Forecasting of GPU Prices Using Transformer Method

1st Risyad Faisal Hadi
School of Computing
Telkom University
Bandung, Indonesia
risyadfaisalhadi@student.telkomuniversity.ac.id

2nd Siti Saadah
School of Computing
Telkom University
Bandung, Indonesia
sitisaadah@telkomuniversity.ac.id

3rd Didit Adytia
School of Computing
Telkom University
Bandung, Indonesia
adytia@telkomuniversity.ac.id

Abstract— **GPU or VGA (graphic processing unit) is a vital component of computers and laptops, used for tasks such as rendering videos, creating game environments, and compiling large amounts of code. The price of GPU/VGA has fluctuated significantly since the start of the COVID-19 pandemic in 2020. This research aims to forecast future GPU prices using deep learning-based time series forecasting using the Transformer model. We use daily prices of NVIDIA RTX 3090 Founder Edition as a test case. We use historical GPU prices to forecast 8, 16, and 30 days. Moreover, we compare the results of the Transformer model with two other models, RNN and LSTM. We found that to forecast 30 days; the Transformer model gets a higher coefficient of correlation (CC) of 0.8743, a lower root mean squared error (RMSE) value of 34.68, and a lower mean absolute percentage error (MAPE) of 0.82 compared to the RNN and LSTM model. These results suggest that the Transformer model is an effective and efficient method for predicting GPU prices.**

Keywords— GPU, Transformer, Forecasting, Time Series Forecasting

I. INTRODUCTION

A. Background

In today's world, the shortage of graphics cards has caused much concern and frustration for people who used computers as their primary tools for jobs. The high cost of these cards makes it difficult for people to afford them, hindering their ability to play games or create content. The fluctuating prices of GPUs further exacerbate the problem, and NVIDIA, one of the leading producers of these cards, must rely on third-party manufacturers for their chipsets. Manufacturers experiencing disruptions in their operations has led to a scarcity of graphics processing units (GPUs) and longer wait times for their production. As a result, the cost of GPUs such as the NVIDIA RTX 3090, which originally had a suggested price of $699, has skyrocketed to as much as $2,400 overnight. This scarcity and cost of graphics cards is a pressing issue that requires attention [1].

For those reasons people are trying to find the perfect time when they can buy a GPU. The forecasting method can be the way to solve the problem. Forecasting is a process of predicting based on historical data and extracting trends that can be approached using statistical or machine learning [2]. In [3]they study the GPU NVIDIA GTX 1060, which is affected by the bitcoin price. In this research, they are using linear regression models to forecast the upcoming price of GPUs. They found that the bitcoin's price affects the GPU's price. Another research that some researchers from CEEJ have done showed that the price of the GPU stock could be forecast using an optimal machine learning technique, the Nested Cross Validation algorithm [4].

One way to approach forecasting GPU prices is to use a deep learning model. This paper uses the recently developed transformer model initially developed to solve the NLP problem. In the transformer, from the input sequence, the model determines what other parts of the sequence are essential at each step [5]. The transformer has two parts: the encoder and the decoder. Theoretically, the transformer will use historical data to predict the upcoming prices in the experiment that some researchers have done. They are comparing the transformer and RNN. Transformers show up with excellent results and significant improvement [6]. In another experiment comparing transformers and LSTM, the transformer came out with a huge benefit because it is more stable and doesn't need so much time to train [7]. For this research, we will use the encoder layer to forecast the time series data that we have collected from keepa.

B. Problem Statement

In this study, we use the transformer model to predict GPU prices and evaluate its accuracy compared to other time series forecasting methods. Transformer models are cutting-edge machine learning techniques with success in natural language processing and time series prediction. This project evaluates the effectiveness of the transformer model in predicting GPU prices by analyzing historical data. The purpose of this study is to provide valuable insights to industry stakeholders and demonstrate the model's superiority over other commonly used forecasting methods.

C. Objective

This study compares Transformer models with LSTM and RNN architectures to predict GPU prices. Transformer is optimized for prediction on sequential data and can provide faster predictions in a single run. Only one GPU model was analyzed, and data was collected from September 2020 to November 2022. Forecast results are evaluated using coefficient correlation, mean squared error, and mean percent error. These metrics assess the effectiveness of the developed model.

II. THEORITICAL REVIEW

A. Literature review

In this study, we are focusing on how good the transformer model can predict the prices of GPU. The transformer model has been used for time series forecasting. In [8], they found that the transformer is effective for forecasting since they come up with a good result [8]. In [9], the transformer is used for solving time series forecasting; from their study, the transformer performs better than the LSTM and RNN-based methods. In [10], the transformer was used for forecasting both univariate and multivariate time series forecasting.

1. Transformer

Transformer architecture has an encoder and decoder. The encoder later will map an input sequence of sequence symbols of continuous representation of z. Then at the same time, the decoder will generate an output sequence of symbols. At each step, the model uses the previously generated symbol as additional input when generating the next symbol. The layers on the transformer are fully connected to each other [11]. The architecture of the transformer will be presented on Fig 1[11].

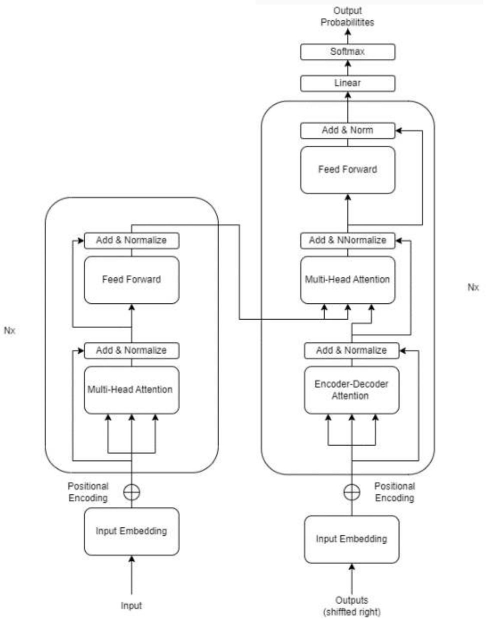

FIG 1.
Transformer Architecture

As stated, before the encoder and the decoder are fully connected. The left side is the encoder, and the right side is the decoder. This architecture contains 2 main attentions, the attention itself is used to mapping a query and set-of-key-value pairs to an output, where all the variables are vectors [11]. The attention that is built in the architecture is scaled dot-product attention and multi-head attention. The scaled dot-product attention will calculate the softmax value, which is represented by this formula.

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \qquad (1)$$

The formula shows the attention with parameters Q, K, and V. Then, there are some steps to calculate the scaled dot attention. The first one is to compute the alignment scores by multiplying the set of queries packed in a matrix. Next, we need to scale the score of the alignment using $\frac{1}{\sqrt{d_k}}$Then we are applying a softmax operation to obtain a set of weights. This softmax function will convert the layers into a vector of probabilities. After we get the demanded weight, we multiply it with the value in matrix V.

Multi-head attention allows the model to focus on information from different representations at the same time. Square below elsewhere [11]. This method is represented by this equation.

$$MultiHead(Q,K,V) = Concat(head_{1,......,}head_v)W^O$$
$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \qquad (2)$$

From the equation above as you can see, with the same parameter we are trying to calculate the value of the multihead attention. First thing first, we need to compute the linearly projected versions of the queries, keys, and value through multiplication with the parameter of (QW_i^Q, KW_i^K, VW_i^V). Then we need to apply an attention function on each head function by multiplying the queries and the key matrices. Apply softmax and calculate the weight for the output. Concentrate the outputs of the $head_i = (1 \ldots . h)$. After that, to obtain the result we need to multiply it with weight matrix W^O. Fig 2 Will show my flowchart for transformer architecture.

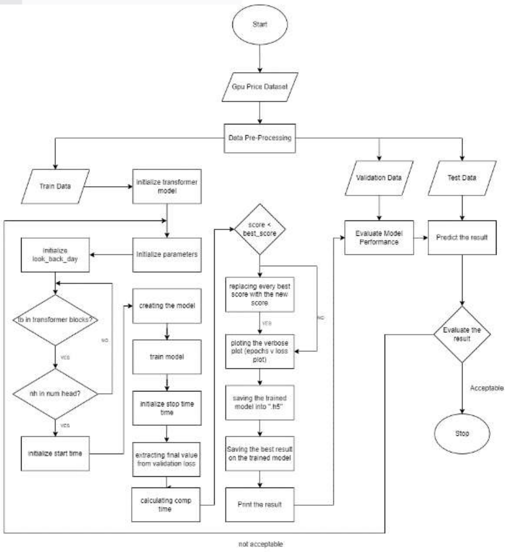

FIG 2.
Fowchart of Trasnformer

Figure 2 shows the Transformer model initialization, parameter initialization, and input after data sharing. Then the nested loops are executed, with the outer loop being the transformer block and the inner loop being the transformer head. The training start time is recorded, and after training is completed, the stop time and validation loss are recorded and the computation time is calculated. The model with the highest score is saved in the "h5" file. Next, the model's performance is evaluated and predictions are made. If you are satisfied, the investigation will be closed. Otherwise, hyperparameter tuning is performed to improve results.

2. Recurrent Neural Network

Recurrent Neural Network (RNN)is a type of artificial neural network that can process sequential data. It consists of a series of interconnected units that pass their output as input to the next unit, forming a directed graph. This allows the network to have an internal state or memory, enabling it to exhibit temporal dynamic behaviors. RNNs are particularly useful for recognizing patterns in sequential data, such as handwriting or speech recognition, or for predicting time-series data[12]. The simple RNN architecture can be seen on Fig 3 [13].

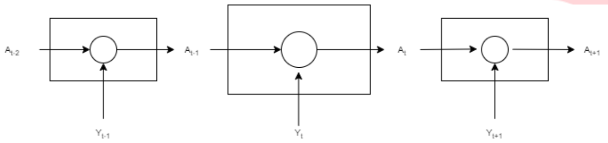

FIG 3.
Simple RNN Architecture

From the figure above we can see that every output will move to every RNN cell, well the RNN cell has its own architecture that will be shown in Fig 4 [14].

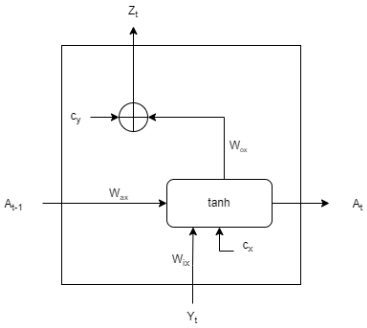

FIG 4.
RNN unit Architecture

The architecture of an RNN can be depicted as shown in Fig 6. At each time step *t* the state S_t is calculated based on the input Y_t and the previous state A_{t-1}using the formula.

$$A_t = \tanh(W_{ix}Y_t + W_{ax}A_{t-1} + c_x) \qquad (3)$$

In the equation, the matrices W_{ix} and W_{ax} are the weight matrices at the input and hidden layers, respectively, and c_x is the bias term. The activation function used in the equation is the hyperbolic tangent function, which is defined as follows.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad (4)$$

The range of the hyperbolic tangent function is from -1 to 1. The output value Z_t is calculated using the following formula.

$$Z_t = W_{ox}A_t + c_y \qquad (5)$$

In the equation, Z_t represents the output, W_{ox} is the weight matrix at the output layer, A_t is the state, and c_y is the bias term. Here is the flowchart for my RNN model.

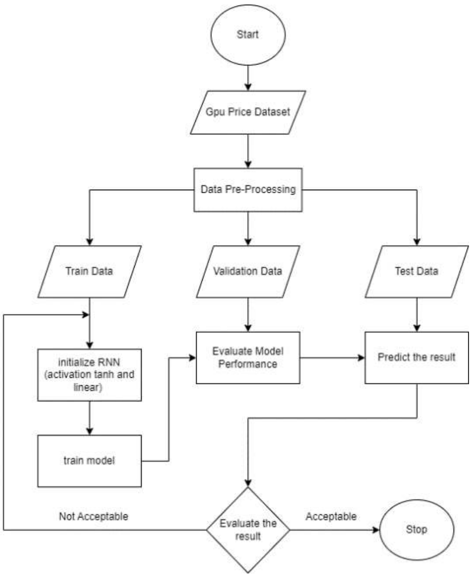

FIG 5.
RNN Flowchart

From Fig 5, the flowchart for the Recurrent Neural Network (RNN) can be seen. The process begins by splitting the data into train and test sets. The RNN model is then initialized, as previously mentioned, the RNN uses two types of activation functions, tanh and linear activation. After that, the trained model is then evaluated using validation data, predictions are made, and the results are obtained. If the results are not satisfactory, hyperparameter tuning is performed until the desired outcome is achieved.

3. Long Short-Term Memmory

Long Short-Term Memory (LSTM) neural network, first introduced by Hochreiter and Schmidhuber in 1997, has an input layer, one or more hidden layers, and an output layer. The hidden layers contain memory cells with input, output and forget gates to regulate the flow of information. The core component of each hidden layer is a memory block, made up of a group of memory cells that share the same gate units. It was later improved by Gers et al. by adding a forget gate [15].The architecture itself will be shown by Fig 6.

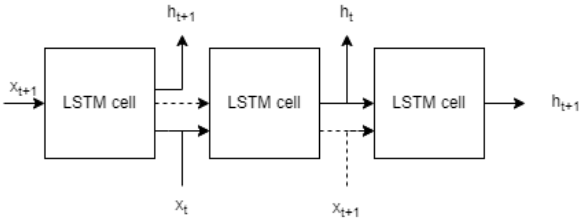

FIG 6.
LSTM architecture

Fig 8 describe that the inputs will go through LSTM cell and every cell has their own structure that will be show on Fig 7 [14].

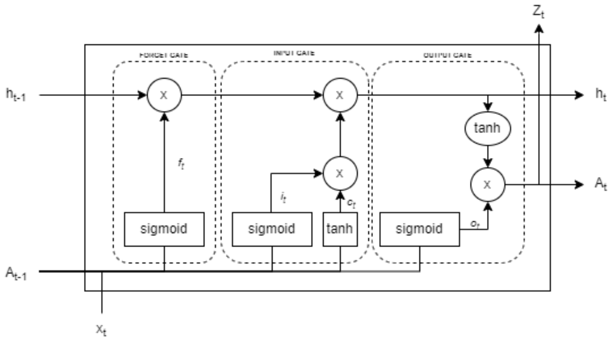

FIG 7.
LSTM cell architecture

From the architecture above we can see that there are 3 main gates on LSTM. Forget gate, Input Gate and Output Gate. Each gate receives two input vectors: the current input, X_t, and the previous output, A_{t-1}. The input gate determines which input values will be used in the current time step, the forget gate determines which values from the previous time step should be forgotten, and the output gate determines which values should be output in the current time step. Together, these gates allow the LSTM cell to effectively store and retrieve information over long periods of time [16].
From its we can also see that the first step of LSTM is the forget gate. In this function (6) X is the input value, and the output is a value between 0 and 1. If the output of the sigmoid function is close to 0, the data will be discarded. If the output is close to 1, the data will be updated or passed through.
In the context of an LSTM (Long Short-Term Memory) cell, the sigmoid function is used to determine the values of the input, forget, and output gates. The input x is a combination of the current input value, X_t, and the hidden state of the previous time step, A_{t-1}. The coefficients W and b are learned during the training process and are used to weight the input values.

Overall, the sigmoid function plays a crucial role in the LSTM cell's ability to effectively store and retrieve information over long periods of time.

$$f_t = \sigma\left(W_{fx}X_t + W_{fs}A_{t-1} + b_f\right) \qquad (6)$$

After the first step is done, then we are moving to next step which is the input gate. The output of the input gate can be calculated by using these formula

$$i_t = \sigma(W_{ix}X_t + W_{is}A_{t-1} + b_i) \qquad (7)$$

$$\widetilde{c_t} = \tanh(W_{cx}X_t + W_{cs}A_{t-1} + b_c) \qquad (8)$$

$$C_t = f_t \times c_{t-1} + i_t \times \widetilde{c_t} \qquad (9)$$

Finally, the last gate which is output gate that can be interpreted by this equation.

$$o_t = \sigma(W_{ox}X_t + W_{os}A_{t-1} + b_o) \qquad (10)$$

$$A_t = o_t \times \tanh(C_t) \qquad (11)$$

The output gate is calculated using the sigmoid function, as defined in equation (10). The output value, namely the cell state value C_t, is then forwarded to the next memory cell calculation, and the current hidden state A_t is generated using the hyperbolic tangent function, as defined in equation (11). Overall, the output gate plays a crucial role in the LSTM cell's ability to store and retrieve information over long periods of time. It allows the cell to selectively pass on information from one time step to the next, enabling it to capture long-term dependencies in data. The flowchart can be seen on Fig 8.

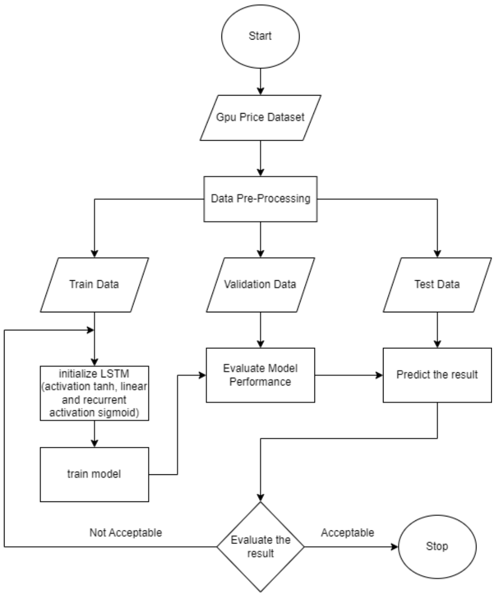

FIG 8.
LSTM flowchart

Figure 8 above shows the flow chart of the LSTM model. It is very similar to RNN flow chart. The main difference is in the activation layer. As you can see from the above architecture, LSTM has three activation layers, here iterative activation layer of Sigmoid. After the model is trained, we can use the validation data to evaluate the results and make predictions. If the results are unacceptable, the load should be repeated. So the research stopped.

4. Evaluation Matrix

For evaluating every model and to comparing each model we are using three evaluation metrices. CC, RMSE and MAPE. Coefficient Correlation (CC) is a statistical measure of the relationship between two variables. When two variables are correlated, a change in the value of one variable is associated with a change in the value of the other variable. The direction of this association can be positive, meaning that the two variables increase or decrease together, or negative, meaning that one variable increases as the other decreases.

The Pearson correlation coefficient is a common measure of correlation that is used to quantify the strength and direction of a linear relationship between two continuous variables. It is typically used when the data follows a bi-variate normal distribution, meaning that the variables are jointly normally distributed. The Pearson correlation coefficient can range from -1 to 1, with values closer to -1 indicating a strong negative correlation, values closer to 1 indicating a strong positive correlation, and values closer to 0 indicating a weaker or no correlation [17]. The coefficient correlation is represented by this equation.

$$r = \frac{n(\sum xy)-(\sum x)(\sum y)}{[n\sum_x 2-(\sum x)^2][n\sum_y 2-(\sum y)^2]} \qquad (12)$$

The symbols represent various sums and products of two values (x and y) in a dataset with n = amount of data and $\sum x$ represent the sum of the first value, $\sum y$ represent the sum of the second value, $\sum xy$ represent the sum of product of the first and the second value, $\sum_x 2$ represents the sum of the square of the first value, and $\sum_y 2$ represents the sum of the square of the second value.

Root Mean Squared Error (RMSE) is a measure of the difference between the predicted values of a model and the actual values. It is often used as a metric for evaluating the performance of a predictive model, such as a regression model. The RMSE is a measure of the average magnitude of the error in the model's predictions. A lower RMSE indicates a better fit of the model to the data, while a higher RMSE indicates a poorer fit. The RMSE can be show by this equation.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \qquad (13)$$

From the equation there is $\hat{y}_i$ and y_i. The first variable relies on the result of the prediction value and the other one is the actual value from the data. Last, we have *n,* where it is implied to the amount of the prediction [18].

Another method to measure the accuracy of prediction models is MAPE. MAPE or Mean Absolute Percentage Error is performance matrix beside RMSE that can be used to measure the accuracy of prediction on forecasting. The difference is we are using percentage as the benchmark. This method can be represented by this equation.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|A_i - F_i|}{|A_i|} \qquad (14)$$

From the equation above, $\frac{1}{n}$ can be changed to 100% since the result would be on percentage value. The A_i variable is implying the actual value form the data and we have F_i which is the forecast value of the data. Lastly just like RMSE we have n where it relies on the amount of the total number of observation data [19].

III. METHOD

A. System Design

To do forecasting using a transformer, they are several steps to do shown in Fig. 9.

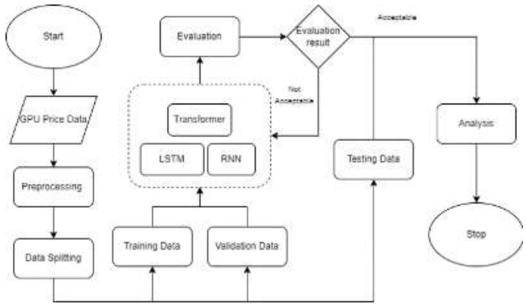

FIG 9.
Flowchart of Research Architecture

A flowchart of the architecture is shown in Figure 9. It details the steps from data collection to survey completion. Select Time Series Forecasting as the forecasting method. After data preprocessing is performed to ensure data purity, the data is split into training, validation, and test sets. Transformer, LSTM and RNN models have been trained, validated and tested. Accuracy is evaluated and analyzed. If the accuracy is greater than 97%, the best results for each model are analyzed and the results are evaluated using coefficient correlation, RMSE and MAPE. Study ends when all steps are completed and eligibility criteria are met.

1. Dataset

The dataset that will be used in my research is the time series dataset. This dataset is downloaded or obtained from the keepa website that can be accessed through keepa website [20]. The dataset contains 783 rows and 4 attributes, where the dataset shows the price from the first time the GPU is launched until November 2022 which will be shown in Table I.

TABLE 1.
Dataset Example

Date	Price (USD)	Last (USD)	Future (USD)
2020-09-21	2000	0	4000
2020-09-22	4000	2000	4000
2020-09-23	4000	4000	3500
2020-09-24	3500	4000	3500

From Table 1 above, we can see that the data is the daily prices of the GPU. The Last Future and Difference is an additional feature that has been added manually. The used attribute in this paper is the first two columns which is Date and Price column.

2. Preprocessing Data

On this research, we do several preprocessing techniques. The data is recorded daily, we check every possibility on our dataset so that we can choose the proportional preprocessing technique. Firstly, we try to detect outlier, after we are dealing with the outlier we are using interpolated data to fill in the missing value on the data. Next, we are using the preprocessing technique where to reshape the data inputation so that it can go through the models. After that we do scaling or normalization so that the inputation will change into value from 0 - 1. Then we split the data with a ratio of 80% training data, 10% validation data, and 10% testing data. The visualization of these data will be shown in this Fig 10.

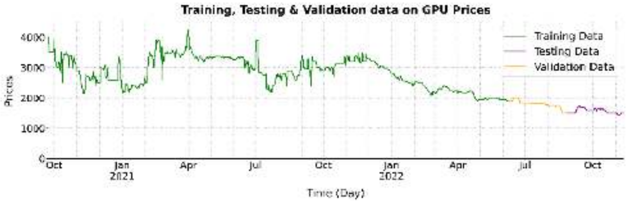

FIG 10.
Visualization of splitted data

IV. RESULTS AND DISCUSSION

A. Evaluation

The GPU pricing data used in this study covers the period from September 2020 to November 2022 and is split into 8-, 16-, and 30-day forecasts for usage. Hyperparameters are checked before training for fair model comparison. An analysis was performed to find the optimal epoch and stack size, and the results are stable at 300 epochs with a stack size of 16. The best dropout rate is defined as shown in Table III. Larger epoch sizes and smaller batch sizes can lead to more accurate solutions because there are more steps to update the weights during the optimization process.

TABLE 2.
Analysis on epoch and batch size

	Batch Size								
	16			32			64		
Epochs	CC	RMSE	MAPE	CC	RMSE	MAPE	CC	RMSE	MAPE
50	0.971	360.72	16.63	0.971	630.11	25.84	0.971	724.31	28.58
100	0.97	59.22	2.78	0.971	331	15.48	0.971	628.65	25.79
150	0.971	62.94	3.15	0.85	238.69	13.81	0.8511	238.55	13.8
200	0.971	54.66	2.68	0.85	238.69	13.81	0.8511	238.55	13.8
250	0.971	57.19	2.83	0.85	238.69	13.81	0.8511	238.55	13.8
300	0.971	32.12	1.11	0.85	238.69	13.81	0.8511	238.55	13.8
350	0.971	32.12	1.11	0.85	238.69	13.81	0.8511	238.55	13.8
400	0.971	32.12	1.11	0.85	238.69	13.81	0.8511	238.55	13.8

TABLE 3.
Analysis of dropout

	Accuracy		
Dropout	CC	RMSE	MAPE
0	0.971	32.12	1.11
0.15	0.909	196.37	10.99
0.25	0.890	200.57	11.25
0.5	0.8911	202.86	11.41
0.75	0.8837	215.01	12.21

According to the table, the best dropout value is 0, the dropout itself is used to prevent overfitting in the neural network model. Increasing the dropout rate decreases the model's ability to fit the training data and lowers accuracy. Also, in this case by adding small amount of dropout making the model less complex, leading to underfitting of the model. Therefore, we set the dropout rate to 0 for all models. We also selected a head size of 128 and a patience of 100. We tested various numbers of heads and transformer blocks which is [1,2,3] and number of heads of [1, 2], then the model determined the optimal combination to be 1 transformer block and 2 heads.

All the hyper parameter tunning is doing using all 10% of data test splitting. This optimum setting then applied on the other model. The prediction results from the three models will be compared using the Coefficient of Correlation (CC), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). The CC, RMSE and MAPE for each prediction method will be presented in tables. Before we training the data using the model, we first see how the model is built. The Figure 11, 12, and 13 will show how every model build their model. Then after we have the model, the model then used to do the training. The results will be used to evaluate the performance of each prediction method and determine the most accurate and efficient method for predicting the target variable. The result will be shown in Table IV Table V, and Table VI.

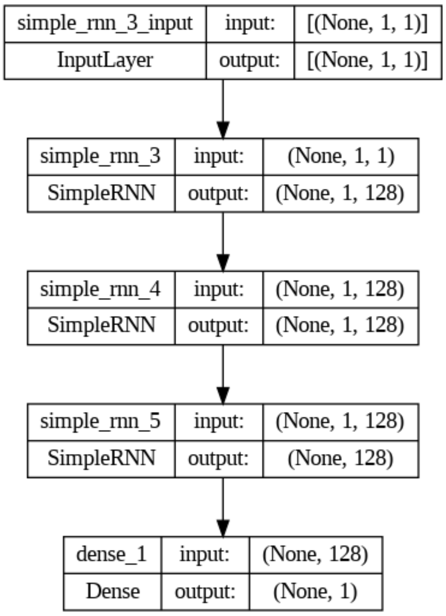

FIG 11.
Visualization of how the RNN model is built

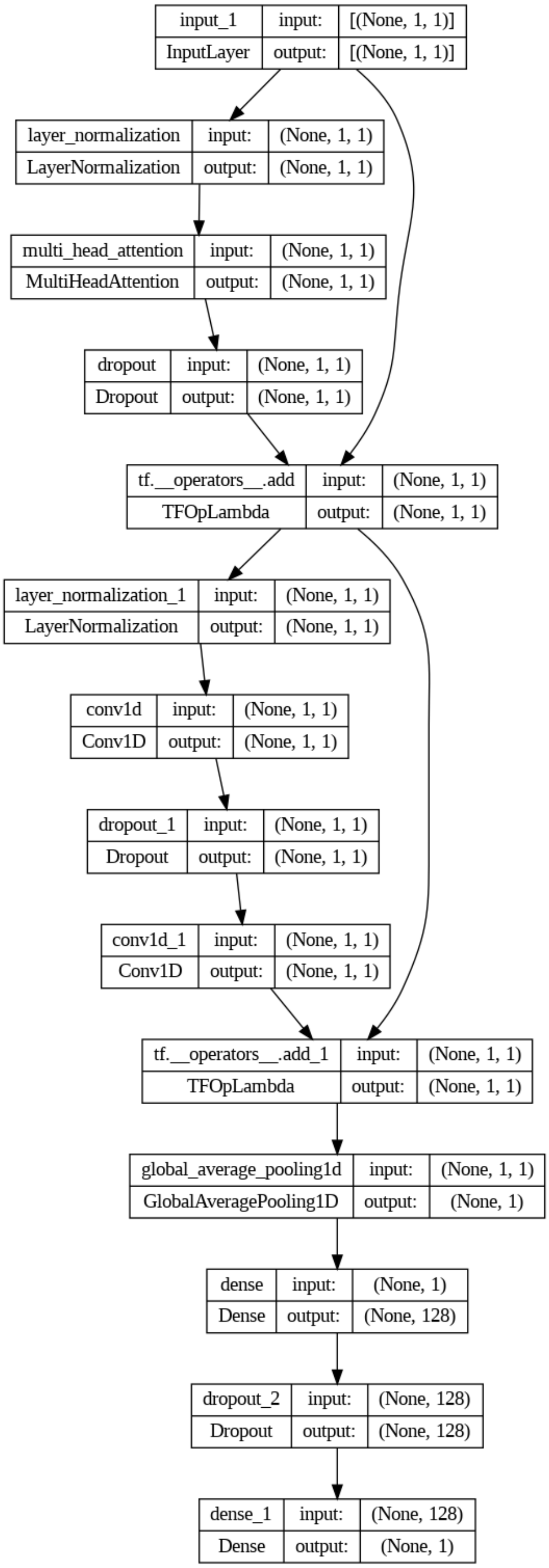
input_1 | input: | [(None, 1, 1)]
InputLayer | output: | [(None, 1, 1)]
layer_normalization | input: | (None, 1, 1)
LayerNormalization | output: | (None, 1, 1)
multi_head_attention | input: | (None, 1, 1)
MultiHeadAttention | output: | (None, 1, 1)
dropout | input: | (None, 1, 1)
Dropout | output: | (None, 1, 1)
tf.__operators__.add | input: | (None, 1, 1)
TFOpLambda | output: | (None, 1, 1)
layer_normalization_1 | input: | (None, 1, 1)
LayerNormalization | output: | (None, 1, 1)
conv1d | input: | (None, 1, 1)
Conv1D | output: | (None, 1, 1)
dropout_1 | input: | (None, 1, 1)
Dropout | output: | (None, 1, 1)
conv1d_1 | input: | (None, 1, 1)
Conv1D | output: | (None, 1, 1)
tf.__operators__.add_1 | input: | (None, 1, 1)
TFOpLambda | output: | (None, 1, 1)
global_average_pooling1d | input: | (None, 1, 1)
GlobalAveragePooling1D | output: | (None, 1)
dense | input: | (None, 1)
Dense | output: | (None, 128)
dropout_2 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_1 | input: | (None, 128)
Dense | output: | (None, 1)

FIG 12,
Visualization of how the transformer model is built

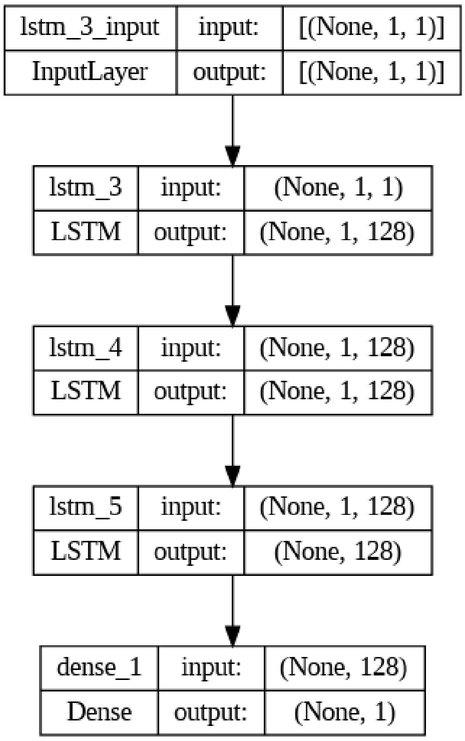
lstm_3_input | input: | [(None, 1, 1)]
InputLayer | output: | [(None, 1, 1)]
lstm_3 | input: | (None, 1, 1)
LSTM | output: | (None, 1, 128)
lstm_4 | input: | (None, 1, 128)
LSTM | output: | (None, 1, 128)
lstm_5 | input: | (None, 1, 128)
LSTM | output: | (None, 128)
dense_1 | input: | (None, 128)
Dense | output: | (None, 1)

FIG 13.
Visualization of how the LSTM model is built

1. Result

TABLE 4.
Accuracy of transformer

	Accuracy		
Days	**CC**	**RMSE**	**MAPE**
8	0.7387	33.1	0.861
16	0.682	44.82	0.96
30	0.8743	34.68	0.82

TABLE 5.
Accuracy of LSTM

	Accuracy		
Days	**CC**	**RMSE**	**MAPE**
8	0.7387	52.12	2.63
16	0.6817	72.27	3.09
30	0.8739	65.37	3.00

TABLE 6.
Accuracy of RNN

	Accuracy		
Days	**CC**	**RMSE**	**MAPE**
8	0.7387	61.03	3.03
16	0.6816	75.01	3.54
30	0.8738	75.01	3.51

From the table we can see that the accuracy on transformer is better than LSTM and RNN. The result is surpassing the result on other models. The ability of transformer to forecast are really close to the real data.

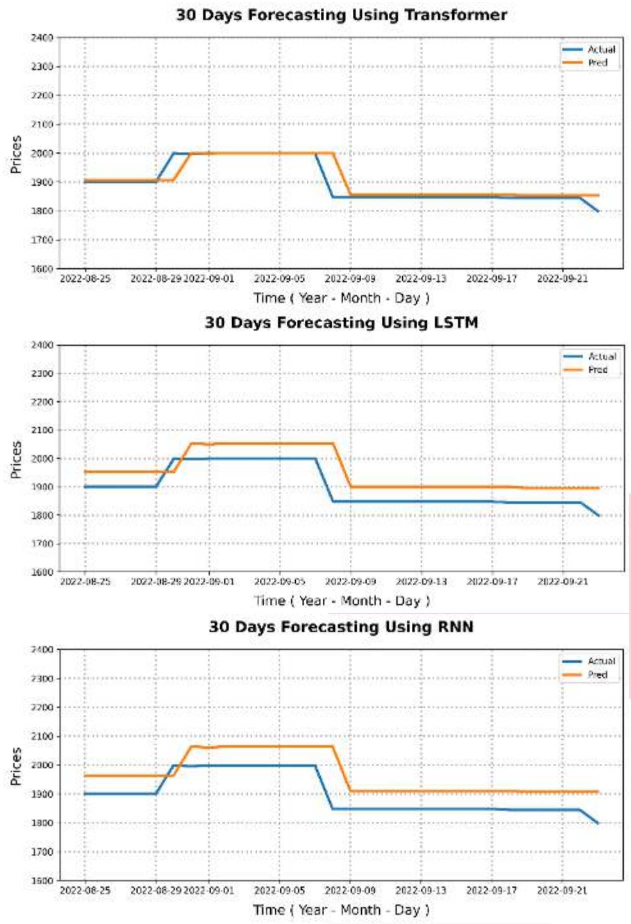

FIG 14.
Forecasting visualization for 30 days of transformer, LSTM, and RNN respectively

2. Analysis of result

According to the visualization results shown in Figure 14 and the numerical comparisons in Tables IV, V, and VI, the Transformer model appears to be more accurate than the LSTM and RNN models in predicting the target variable. This is due to the optimized layer structure. Additionally, a self-aware mechanism that allows the transformer to assign a weight to each input token and output the optimal weight for multiple input tokens. Also, the 16-day forecast results seem to follow a different pattern than the 8-day and 30-day forecasts. This performance difference can be attributed to several factors affecting the 16-day forecast horizon, including: B. Data quality and presentation, data variability, and data trend smoothness. Overall, these results demonstrate that the Transformer model is a more effective and efficient method for predicting the target variable compared to the LSTM and RNN models.

From the result we can see that the result is very different with the previous result [3]. Where the result by using only linear regression technique show the accuracy of 98.57%. compared with 99.18%. The transformer can also be used to forecast more than 1 day with more accurate results.

V. CONCLUSION

A. Conclusion

This paper investigates the Transformer model's ability to predict daily GPU prices over 8, 16, and 30-day periods using a dataset of NVIDIA RTX 3090 Founders Edition prices over 2 years. The Transformer model outperformed the RNN and LSTM models in terms of accuracy, demonstrating higher correlation coefficients and lower root mean squared and mean absolute percentage errors. This study concludes that the Transformer model is effective in predicting daily GPU prices and provides better accuracy compared to the RNN and LSTM models.

In recent years, the prices of GPUs have become an increasingly popular topic of interest, with many studies focused on forecasting their prices. The use of Transformer networks for this purpose has shown promising results. However, there is still much room for further research in this area. Compared to traditional methods like LSTM and RNN, transformer-based model could provide a better performance and accuracy. Nonetheless, it would be beneficial to evaluate the computational time and cost-effectiveness of the transformer-based model and compare it with other popular models. In addition, using more data or utilizing multiple GPUs during training could potentially improve the performance of the transformer model.

REFRENCES

[1] The Economist, “Crypto-miners are probably to blame for the graphics-chip shortage,” 2021. https://www.usnews.com/news/top-news/articles/2022-09-20/nvidia-unveils-new-gaming-chip-with-ai-features-taps-tsmc-for-manufacturing#:~:text=Nvidia%20designs%20its%20chips%20but,by%20Samsung%20Electronics%20Co%20Ltd. (accessed Apr. 25, 2022).

[2] Z. Zhao *et al.*, “Short-Term Load Forecasting Based on the Transformer Model,” *Information (Switzerland)*, vol. 12, no. 12, Dec. 2021, doi: 10.3390/INFO12120516.

[3] S. A. A. Leksono, Z. G. Prastyawan, and I. Veriawati, “Prediksi Harga Kartu Grafis Yang Dipengaruhi oleh Nilai Bitcoin,” *JURNAL ILMIAH FIFO*, vol. XI, no. 1, pp. 65–74, Apr. 2019.

[4] M. Chlebus, M. Dyczko, and M. Woźniak, “Nvidia’s Stock Returns Prediction Using Machine Learning Techniques for Time Series Forecasting Problem,” *Central European Economic Journal*, vol. 8, no. 55, pp. 44–62, Jan. 2021, doi: 10.2478/ceej-2021-0004.

[5] Maxime, “What is Transformer?,” 2019. https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04 (accessed Mar. 09, 2022).

[6] E. Yalta Soplin *et al.*, “A Comparative Study on Transformer Vs RNN in Speech Applications,” ASRU, 2019. [Online]. Available: http://www.merl.com

[7] A. Zeyer, P. Bahar, K. Irie, R. Schluter, and H. Ney, “A Comparison of Transformer and LSTM Encoder Decoder Models for ASR,” in *2019 IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019 - Proceedings*, Dec. 2019, pp. 8–15. doi: 10.1109/ASRU46091.2019.9004025.

[8] G. A. Galindo Padilha, J. R. Ko, J. J. Jung, and P. S. G. de Mattos Neto, “Transformer-Based Hybrid Forecasting Model for Multivariate Renewable Energy,” *Applied Sciences (Switzerland)*, vol. 12, no. 21, Nov. 2022, doi: 10.3390/app122110985.

[9] S. Li *et al.*, “Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting,” Jun. 2019.

[10] N. Wu, B. Green, X. Ben, and S. O’Banion, “Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case,” Jan. 2020, [Online]. Available: http://arxiv.org/abs/2001.08317

[11] A. Vaswani *et al.*, “Attention Is All You Need,” Jun. 2017.

[12] Han’guk T’ongsin Hakhoe, IEEE Communications Society, Denshi Jōhō Tsūshin Gakkai (Japan). Tsūshin Sosaieti, and Institute of Electrical and Electronics Engineers, *RNN-based Deep Learning for One-hour ahead Load Forecasting*. 2020.

[13] H. Apaydin, H. Feizi, M. T. Sattari, M. S. Colak, S. Shamshirband, and K. W. Chau, “Comparative analysis of recurrent neural network architectures for reservoir inflow forecasting,” *Water (Switzerland)*, vol. 12, no. 5, May 2020, doi: 10.3390/w12051500.

[14] D. Zhang, Q. Peng, J. Lin, D. Wang, X. Liu, and J. Zhuang, “Simulating reservoir operation using a recurrent neural network algorithm,” *Water (Switzerland)*, vol. 11, no. 4, Apr. 2019, doi: 10.3390/w11040865.

[15] M. S. Hossain and H. Mahmood, “Short-term photovoltaic power forecasting using an LSTM neural network and synthetic weather forecast,” *IEEE Access*, vol. 8, pp. 172524–172533, 2020, doi: 10.1109/ACCESS.2020.3024901.

[16] S. R. Venna, A. Tavanaei, R. N. Gottumukkala, V. v. Raghavan, A. S. Maida, and S. Nichols, “A Novel Data-Driven Model for Real-Time Influenza Forecasting,” *IEEE Access*, vol. 7, pp. 7691–7701, 2019, doi: 10.1109/ACCESS.2018.2888585.

[17] P. Schober and L. A. Schwarte, “Correlation coefficients: Appropriate use and interpretation,” *Anesth Analg*, vol. 126, no. 5, pp. 1763–1768, May 2018, doi: 10.1213/ANE.0000000000002864.

[18] M. A. Istiake Sunny, M. M. S. Maswood, and A. G. Alharbi, “Deep Learning-Based Stock Price Prediction Using LSTM and Bi-Directional LSTM Model,” in *2nd Novel Intelligent and Leading Emerging Sciences Conference, NILES 2020*, Oct. 2020, pp. 87–92. doi: 10.1109/NILES50944.2020.9257950.

[19] A. de Myttenaere, B. Golden, B. le Grand, and F. Rossi, “Mean Absolute Percentage Error for regression models,” *Neurocomputing*, vol. 192, pp. 38–48, Jun. 2016, doi: 10.1016/j.neucom.2015.12.114.

[20] Keepa, “NVIDIA GeForce RTX 3090 Founders Edition Graphics Card,” 2019. https://keepa.com/#!product/1-B08HR6ZBYJ (accessed May 04, 2022).