gates [41]

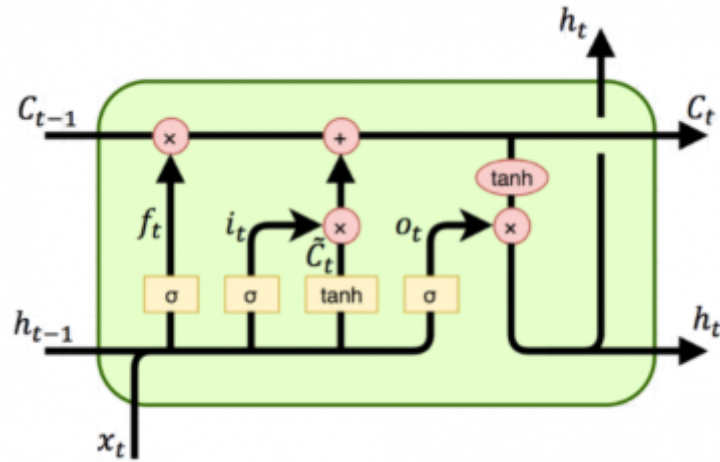


Figure 6. The architecture of a long short-term memory (LSTM) cell-operations [42]

Notation	Description
$\sigma$	Sigmoid Activation Function
$\times$	Pointwise Multiplication
$+$	Pointwise Addition
$\tanh$	tanh Activation Function
$C_t$	New Cell State
$C_{t-1}$	Previous Cell State
$h_t$	New Hidden State
$h_{t-1}$	Previous Hidden State
$f_t$	Forget Gate Output
$i_t$	Input Gate Output
$o_t$	Output Gate Output
$x_t$	Current Input
$\tilde{C}_t$	Candidate Cell State

Table 1. Description of the notations used in the operations of the internal architecture of LSTM

The forget gate decides which information is important to keep from prior steps in the sequence and which to discard. In this gate, information from the current input,  $x_t$ , and the previous hidden state,  $h_{t-1}$ , are passed through the Sigmoid function,  $\sigma$ . The Sigmoid function compresses the values between the range of 0 and 1. The output of the Sigmoid function helps decide which information to forget and which to keep. A number multiplied by 0 is 0, meaning the values disappear and are forgotten. A number multiplied by 1 remains the same, thus the value is completely kept. This gives the forget gate the ability to decide which information to keep or disregard. The closer the output of the Sigmoid function to 1 means to keep, while the closer to 0 means to forget. The output of the forget gate,  $f_t$ , is represented by Equation 1 where  $W_f$  and  $b_f$  are respectively the weight matrix and the bias vector of the forget gate.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1)$$

The input gate decides which information is relevant to add from the current step in the sequence. The output of the input gate,  $i_t$ , is calculated by passing the previous hidden state,  $h_{t-1}$ , and the current input,  $x_t$ , into a Sigmoid function,  $\sigma$ . This output is used to update the cell state. The Sigmoid function is used to decide which values are important to be included in the updated cell state. The process is represented by Equation 2, and it is very similar to Equation 1. The only difference is the usage of different weight matrix and bias vector. Another factor that is used to update the cell state is the candidate state. The candidate state,  $\tilde{C}_t$ , is calculated by passing the previous hidden state,  $h_{t-1}$ , and the current input,  $x_t$ , into a tanh function as shown in Equation 3 with  $W_c$  being the weight matrix of the cell state and  $b_c$  being the bias vector of the cell state. The tanh is an activation function that regulates the flow of values in the neural network. It takes the input and outputs a value between -1 and 1. Vectors go through multiple transformations in a network causing them to either explode with astronomical numbers, making other values seem insignificant, or shrink exponentially, making themselves insignificant compared to other values. A tanh function's job is to ensure that the values remain between -1 and 1 to maintain their significance when compared to other values.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (3)$$

Equation 4 illustrates the process of calculating the new cell state,  $C_t$ , used to pass relevant information in data-sequence learning. The output of the input gate,  $i_t$ , and the candidate state,  $\tilde{C}_t$ , are pointwise multiplied. The important information from the candidate state,  $\tilde{C}_t$ , is chosen to be passed to the new cell state depending on the output of the Sigmoid function,  $\sigma$ , used to compute the output of the input gate,  $i_t$ . The output of the multiplication of the candidate state,  $\tilde{C}_t$ , and the output of the input gate,  $i_t$ , is added to the multiplication of the output of the forget state,  $f_t$ , and the previous cell state,  $C_{t-1}$ , to produce the new cell state,  $C_t$ . The multiplication of the previous cell state,  $C_{t-1}$ , with the output of the forget gate,  $f_t$ , computed by the Sigmoid activation function,  $\sigma$ , results in dropping and forgetting values that are multiplied with values close to 0. This design ensures that the new cell state,  $C_t$ , continuously contains updated values that are relevant and keeps them throughout the entire sequence.

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \quad (4)$$

The output gate helps in figuring out the new hidden state,  $h_t$ , for the next step. According to Equation 5, the current input,  $x_t$ , and the previous hidden state,  $h_{t-1}$ , are inputted to the Sigmoid function,  $\sigma$ , to produce the output of the output gate,  $o_t$ . The output of the output gate,  $o_t$ , is then pointwise multiplied with the tanh of the new and updated cell state,  $C_t$ , to determine the new hidden state,  $h_t$ . This operation is expressed by Equation 6. The output of the output gate,  $o_t$ , that is figured out through the Sigmoid function,  $\sigma$ , is multiplied with the tanh of the modified cell state,  $C_t$ , to decide which information from the current time step should the hidden



state,  $h_t$ , carry to the next time step. The new cell state,  $C_t$ , that contains relevant information from previous time steps and current time step and the new hidden state,  $h_t$ , that contains information about previous inputs and current input are passed to the next time step of the sequence.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t \times \tanh(C_t) \quad (6)$$

### 3.2. GRU

Another model that is inspired by RNN and aims to solve the short-term memory problem of RNN is the Gated Recurrent Unit (GRU) model introduced in [39]. GRU is considered a part of the new generation of Recurrent Neural Networks and it functions in a similar way to LSTM. The major architectural difference between LSTM and GRU is that GRU only contains two gates known as reset gate and update gate as shown in Figure 7. It combines the input gate and the forget gate of LSTM into a single update gate, and it further merges the hidden state and the cell state into a single hidden state. A gated recurrent unit cell is characterized by a simpler internal structure and operations than a long short-term memory cell as shown in Figure 8. The mathematical notations behind these operations are introduced and described in Table 2. The difference in design complexity between the two models makes GRU faster than LSTM [39]. Like LSTM, GRU works on the principle of storing the relevant data in a given time step and propagating it throughout the entire sequence through a hidden state. The hidden state in GRU acts as a memory to the network that possesses the important information of previous time steps and current time-step. The unique internal structure and mechanism of GRU make it achieve its purpose of tackling the short-term memory problem in a different way than LSTM. The gates control which hidden state information should be updated and passed to the next time step or reset the hidden state when information should be forgotten and neglected.

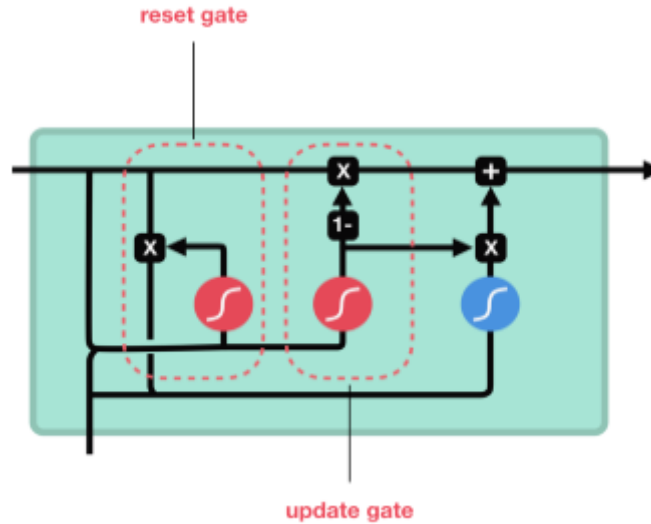
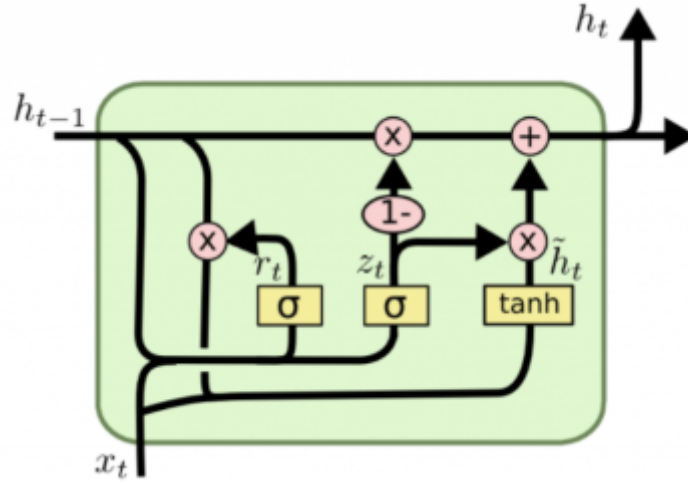


Figure 7. The architecture of a gated recurrent unit (GRU) cell-gates [41]



**Figure 8.** The architecture of a gated recurrent unit (GRU) cell-operations [42]

Notation	Description
$\sigma$	Sigmoid Activation Function
$\times$	Pointwise Multiplication
$+$	Pointwise Addition
$\tanh$	tanh Activation Function
$x_t$	Current Input
$h_t$	New Hidden State
$h_{t-1}$	Previous Hidden State
$r_t$	Reset Gate Output
$z_t$	Update Gate Output
$\tilde{h}_t$	Candidate Cell State

**Table 2.** Description of the notations used in the operations of the internal architecture of GRU

The update gate is responsible for deciding the volume of information to keep and the volume of information to forget from prior time steps and current time-step. The update gate output,  $z_t$ , is determined as indicated by Equation 7. The previous hidden state,  $h_{t-1}$ , and the current input,  $x_t$ , are passed into the Sigmoid activation function,  $\sigma$ . The  $W_z$  and  $b_z$  in Equation 7 represent the weight matrix and bias vector of the update gate. The result of the Sigmoid function,  $\sigma$ , assists in determining the relevant information to keep from both the current and previous inputs. This information is stored in the form of the output of the update gate,  $z_t$ .

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \quad (7)$$

The reset gate figures the amount of past information that needs to be neglected going forward in the time-series sequence. The output of the reset gate,  $r_t$ , is computed by passing the current input,  $x_t$ , and the previous hidden state,  $h_{t-1}$ , into a Sigmoid function,  $\sigma$ . These steps are demonstrated by Equation 8 which is very similar to Equation 7, however, the Sigmoid function,  $\sigma$ , purpose in the reset gate is to highlight the information to forget from previous time steps. The output of the reset gate,  $r_t$ , is used to form the current memory function,  $\tilde{h}_t$ , that acts as a candidate hidden state. Equation 9 shows how the current memory function,  $\tilde{h}_t$ , is calculated. The output of the reset gate,  $r_t$ , that is computed through the Sigmoid function,  $\sigma$ , is multiplied with the previous hidden state,  $h_{t-1}$ . This multiplication results in neglecting the irrelevant information from previous inputs in the current memory function,  $\tilde{h}_t$ . The outcome of the multiplication and the current input,  $x_t$ , are passed into a tanh function to regulate the values of the current memory function,  $\tilde{h}_t$ . The output of

Equation 9 ensures the ability of current memory function,  $\tilde{h}_t$ , in maintaining the important information from previous inputs and information from the current input.

$$r_t = \sigma(W_r.[h_{t-1}, x_t] + b_r) \quad (8)$$

$$\tilde{h}_t = \tanh(W_h.[r_t \times h_{t-1}, x_t] + b_h) \quad (9)$$

Equation 10 clarifies the process of calculating the new hidden state,  $h_t$ , used to transmit relevant information and disregard irrelevant information throughout the sequence of time steps. The output of the update gate,  $z_t$ , has a vital role in determining the new hidden state,  $h_t$ , as it decides the important information from both the prior time steps and the current time-step. The current memory function,  $\tilde{h}_t$ , that holds relevant information from previous inputs is pointwise multiplied with the update gate output,  $z_t$ . This operation outputs values that contain all the relevant information from both prior inputs as well as the current input. The result of this multiplication is added to the multiplication of the previous hidden state,  $h_{t-1}$ , with the inverse of the update gate output denoted by  $1-z_t$ . The inverse represents the irrelevant information from the previous time steps in the sequence including the current one. This makes the multiplication between the inverse of the output of the update gate,  $1-z_t$ , and the previous hidden state,  $h_{t-1}$ , a portrayal of the data that needs to be forgotten from the sequence. As a result of Equation 10 the new hidden state will be updated with all the relevant information from prior time steps and current time step, and it will also be reset by forgetting the unimportant data from all the time steps traversed so far.

$$h_t = (1 - z_t) \times h_{t-1} + z_t \times \tilde{h}_t \quad (10)$$

Bitcoin price prediction is a time-series problem where the price at a given time-step depends on previous time-steps. Although RNN is designed specifically to tackle sequence-related problems, it yields inaccurate results when used for Bitcoin price prediction. The price of Bitcoin at a given time-step is determined by information from numerous previous time-steps that create a long sequence of information to be remembered and carried on to the next time-step. This explains the inadequacy of RNN in the Bitcoin price prediction problem as it suffers from a short-term memory problem that creates hardships in dealing with long-sequences in a time-series problem such as the problem present here. LSTM and GRU are an enhancement of RNN that solves the short-term memory problem thus making the two models an effective and promising solution for the Bitcoin price prediction problem.

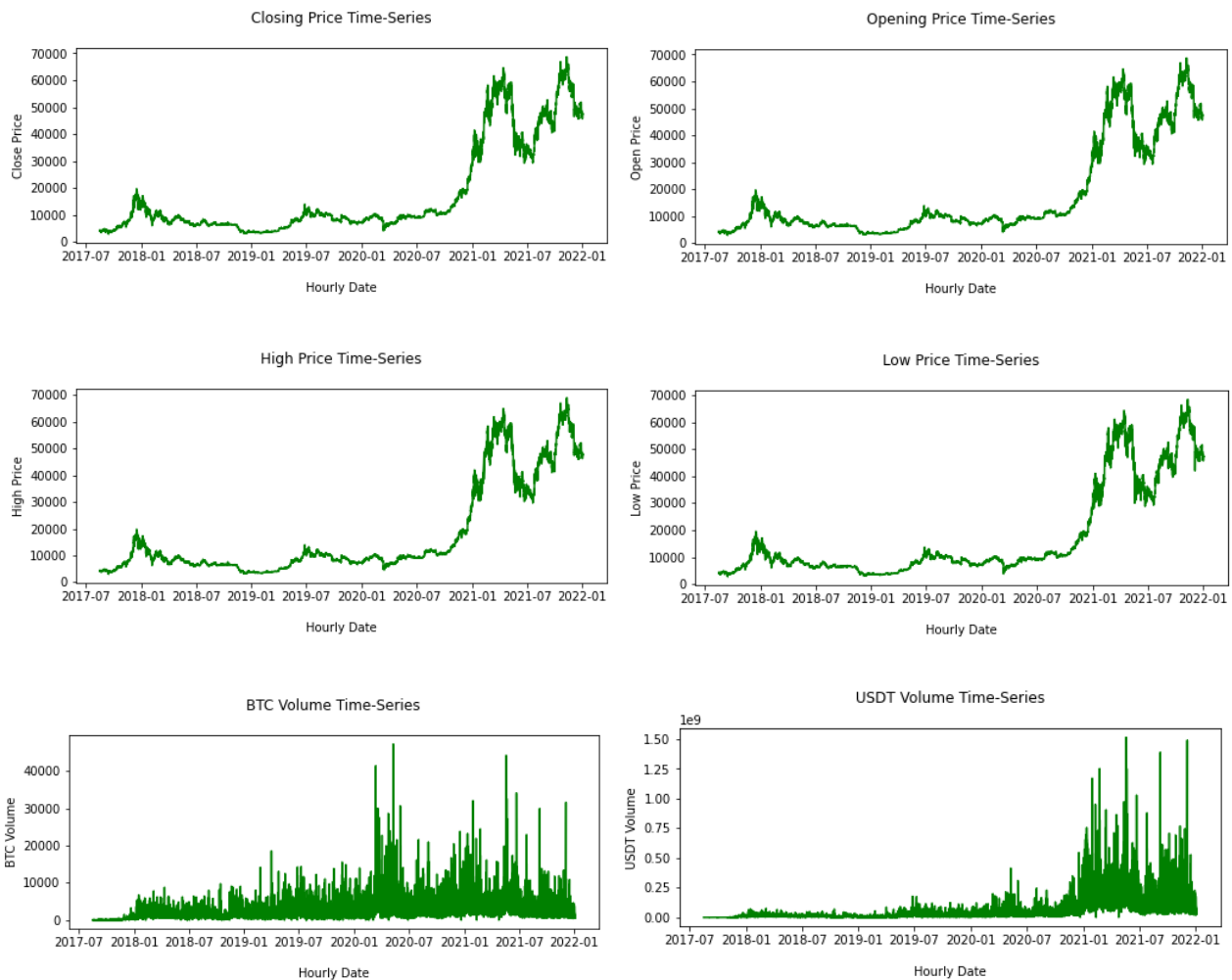
#### 4. Data Collection

Bitcoin data for the current study was collected from Binance, a cryptocurrency exchange platform, accessible on <https://www.cryptodatadownload.com/data/binance/>. The nature of the data is a time-series data where the time separating two consecutive time-steps is one hour meaning the data gathered is partitioned by 1-hour intervals. The Bitcoin accumulated data is in USD and it contains six features. The key features focused on in this study are opening price, highest price, lowest price, Bitcoin (BTC) volume, Tether USD (USDT) volume, and the closing price which is the feature to be predicted. The connotation of each of these features is illustrated in Table 3. The OHLC (open, high, low, and close) features in addition to the BTC volume and the USDT volume are crucial endogenous determinants of the Bitcoin price [8,9,11,13,16]. The historical hourly Bitcoin data was collected for the period of August 17, 2017 starting at 4 am to January 2, 2022 ending at 12 am. The Bitcoin time-series established contains 38,351 time-steps denoting 38,351 hours which are equivalent to 4.37 years worth of Bitcoin data. Each time-step includes the opening price at a specific hour of a particular date (e.g. 3 pm of October 27, 2019), the lowest and highest prices of Bitcoin during that hour (from 3 pm until 4 pm of the exact date), the BTC volume and its equivalent USDT volume during that hour, and the closing price of the hour which is the opening price of the next hour (price of Bitcoin at 4 pm of October 27, 2019). The time-series for the Bitcoin prices (open, high, low, and close prices), BTC volume, and USDT volume collected

are shown in Figure 9. The data exhibited in Figure 9 was visualized through the usage of Python’s Matplotlib library.

Features	Description
Open	The price of Bitcoin at the opening of a period of time.
High	The highest price of Bitcoin during a period of time.
Low	The lowest price of Bitcoin during a period of time.
Close	The price of Bitcoin at the closing of a period of time.
BTC Volume	The amount of Bitcoins being exchanged (bought and sold) during a period of time.
USDT Volume	The equivalent amount of Bitcoins being exchanged in USD during a period of time.

**Table 3.** Description of the features included in the data set with a period of time of one hour



**Figure 9.** Plots of the historical Bitcoin time-series data collected for each of the six features used in this study

#### 4.1. Data Pre-Processing

We converted the 4-years data gathered of Bitcoin into a 4-years data set after we cleaned the data using regular expression (regex), Java, and Python’s pandas library. The 4-years data set is a time-series data set with each time-step holding Bitcoin-dependent features that are listed in Table 3. Each time-step in the time-series data set is uniquely identified through a combination of a specific date (e.g. 4/28/2022) and a specific hour in that date (e.g. 14:00) as the 4-years data set has 1-hour partitioning between time steps. In this study, the aim is to

determine the better model for long-term investments between LSTM and GRU by comparing the performance of the two models on data sets of different sizes in term of the number of time-steps in the time-series and the different time-series intervals (e.g. comparing a 3-years data set with a 5-years data set). To perform a comparison between LSTM and GRU that fits the goal of this work, a 2-years data set and a 1-year data set were derived from the 4-years data set.

The 2-years data set carries hourly historical Bitcoin data of the same features present in the 4-years data set (OHLC features, BTC volume, and USDT volume). This derived data set spans a period of 2.2 years starting from August 17, 2017 at 4 am and ending on November 28, 2019 at 4 pm. A total of 20,000 hours (20,000 time-steps) constitute the 2-years time-series data set. The period of the 2-years data set (from August 17, 2017 at 4 am until November 28, 2019 at 4 pm) is a subset of the period of the 4-years data set (from August 17, 2017 at 4 am until January 2, 2022 at 12 am).

An hourly data set spanning the period of 1.1 years is also derived from the 4-years data set. This 1-year data set carries the same features present in the 4-years data set and has a total of 9,999 time-steps representing 9,999 hours. The 1-year data set includes a time interval from August 17, 2017 at 4 am until October 7, 2018 at 11 pm that represents a subset of the 2-years data set interval (from August 17, 2017 at 4 am until November 28, 2019 at 4 pm).

The data sets of 1-year, 2-years, and 4-years have a time-series sequence length of 9,999 hours, 20,000 hours, and 38,351 hours respectively. This enables the comparison between LSTM and GRU based upon data sets of varying and increasing volumes. The comparison will also incorporate different and increasing time intervals of 1-year, 2-years, and 4-years. The presence of increasing time intervals is the factor resulting in the increase of size for the data sets with a constant 1-hour partitioning interval. The expansion of the data set size through increasing the time interval permits the addition of new data patterns to the time-series sequence by the addition of more time periods (more hours) to the time-series. This simulates the real behavior of Bitcoin historical data as the time-series of Bitcoin keeps on increasing in size by adding more time periods to the historical sequence (e.g. more seconds, minutes, hours, or days).

## 5. Experiments

Three separate experiments were conducted to compare the performance of LSTM and GRU in Bitcoin price prediction. The first experiment uses the 1-year data set, the second experiment relies on the 2-years data set, and the third experiment utilizes the 4-years data set. In each experiment, GRU and LSTM models were built, evaluated, and compared to determine the difference in the performance of the two models on a particular data set. The results of each experiment are the basis of the comparison of the scalability of the performance of LSTM and GRU. The results of the experiments will reflect the evolution of the performance of LSTM and GRU as the data set size and time-interval increases from the first experiment (1-year data set) to the second experiment (2-years data set) to the third and final experiment (4-years data set). The same evaluation metrics for performance and the same data pre-processing techniques were used in the three experiments. The LSTM and GRU models implemented had the exact same configuration in terms of hyperparameters tuning in the same experiment and different configurations between different experiments. The goal was to optimize the performance of both LSTM and GRU in each experiment with the usage of the optimal hyperparameters that have to be tailored differently in the three experiments due to the usage of data sets of variant time-intervals and time-series sequences length.

### 5.1. Data Splitting

The data pre-processing step in all experiments was conducted using Python. In each experiment, the data set was imported using the pandas library of Python. The date column in each data set is placed as the index column that acts as an identifier for each time-step in the time-series. The data sets in the first and the second experiments, 1-year data set and 2-years data set respectively, are divided into 70% training set, 20% validation set, and 10% testing set. In the third experiment, the 4-years data set is divided into 85% training set, 5% validation set, and 10% testing set. The choice of the percentages of the training and validation set in each experiment is based on optimizing the performance of the GRU and LSTM models. The details of splitting each data set into training, validation, and testing sets are found in Table 4. The features of each of the training, validation, and testing sets in each of the experiments are separated into five input features and one output feature. The input features are opening price, high price, low price, BTC volume, and USDT volume while the output feature is the closing price. Both LSTM and GRU use the input features as predictors that determine the



output feature, the closing price, which is the value to be predicted. According to Figure 9, the range of the values of each of the Bitcoin endogenous features is unlike. Features with values of high range, like USDT volume, will influence the results of the model more than features of low range, such as BTC volume, which the models will regard as unimportant. To ensure that each feature impacts the model in an equal and balanced manner, Data normalization is needed. The normalization method used is MinMaxScaler that scales the values' range of each feature to a range between 0 and 1 without changing the shape of the distribution of the data. Equation 11 shows the process of scaling each value in a feature to normalize the entire time-series of the feature.

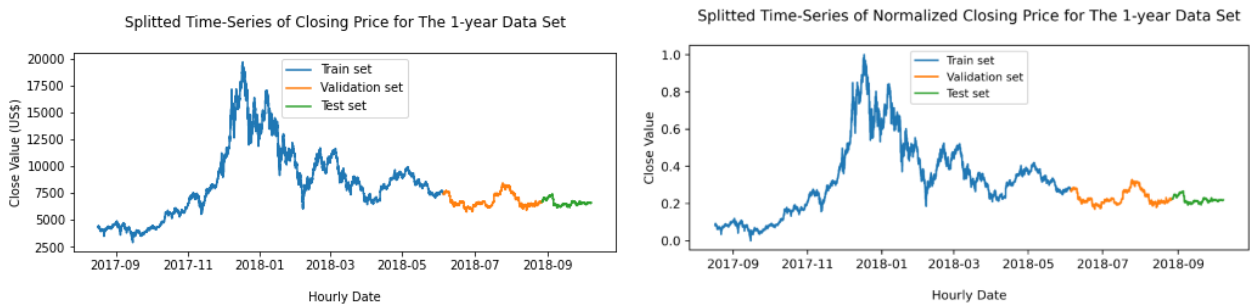
$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (11)$$

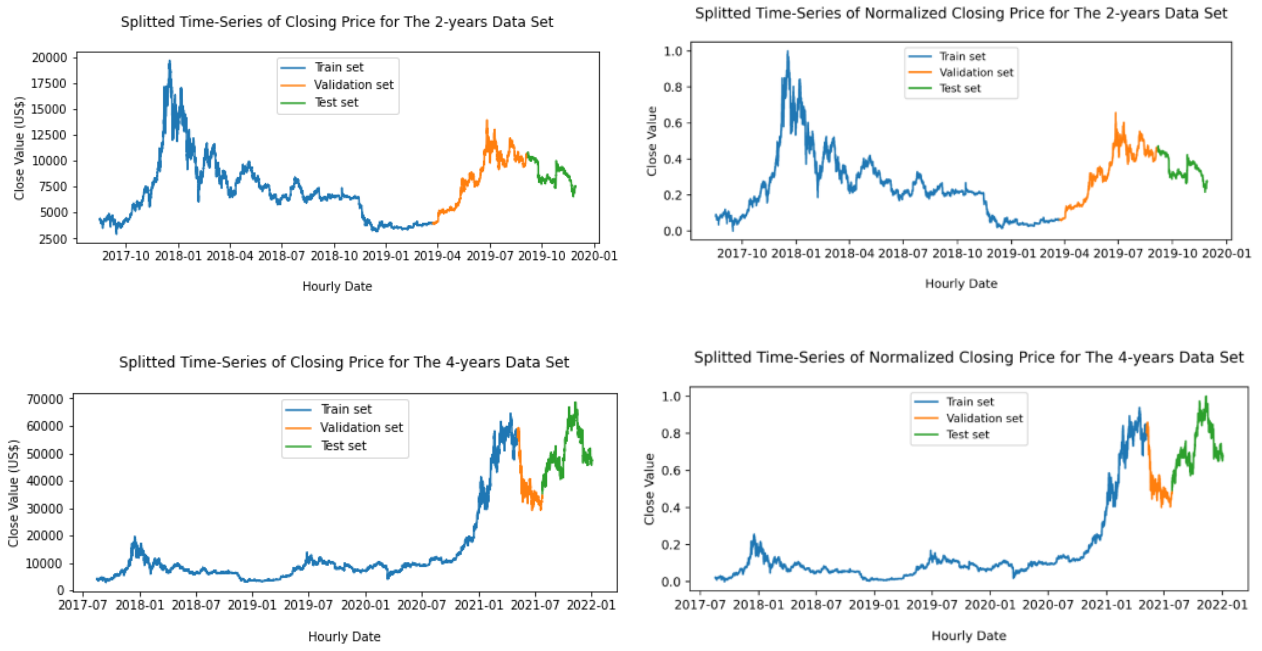
To scale a value in a feature, the value is subtracted with the minimum value in a feature's time-series. The result of the subtraction is then divided by the result of subtracting the maximum value of a feature's time-series with its minimum value which signifies the range of the values of the feature. The output of the division is the new scaled value that is guaranteed to be between 0 and 1.

All the features of the training, validation, and testing sets in each experiment were normalized through MinMaxScaler that was implemented using Python's scikit-learn library. The closing price splitted over training, validation, and testing in each experiment along its equivalent normalized and also splitted closing price are shown in Figure 10. In Figure 10, it is observable that regardless of scaling the range of the closing price feature between 0 and 1, the shape of the distribution of the time-series is still the same. The last step in the data pre-processing phase was restructuring the dimensions of the input features and output feature for each of the training, validation, and testing sets in every experiment to a form that fits the deep learning algorithms used: LSTM and GRU. This step was completed using the sliding window technique which uses a variable number of previous time-steps, known as the lookback period, of input features to predict the output feature of the next time-step.

Data Set	Total Size	Training Size	Validation Size	Testing Size
1-year	9,999	6,999	2,000	1,000
2-years	20,000	14,000	4,000	2,000
4-years	38,351	32,598	1917	3,836

**Table 4.** The size (in time-steps) of the used data sets and their respective splitted training, validation, and testing data sets



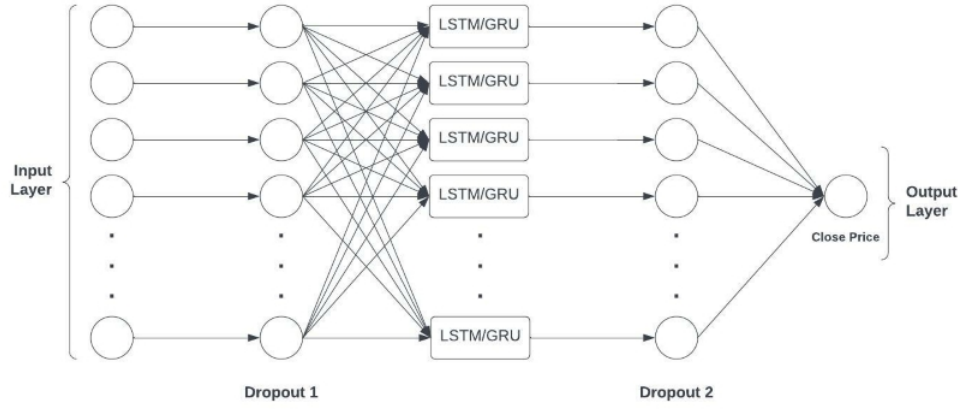


**Figure 10.** Plots of the splitted closing price of each data set with its equivalent normalized closing price

### 5.2. Models' Implementation

The LSTM and GRU models are implemented in each of the three experiments using the Keras library of Python. The same layered structure for LSTM and GRU is used throughout all experiments. This layered structure is illustrated in Figure 11. As shown in the figure, the models consist of an input layer, one hidden layer, and an output dense layer. Each of the input and hidden layers are made of 72 neuron, and the output layer has one neuron that denote the single output feature, the closing price. Dropout was incorporated into both input and hidden layers to avoid overfitting [2] in addition to the Sigmoid activation function to maintain the output of layers between the range of 0 and 1, the scaled range of the data sets' features. The data fed to the models for training had  $5 \times \alpha$  dimensions where 5 is the number of the input features and  $\alpha$  is value of the lookback period determined by the sliding window. The compilation of the models used the Mean Squared Error (MSE) as a loss function to calculate the error of the models in the training and validation phases. The loss function assists in the optimization process of models by changing the weights of the neural network thus driving the neural network to learn. The models were also trained using the Adam optimizer [2] and early stopping to effectively reduce the time it takes to train a given model in case it exhibits signs of overfitting. Although the implemented models have similar layered structure and design decisions such as the choice of activation and loss functions, models in each experiment were configured through a different set of hyperparameters that require subjective values. The hyperparameters employed in LSTM and GRU models in this study are learning rate, number of hidden layers, number of neurons in each layer, batch size, number of epochs, activation function, ratio of dropout, loss function, weight optimizer, and lookback period. A combination of the choice of values of the used hyperparameters influences the performance of the models directly. Hyperparameters tuning was required to ensure the optimization of LSTM and GRU in each experiment. The two models, LSTM and GRU, in the same experiment have the same hyperparameters as tuning the hyperparameters values to optimize one model was also optimizing the other model due to their similar architectural designs [28]. The tuning of the hyperparameters differed between different experiments as the data size and time-interval varied. The utilization of different values of hyperparameters between different experiments was essential to explore the full potential of all the implemented LSTM and GRU models. The process of hyperparameters tuning was established by trial and error in each of the experiments. The distinct values of hyperparameters in each experiment were figured through multiple attempts that uncovered the correlation between the behavior of the performance of the models and the directed changes in each hyperparameter used. The result was three distinct sets of hyperparameters, one for each experiment, where each set optimizes the performance of both LSTM and GRU in a given experiment. The complete list of the configured hyperparameters and their respective values

for the first experiment (1-year data set), the second experiment (2-years data set), and the third experiment (4-years data set) are shown respectively in Table 5, Table 6, and Table 7.



**Figure 11.** The neural network layered structure of the implemented LSTM and GRU models in all experiments

Hyperparameter	Value
Learning Rate	0.001
Hidden Layers	1
Neurons	72
Batch Size	128
Epochs	100
Activation Function	Sigmoid
Dropout Rate	0.2
Loss Function	Mean Squared Error
Weight Optimizer	Adam
Lookback Period	2

**Table 5.** The values of the tuned hyperparameters of the LSTM and GRU models in the first experiment (1-year data set)

Hyperparameter	Value
Learning Rate	0.0001
Hidden Layers	1
Neurons	72
Batch Size	256
Epochs	600
Activation Function	Sigmoid
Dropout Rate	0.2
Loss Function	Mean Squared Error
Weight Optimizer	Adam
Lookback Period	2

**Table 6.** The values of the tuned hyperparameters of the LSTM and GRU models in the second experiment (2-years data set)

Hyperparameter	Value
Learning Rate	0.0001
Hidden Layers	1
Neurons	72
Batch Size	256
Epochs	300
Activation Function	Sigmoid
Dropout Rate	0.1
Loss Function	Mean Squared Error
Weight Optimizer	Adam
Lookback Period	1

**Table 7.** The values of the tuned hyperparameters of the LSTM and GRU models in the third experiment (4-years data set)

### 5.3. Performance Metrics

To evaluate the performance of each LSTM and GRU model implemented in the three experiments, two metrics were used: Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE).

RMSE is the square root of the Mean Squared Error (MSE). MSE is the mean value of the squared error between each predicted value and its corresponding actual value in a set of predicted values. The range of RMSE is from 0 to infinity, and lower RMSE value signifies better performance. The mathematical formula for RMSE is shown in Equation 12.

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{n}} \quad (12)$$

The value of RMSE is calculated by first subtracting the predicted value,  $\hat{y}_i$ , and the actual value,  $y_i$ . The result of the subtraction represents the error in the prediction value. The error is then squared thus giving the squared error value of the gap between the actual value and its respective predicted value. This grants RMSE the property of punishing large errors. Large errors will have more weight in determining the value of RMSE. The higher the RMSE value the more frequency of large errors is present and vice versa. The process of calculating the squared error is repeated  $n$  times, where  $n$  is the total number of predicted values, for each predicted value. The squared error of each predicted value is added together. The outcome of the summation is then divided by the number of predicted values,  $n$ , thus producing the mean squared error of all predicted values. To output the RMSE, the square root of the mean squared error is calculated. The unit of the calculated RMSE is the same as the predicted value. The unit of RMSE in this study is normalized USD as the closing price feature is the predicted feature and its values are normalized. RMSE does not take into consideration the direction of the error since the difference between the predicted value,  $\hat{y}_i$ , and the actual value,  $y_i$ , whether being positive or negative is squared which always lead to a positive value.

MAPE is the average percentage of error in a set of predicted values. MAPE has a range between 0 and infinity, and lower MAPE value is a sign of better performance as the average percentage of error decreases. MAPE is expressed mathematically in Equation 13.

$$MAPE = \frac{100\%}{n} \times \sum_{i=1}^n \left| \frac{(\hat{y}_i - y_i)}{y_i} \right| \quad (13)$$

To compute MAPE, the actual value,  $y_i$ , is subtracted from the predicted value,  $\hat{y}_i$ . The outcome of the subtraction represents the error of prediction and is divided by the actual value,  $y_i$ , to result in the fraction of the prediction error from the actual value. The fraction is then multiplied by 100 to give the percentage of the prediction error with respect to the actual value,  $y_i$ . Absolute value operation is applied to the percentage of prediction error which disregards the direction of the error by always computing a positive percentage error

for a prediction. This procedure is repeated as many times as the size of predicted values set which is  $n$ . The repeated calculations are summed together thus producing the total percentage error of the predicted values. The total percentage error is then divided by  $n$ , the size of the predicted values set, to produce the average percentage error (MAPE) of all predicted values. MAPE does not depend on any units of data since it is a percentage.

In this study, a predicted value,  $\hat{y}_i$ , is the closing price predicted by either LSTM or GRU in a given experiment and its corresponding actual value,  $y_i$ , is the actual closing price of a time-step in one of the three testing data sets used for comparison with the predicted value, while the total number of predicted values,  $n$ , is the size of the testing set in one of the three experiments after being reshaped by the sliding window.

## 6. Results

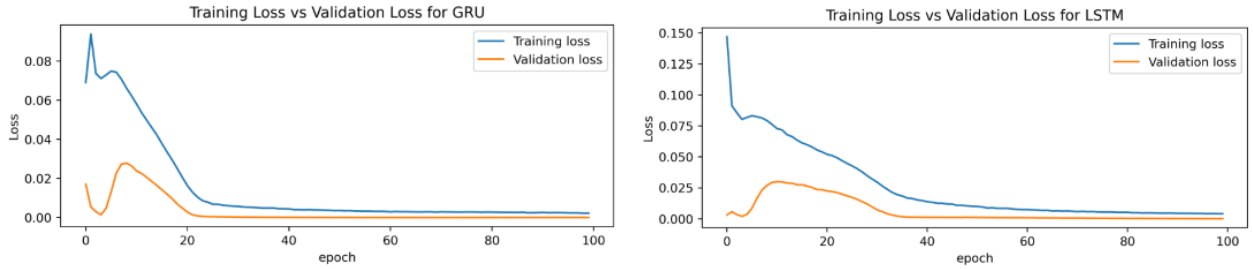
We implemented LSTM and GRU models in three separate experiments. Each experiment is characterized by a different set of hyperparameters tailored to optimize the performance of LSTM and GRU along a data set of exclusive size and time-interval. In the first experiment, we tuned the hyperparameters of LSTM and GRU according to the values shown in Table 5 and fed the 1-year data set to these models. In the second experiment, Table 6 exhibits the employed hyperparameters for both LSTM and GRU that performed price predictions using the 2-years data set. The 4-years data set and the hyperparameters of Table 7 were utilized in the third experiment. The training of both LSTM and GRU in each experiment used MSE as loss function to monitor the performance of the models over the splitted training data set of the corresponding experiment. A low MSE value indicates that the trained model performs well over the training data set. Although MSE is a reliable indicator of the performance of the trained model on the training data, it is incapable of showcasing the ability of the model to generalize its performance on new and unseen data. The phenomenon of the inability of the trained model to generalize its performance over unseen data is known as overfitting. To tackle the problem of overfitting, we also used a validation data set that resembles new and unseen data in the training of a given model. The performance of a given model over the validation data set is measured using MSE as well. If the MSE value of the validation data set is close to the MSE value of the training data set, then the trained model has the ability to generalize its performance. Otherwise, the trained model displays signs of overfitting. To further assess the credibility of performance of the utilized LSTM and GRU models in each experiment, we employed the RMSE and MAPE metrics mentioned in Section 5.3 to compute the performance of the trained models over the testing data set that belongs to its respective experiment. The used metrics in the training, validation, and testing phases reflect the ability of LSTM and GRU in predicting the hourly closing price of Bitcoin using opening price, high price, low price, BTC volume, and USDT volume as endogenous input features. We used Python's Keras library to compute MSE for the training of the models over the training and validation data sets of a given experiment. To compute RMSE and MAPE of LSTM and GRU on the testing data set of each experiment, scikit library of Python was utilized.

### 6.1. First Experiment

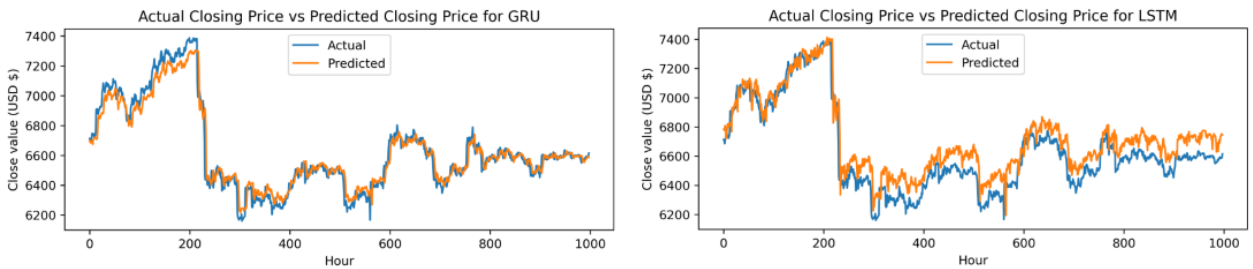
Figure 12 displays the development of the MSE loss of the training data set and the validation data set while training LSTM and GRU in the first experiment that utilizes the 1-year data set. The MSE loss after the GRU model finishes training is 0.0024 for the training data set and 0.000044 for the validation data set. The GRU model trains efficiently as the MSE for the training data set is low and close to the MSE of the training data set. The trained GRU model does not exhibit signs of overfitting since the MSE of the validation data set is lower than the MSE of the training data set. The trained LSTM has MSE of 0.0031 and 0.000061 for the training data set and the validation data set respectively after the model finishes training. LSTM is capable of generalizing its good performance on new and unseen data as the values of MSE of training and validation data sets are both low and close to each other as shown in Figure 12. The increase in the learning rate, lookback period, and the number of hidden layers from the current used values negatively affected the training of both LSTM and GRU models. The models reported signs of inefficient training phase carried with overfitting as the MSE for both training and validation data sets increased as well as the difference between the MSE of the two data sets with the MSE of the validation data set being higher. The same behavior was witnessed when we tried to decrease the dropout rate and the batch size from their current values. Figure 13 compares the capability of the trained LSTM and GRU models in predicting the closing price for a series of time-steps with the actual closing price of a series of time-steps using the testing data set of the 1-year data set. As observed in Figure 13, both LSTM and GRU produce a time-series of the predicted closing price that has a similar general pattern of price



movements (up and down) with the time-series of the actual closing price. The plots of the actual price and the predicted price are close to each other for both models. The metrics that describe the ability of LSTM and GRU in predicting the closing price of Bitcoin using the 1-year data set, illustrated in Figure 13, are shown in Table 8. GRU has a MAPE of 0.011 (1.1%) and RMSE of 0.0038 while LSTM has a MAPE of 0.027 (2.7%) and RMSE of 0.0069. These measurements reveal that GRU produces lower prediction error average (lower MAPE) and less frequent extreme prediction errors (less RMSE) than LSTM, therefore, GRU outperforms LSTM using the 1-year data set in the first experiment.



**Figure 12.** Plots of the training loss and validation loss for each of LSTM and GRU in the first experiment



**Figure 13.** Plots of the predicted closing price by LSTM and GRU versus the actual closing price in the first experiment

Model	RMSE	MAPE
GRU	0.0038	0.011
LSTM	0.0069	0.027

**Table 8.** The performance metrics (RMSE and MAPE) of LSTM and GRU in the first experiment (1-year data set)

## 6.2. Second Experiment

The plots of the MSE of the training data set and the validation data set during training LSTM and GRU based on the 2-years data set are shown in Figure 14. After training the GRU model, the training data set had a MSE of 0.00053 while the validation data set had MSE of 0.00033. The MSE of the training data set and validation data set are low and close to each other thus signifying the ability of the trained GRU model in performing well when present with unseen data. For the implemented LSTM model, the training phase ended with a 0.00061 MSE for the training data set and 0.00027 MSE for the validation data set. The problem of overfitting is also avoided in the trained LSTM model. Although the MSE of the validation data set displays an oscillation pattern in Figure 14 for both LSTM and GRU, the difference between the MSE of the training and validation data sets remains minimal which is an indication that the models were trained with no signs of overfitting. LSTM and GRU models in this experiment displayed a similar behavior to the LSTM and GRU models employed in the first experiment when changes occurred in various hyperparameters. The increase in learning rate, number of hidden layers, and the lookback period led to overfitting in training the models as the MSE of the validation

data set drastically increased surpassing the MSE of the training data set by a significant gap. The decrease in the batch size, epochs, and dropout rate produced the same outcome. The testing data set of the 2-years data set was used to assess the trained LSTM and GRU models in predicting the closing price through RMSE and MAPE. The plot lines of the actual closing price and the closing price predicted by LSTM and GRU is visualised in Figure 15. The forecasted closing price time-series by LSTM and GRU is close to the actual time-series of the closing price and both follow the same price fluctuations trend. The performance metrics underlying the visualized forecasting of the trained models are reported in Table 9. The GRU model yields a MAPE value of 0.023 (2.3%) while the LSTM model yields a MAPE value of 0.018 (1.8%). The trained LSTM model possesses a lower and better average prediction error. LSTM also has 0.0086 RMSE which is lower and better than GRU's 0.01 RMSE thus producing less frequent large errors. LSTM has a better performance than GRU using the 2-years data set as it has both lower RMSE and MAPE.

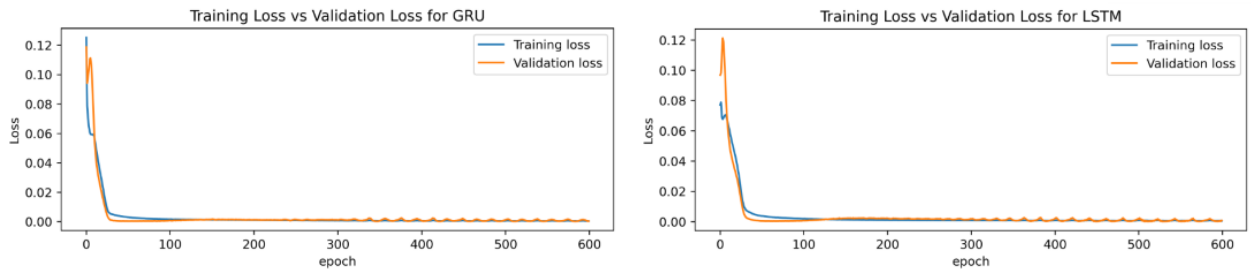


Figure 14. Plots of the training loss and validation loss for each of LSTM and GRU in the second experiment

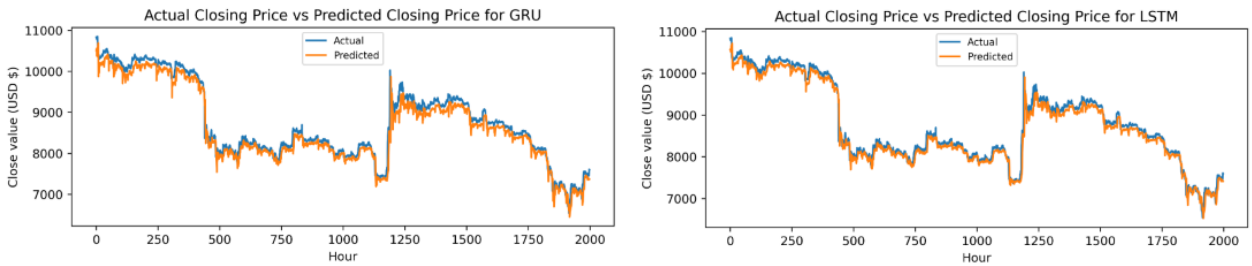


Figure 15. Plots of the predicted closing price by LSTM and GRU versus the actual closing price in the second experiment

Model	RMSE	MAPE
GRU	0.01	0.023
LSTM	0.0086	0.018

Table 9. The performance metrics (RMSE and MAPE) of LSTM and GRU in the second experiment (2-years data set)

### 6.3. Third Experiment

The relation between the MSE of LSTM and GRU over the training and validation data sets in the third experiment that utilizes the 4-years data set is shown in Figure 16. Although the oscillations in the MSE of the validation data set for both LSTM and GRU present in Figure 16 are more severe than the oscillations present in Figure 14 of the second experiment, the MSE of the validation data set keeps on decreasing with values close to the MSE of the training data set. The trained models are still able to generalize their performance in an efficient manner. After training, the MSE of the GRU and LSTM models over the training data set are 0.00032 and 0.00059 respectively. The MSE of the LSTM model over the validation data set is 0.0028 and that of GRU is 0.00019. An increase in learning rate, number of hidden layers, and lookback period in addition to a decrease in batch size, epochs, and dropout rate resulted in a bad training performance accompanied with overfitting

behavior signaled by high values of MSE for the training and validation data sets in both models. In Figure 17, LSTM and GRU predict the closing price of Bitcoin using the testing time-series of the 4-years data set. The results are compared with the actual closing price to illustrate the quality of performance of both models in Bitcoin price prediction. Like the first and second experiments, the predicted closing price given by both LSTM and GRU follows the general price changes pattern for the real closing price. According to Table 10, GRU has 0.014 (1.4%) MAPE and 0.014 RMSE while LSTM has 0.016 (1.6%) MAPE and 0.021 RMSE. GRU outperforms LSTM in the third experiment that uses the 4-years data set as it has lower average prediction error (lower MAPE) and less frequent extreme errors when compared to LSTM (lower RMSE).

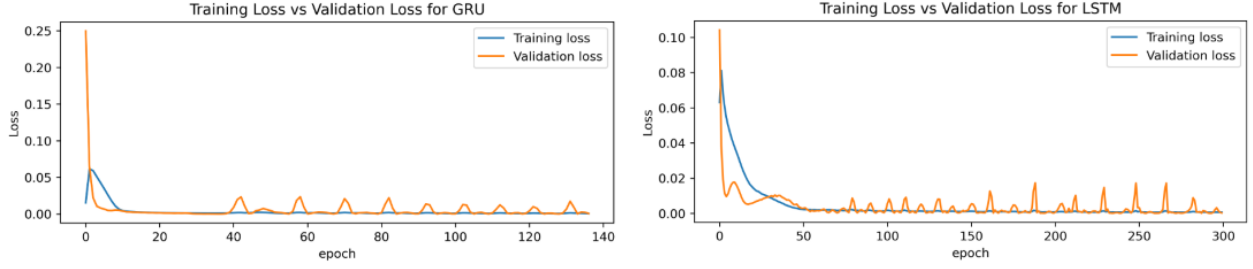


Figure 16. Plots of the training loss and validation loss for each of LSTM and GRU in the third experiment

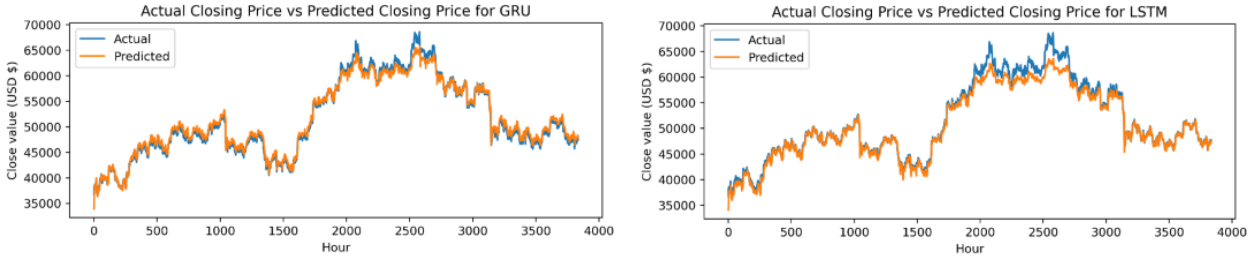


Figure 17. Plots of the predicted closing price by LSTM and GRU versus the actual closing price in the third experiment

Model	RMSE	MAPE
GRU	0.014	0.014
LSTM	0.021	0.016

Table 10. The performance metrics (RMSE and MAPE) of LSTM and GRU in the third experiment (4-years data set)

## 7. Discussion

Based on RMSE and MAPE, GRU outperformed LSTM in the first and third experiments that utilize the 1-year and 4-years data sets respectively. Meanwhile, in the second experiment, LSTM performed better than GRU in closing price prediction according to both MAPE and RMSE using the 2-years data set. To compare between LSTM and GRU, it is not sufficient to rely on the results of the three separate experiments. The better model in Bitcoin price prediction is not the model with better MAPE and RMSE in a given experiment, but the model that has a better scalability in MAPE and RMSE as the size of the data set and the interval of the time-series increase. The historical Bitcoin time-series increases in size over time by increasing its time-interval. For example, if the time-interval of Bitcoin time-series is 3-years currently, it will be 4-years a year after. If a model has an excellent performance in price prediction over a fixed time-interval, it does not mean that this performance will be stable as the time-interval increases. To properly compare between LSTM and GRU, we examine the correlation of their performance in the results of the three experiments. The model whose

performance scales better as the time-interval of the data set increases from 1-year to 2-years to 4-years is the model that better predicts the behavior of the Bitcoin time-series.

Table 11 showcases the development of the MAPE performance metric for both LSTM and GRU as the time-interval of the data set increases from 1-year to 2-year to 4-years. The paired t-test, a statistical test, was conducted to infer if the differences in the MAPE values for each of LSTM and GRU as the time-interval increases from 1-year to 2-years to 4-years are statistically significant. The reported  $p$ -values in Table 11 indicate that there exists statistical significance behind the development of MAPE of LSTM and GRU over a data set of increasing time-interval. The MAPE of the GRU model in the 1-year data set is 0.011 (1.1%). This value undergoes a 109% increase to reach 0.023 (2.3%) in the second experiment. This signifies that the MAPE of GRU scaled negatively and the performance decreased as the time-interval of the data set increased from 1-year to 2-years. As the time-interval continues to increase from 2-years to 4-years, the MAPE of GRU scales positively and decreases by 39.1% to reach 0.014 (1.4%). Although the MAPE of GRU gets better with the increase of the time-interval from 2-years to 4-years, the scalability of GRU performance measured with MAPE is not stable. The MAPE of GRU scales negatively (decrease in performance) between the 1-year and 2-years time-interval, scales positively between the 2-years and 4-years time-interval (increase in performance), and scales negatively overall with the value increasing from 0.011 (1.1%) to 0.014 (1.4%) with a 27.2% value-increase as the time-interval of the data set grows from 1-year to 4-years. The LSTM model exhibits more stable and scalable MAPE measurements as the time-interval of the data set increases. LSTM's MAPE decreases from 0.027 (2.7%) to reach 0.018 (1.8%), a 33.3% value-decrease, with the increase of the time-interval from 1-year to 2-years. This lower MAPE value denotes that the performance of LSTM got better, and it continues to scale positively as the time interval increases from 2-years to 4-years. The MAPE of LSTM decreased by 11.1% to reach 0.016 (1.6%) utilizing a 4-years time-interval data set. The performance of LSTM measured by MAPE is scalable since it gets better with the increase of the time-interval from 1-year to 2-years to 4-years. The MAPE of GRU is better than the MAPE of LSTM in the first and third experiments, however, LSTM is the model that has better performance based on MAPE for price prediction. The MAPE of LSTM is scalable with the increase of Bitcoin time-intervals as the average error for predictions continues to decrease with the addition of new time-steps to the existing time-series. The same cannot be said about GRU based on the three experiments we conducted. The evolution of GRU's MAPE shown in Table 11 indicates that the scalability of the MAPE with the increase of the time-interval is not stable as the average error of predictions fluctuates between the three experiments. This supports the usage of LSTM over GRU as a Bitcoin price prediction model that is capable of maintaining and improving the average errors in predictions on a long term.

Model	1-year	2-years	4-years	$p$ -Value
GRU	0.011	0.023	0.014	0.045
LSTM	0.027	0.018	0.016	0.045

**Table 11.** The evolution of MAPE of the implemented LSTM and GRU models as the time-interval of the data sets increase

The changes in the RMSE performance metric of both GRU and LSTM models as the time-interval of the used data set increases from 1-year to 2-years to 4-years are presented in Table 12. To test whether there is a statistical significance behind the variations of GRU's and LSTM's RMSE with the expansion of the time-interval, a paired t-test was applied. In Table 12, the  $p$ -values reveal that there is a statistical significance between the variations of RMSE as the time-series size increase through an increase in the time-interval. The RMSE of the GRU model on the 1-year interval, 0.0038, increases by 163% to become 0.01 as the time-interval expands to reach 2-years. The increasing pattern in GRU's RMSE continues as the time interval increases again from 2-years to 4-years, RMSE grows by 40% to become 0.014. The rise in RMSE with an increasing time-interval shows that the performance of GRU drops in minimizing the frequency of large errors. The RMSE of LSTM exhibits similar responses to the increase in time-interval. The RMSE computed over the 1-year data set, 0.0069, increases by 24.6% to reach 0.0086 in the second experiment and increases again by 144% to become 0.021 using the 4-years interval. The changes of LSTM's RMSE as the time-interval of the data set increases indicate the increase in the frequency of large errors while predicting the closing price. The performance of both GRU and LSTM in Bitcoin price prediction based on the frequency of large errors, measured by RMSE, drops and scales negatively with the increase in time-interval and time-series size. The increase in RMSE of the implemented models correlates

with the increase in the range of the predicted closing price before normalization, shown in Figure 10, as the data set increases its interval throughout the three experiments. The increase in the range of the closing price increases the probability of producing large errors, thus the reported behavior of RMSE of GRU and LSTM is considered normal and expected.

Model	1-year	2-years	4-years	<i>p</i> -Value
GRU	0.0038	0.01	0.014	0.04
LSTM	0.0069	0.0086	0.021	0.045

**Table 12.** The evolution of RMSE of the implemented LSTM and GRU models as the time-interval of the data sets increase

## 8. Conclusion and Future Work

Bitcoin price is characterized with high volatility over short period of time that create issues and risks for investors. Predicting the price of Bitcoin with a decent methodology does not only reduce the probability of risk, but also takes advantage of the price fluctuations as a rapid process to make capital. This makes price prediction of high importance for investors. LSTM and GRU are two models that exhibited extraordinary ability in learning the underlying patterns of the Bitcoin historical time-series and forecasting predicted prices with high performance. Both LSTM and GRU have a internal mechanisms to specifically work with long sequences of data where a step in this sequence is dependent on all previous steps. This design fits the requirements for efficiently predicting the price of Bitcoin as a time-step in the time-series sequence of Bitcoin is dependent on all previous time-steps. Although both LSTM and GRU perform well in the task of price prediction, the concern of investors is which model is more worthy of investing in. The historical time-series of Bitcoin increases in size every moment, so it is not efficient to work with a model that performs well over a time-series that has a constant number of time-steps. This choice will disregard the ability of the model of maintaining and scaling its performance as the size of the time-series increase which is a definite behavior of the real Bitcoin historical time-series. Previous research attempted to compare the performance of LSTM and GRU as the time-series size varies. The variation of the time-series size was done through taking a time-series of constant size and partitioning it into different time-intervals to create multiple time-series of varying and increasing sizes. Although this approach properly produces answers to the scalability of the performance of LSTM and GRU as the time-series size increase, it still does not replicate the real behavior of Bitcoin historical time-series. The increase in size of the real Bitcoin historical time-series is due to the increase of the time-interval and not partitioning of a given time-interval. The expansion in the time-interval is due to the addition of new data that drives new patterns in the time-series. A model that is desired by investors should have a scalable performance with the increase of the size of the time-series caused by the increase of the time-interval of the time-series. This approach grants investors the opportunity to possess a prediction model that is capable of handling the behavior of the real Bitcoin historical time-series and predicting prices accurately on a long term and not just a temporary period. The results of this paper provide insight on the performance scalability of LSTM and GRU using a time series that increases in size through the expansion of its time-interval. Results indicate that the performance of LSTM is more scalable than the performance of GRU in Bitcoin price prediction and is more suitable for real-world investments. Future works can further tackle the comparison of scalability between the performance of LSTM and GRU by including a larger set of Bitcoin features and a larger data set to work with through its various time-intervals. The work of future research can also extend to take into consideration a comparison between LSTM and GRU based on a classification Bitcoin price prediction problem (whether price goes up or down) rather than a classic regression approach. Considerations of exposing the performance scalability of other machine learning models in Bitcoin price prediction through data sets of different time-intervals may as well be included in further efforts.

## References

1. S. Ji, J. Kim, and H. Im, "A Comparative Study of Bitcoin Price Prediction Using Deep Learning," *Mathematics* **2019**, 7, pp. 898.
2. A. Dutta, S. Kumar, and M. Basu, "A Gated Recurrent Unit Approach to Bitcoin Price Prediction," *Journal of Risk and Financial Management* **2020**, 13, pp. 23.



3. Y. Li, Z. Zheng, and H.-N. Dai, "Enhancing Bitcoin Price Fluctuation Prediction Using Attentive LSTM and Embedding Network," *Applied Sciences* **2020**, *10*, pp. 4872.
4. Y. Hua, "Bitcoin price prediction using ARIMA and LSTM," *E3S Web of Conferences* **2020**, *218*, pp. 1050.
5. H. Jang and J. Lee, "An Empirical Study on Modeling and Prediction of Bitcoin Prices With Bayesian Neural Networks Based on Blockchain Information," *IEEE Access* **2017**, *6*.
6. M. Rizwan, Sanam Narejo, and M. Javed, "Bitcoin price prediction using Deep Learning Algorithm," *13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)* **2019**.
7. P. Lamothe-Fernández, D. Alaminos, P. Lamothe-López, and M. A. Fernández-Gámez, "Deep Learning Methods for Modeling Bitcoin Price," *Mathematics* **2020**, *8*, pp. 1245.
8. G. Cohen, "Forecasting Bitcoin Trends Using Algorithmic Learning Systems," *Entropy* **2020**, *22*, pp. 838.
9. K. Rathan, S. Venkat Sai, and T. Sai Manikanta, "Crypto-Currency price prediction using Decision Tree and Regression techniques," *Third International Conference on Trends in Electronics and Informatics (ICOEI 2019)* **2019**.
10. Y. Hirano, L. Pichl, C. Eom, and T. Kaizoji, "ANALYSIS OF BITCOIN MARKET EFFICIENCY BY USING MACHINE LEARNING," *CBU International Conference Proceedings* **2018**, *6*, pp. 175-180.
11. K. H. U. Kumar Sinha, and S. S. Jain, "Performance Evaluation of Machine Learning Algorithms for Bitcoin Price Prediction," *Fourth International Conference on Inventive Systems and Control (ICISC 2020)* **2020**.
12. N. Uras, L. Marchesi, M. Marchesi, and R. Tonelli, "Forecasting Bitcoin closing price series using linear regression and neural networks models," *PeerJ Computer Science* **2020**, *6*, pp. 279.
13. T. Phaladisailoed and T. Numnonda, "Machine Learning Models Comparison for Bitcoin Price Prediction," *10th International Conference on Information Technology and Electrical Engineering (ICITEE)* **2018**.
14. M. Seo and G. Kim, "Hybrid Forecasting Models Based on the Neural Networks for the Volatility of Bitcoin," *Applied Sciences* **2020**, *10*, pp. 4768.
15. A. M. Balfagih and V. Keselj, "Evaluating Sentiment Classifiers for Bitcoin Tweets in Price Prediction Task," *IEEE International Conference on Big Data (Big Data)* **2019**.
16. Y. Xu, Bitcoin Price Forecast Using LSTM and GRU Recurrent networks, and Hidden Markov Model. Master of Science in Statistics, University of California, Los Angeles U.S., **2020**.
17. Y. Li and W. Dai, "Bitcoin price forecasting method based on CNN-LSTM hybrid neural network model," *The Journal of Engineering* **2020**, *2020*, pp. 344-347.
18. T. Ray Volstad, Bitcoin Value Prediction: How Random Forest, ARIMA, and Logistic Regression's Predictive Potential Performs When Employed on Bitcoin's Twitter Sentiment, Google Trend and Volatility Data. Master of Science in Data Science, Utica College, Utica U.S., **2019**.
19. S. Velankar, S. Valecha, and S. Maji, "Bitcoin Price Prediction using Machine Learning," *International Conference on Advanced Communications Technology (ICTACT)* **2018**.
20. Z. Chen, C. Li, and W. Sun, "Bitcoin price prediction using machine learning: An approach to sample dimension engineering," *Journal of Computational and Applied Mathematics* **2020**, *365*, pp. 112395.
21. S. McNally, Predicting the price of Bitcoin using Machine Learning. Master of Science in Data Analytics, National College of Ireland, Dublin Ireland, **2016**.
22. M. Liu, G. Li, J. Li, X. Zhu, and Y. Yao, "Forecasting the price of Bitcoin using deep learning," *Finance Research Letters* **2021**, *40*, pp. 101755.
23. L. Cocco, R. Tonelli, and M. Marchesi, "Predictions of bitcoin prices through machine learning based frameworks," *PeerJ Computer Science* **2021**, *7*, pp. 413.
24. J.-P. Huang and G. Sembiring Depari, "Forecasting Bitcoin Return: A Data Mining Approach" *Review of Integrative Business and Economics Research* **2020**, *10*.
25. H. Sebastião and P. Godinho, "Forecasting and trading cryptocurrencies with machine learning under changing market conditions," *Financial Innovation* **2021**, *7*.
26. I. E. Livieris, N. Kiriakidou, S. Stavroyiannis, and P. Pintelas, "An Advanced CNN-LSTM Model for Cryptocurrency Forecasting," *Electronics* **2021**, *10*, pp. 287.
27. A. Shankhdhar, A. K. Singh, S. Naugraiya, and P. K. Saini, "Bitcoin Price Alert and Prediction System using various Models," *IOP Conference Series: Materials Science and Engineering* **2021**, *1131*, pp. 12009.
28. R. K. Alkhodhairi, S. R. Aljalhami, N. K. Rusayni, J. F. Alshobaili, A. A. Al-Shargabi, and A. Alabdulatif, "Bitcoin Candlestick Prediction with Deep Neural Networks Based on Real Time Data," *Computers, Materials & Continua* **2021**, *68*, pp. 3215-3233.
29. Cryptocurrency Prices, Charts And Market Capitalizations. Available online: <https://coinmarketcap.com/> (accessed Mar. 11, 2022).
30. Nakamoto, S. Bitcoin: A Peer-To-Peer Electronic Cash System. Bitcoin. Available online: <https://git.dhimmel.com/bitcoin-whitepaper/> (accessed Mar. 11, 2022).
31. S. Abboushi, "Global virtual currency—Brief overview," *Journal of Applied Business and Economics* **2017**, *19*, pp. 10-18.
32. A. Urquhart, "Price clustering in Bitcoin," *Economics Letters* **2017**, *159*, pp. 145-148.

33. Zheng, Z.; Xie, S.; Dai, H.; Chen, X.; Wang, H. "An overview of blockchain technology: Architecture, consensus, and future trends," *IEEE International Congress on Big Data (BigData Congress)* **2017**, pp. 557-564.
34. Chen, W.; Wu, J.; Zheng, Z.; Chen, C.; Zhou, Y. "Market Manipulation of Bitcoin: Evidence from Mining the Mt. Gox Transaction Network," *IEEE INFOCOM 2019-IEEE Conference on Computer Communications* **2019**, pp. 964-972.
35. R. Albariqi and E. Winarko, "Prediction of Bitcoin price change using neural networks," *Int. Conf. on Smart Technology and Applications* **2020**, pp. 1-4.
36. Siامي-Namini, Sima, and Akbar S. Namin. 2018. Forecasting Economics and Financial Time Series: ARIMA vs. LSTM. Available online: <https://arxiv.org/abs/1803.06386v1> (accessed Oct. 10, 2021).
37. McNally, Sean, Jason Roche, and Simon Caton. 2018. "Predicting the Price of Bitcoin Using Machine Learning," *26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)* **2018**.
38. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation* **1997**, 9, pp. 1735-1780.
39. Chung, Junyoung, Caglar Gulcehre, Kyung H. Cho, and Yoshua Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," **2014**.
40. Working of RNN in TensorFlow - Javatpoint. Available online: <https://www.javatpoint.com/working-of-rnn-in-tensorflow> (accessed Apr. 22, 2022).
41. M. Phi, Illustrated Guide to LSTM's and GRU's: A step by step explanation. Available online: <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21> (accessed Apr. 22, 2022).
42. RNN, LSTM GRU. Available Online: <http://dprogrammer.org/rnn-lstm-gru> (accessed Apr. 23, 2022).