# Forecasting of GPU Prices Using Transformer Method

Risyad Faisal Hadi [1] *, Siti Saadah [2], Didit Aditya [3]
School of Computing [1], [2], [3]
Telkom University
Bandung, Indonesia
risyadfaisalhadi@student.telkomuniversity.ac.id [1], sitisaadah@telkomuniversity.ac.id [2], adytia@telkomuniversity.ac.id [3]

***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, due in part to the increased demand for GPUs for remote work and online activities. Furthermore, accurate GPU price forecasting can have broader implications beyond the computer hardware industry, with potential applications in investment decision-making, production planning, and pricing strategies for manufacturers. 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, Transformer 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 model is an effective and efficient method for predicting GPU prices.**

***Keywords*— GPU, Transformer, Forecasting, Time Series Forecasting**

***Abstrak*— GPU atau VGA (*graphic processing unit*) adalah komponen vital komputer dan laptop, digunakan untuk tugas-tugas seperti merender video, membuat lingkungan game, dan menyusun kode dalam jumlah besar. Harga GPU/VGA berfluktuasi secara signifikan sejak dimulainya pandemi COVID-19 pada tahun 2020, antara lain karena meningkatnya permintaan GPU untuk pekerjaan jarak jauh dan aktivitas daring. Selain itu, perkiraan harga GPU yang akurat dapat memiliki implikasi yang lebih luas di luar industri perangkat keras komputer, dengan aplikasi potensial dalam pengambilan keputusan investasi, perencanaan produksi, dan strategi penetapan harga untuk produsen. Penelitian ini bertujuan untuk meramalkan harga GPU pada masa mendatang menggunakan *time series forecasting* berbasis *deep learning* menggunakan model Transformer. Kami menggunakan harga harian NVIDIA RTX 3090 Founder Edition sebagai *test case*. Kami menggunakan harga GPU historis untuk memperkirakan 8, 16, dan 30 hari. Kemudian Transformer kita bandingkan hasil model Transformer dengan dua model lainnya, RNN dan LSTM. Kami menemukan bahwa untuk memperkirakan 30 hari; model Transformer mendapatkan koefisien korelasi (CC) yang lebih tinggi sebesar 0,8743, nilai *root mean squared error* (RMSE) yang lebih rendah sebesar 34,68, dan *mean absolute percentage error* (MAPE) yang lebih rendah sebesar 0,82 dibandingkan dengan model RNN dan LSTM. Hasil ini menunjukkan bahwa model tersebut merupakan metode yang efektif dan efisien untuk memprediksi harga GPU.**

***Kata Kunci*— GPU, Transformer, Forecasting, Time Series Forecasting**

## I. INTRODUCTION

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. Furthermore, the massive buying of GPUs for mining is causing a limitation in the availability of GPUs. 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

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v12i1.1569, Copyright ©2023
Submitted : January 17, 2023, Revised : March 6, 2023, Accepted : March 17, 2023, Published : March 27, 2023

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.

In In this study, we are focusing on how a good transformer can predict the prices of GPU. Besides that, we also compare the model with other architectures, such as LSTM and RNN. The transformer itself is designed for forecasting sequential data. By using a transformer, the prediction can be done faster since the process in this model only runs once. Also, we limit the GPU variation so that there will be only 1 GPU that will be used for this research. Additionally, we only take the data from the first time the GPU is launched; September 2020, until November 2022. The result of the prediction will be quantified by using Coefficient Correlation (CC), Root Mean Square Error (RMSE), and Mean Averaged Percentage Errors (MAPE). All these metrics will evaluate the result of the model we have made.

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. In the end the research will help people to determine what is the right time for them to buy the GPU and showing the power of transformer for forecasting.

This paper will be divided into several sections. The following section will discuss the related work of this study in the past related to forecasting GPU prices. Moreover, section 2 there will explain the methodology. Next, in section 3, there will be an explanation of the result and the analysis for this paper. Lastly, we have a conclusion to conclude what we have so far and to close the paper.

## II. METHODOLOGY

To do forecasting using a transformer, they are several steps to do shown in Fig. 1.

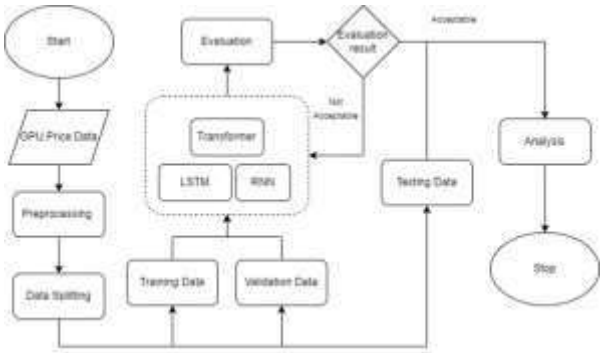



Fig. 1. Flowchart of Research Architecture

Fig. 1 shows the flowchart of the research's architecture. Starting from collecting data until the research is stopped. The method of forecasting that will be used is time series forecasting. After the data has been collected then we need to preprocess it to make the data as clean as possible, so the data is ready to use. Then, the data will be split into train, validation, and test data. Next, we will train, validate, and test the data by using transformer, LSTM, and RNN. After that, we need to evaluate your results first. If the accuracy is more than 95%, then we can continue to analyze gaining the optimum result of each model. Moreover, we will evaluate the data using Coefficient Correlation, RMSE, and MAPE to find the optimum result. After all the procedure has been completed and qualified, then the research will be stopped.

### A. 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 [11]. 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 1.

TABLE I. 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.

### B. 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. After that, 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 2.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v12i1.1569, Copyright ©2023
Submitted : January 17, 2023, Revised : March 6, 2023, Accepted : March 17, 2023, Published : March 27, 2023


Fig. 2. Visualization of splitted data

## C. Model

### a. 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 [12]. The architecture of the transformer will be presented on Fig 2.

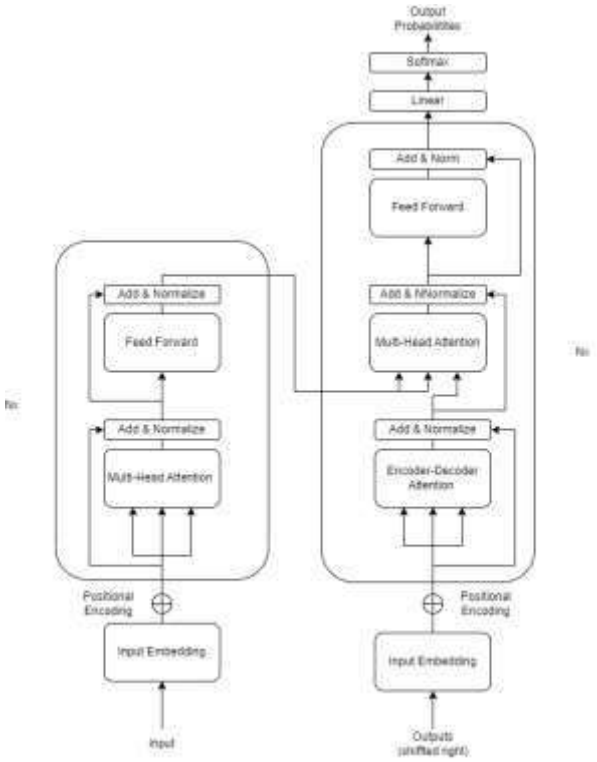



Fig. 3. Transformer Architecture

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 [12]. 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 (1).

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

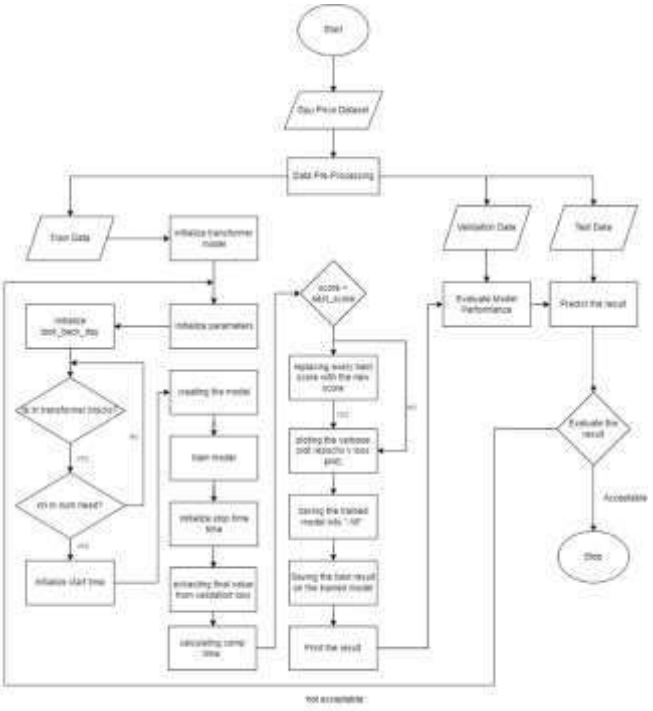
The formula shows the attention with parameters Q, K, and V. Then, there are some steps to calculate the scaled dot attention. The first one is to compute the alignment scores by multiplying the set of queries packed in a matrix. Next, we need to scale the score of the alignment using $\frac{1}{\sqrt{d_k}}$ 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 [12]. This method is represented by this equation (2).

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

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 4 will show my flowchart for transformation architecture



p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v12i1.1569, Copyright ©2023
Submitted : January 17, 2023, Revised : March 6, 2023, Accepted : March 17, 2023, Published : March 27, 2023

Fig. 4. Fowchart of Trasnformer

From Fig 4, it can be seen that after the data is split, the transformer model is initialized. Next, the parameters for the model are initialized and a look-back value is set for inputation. A nested loop is then implemented, with the outer loop iterating over the number of transformer blocks and the inner loop iterating over the number of transformer heads. The start time is recorded, and the model is trained. Once training is complete, the stop time is recorded and the validation loss is extracted. The computation time for training is then calculated. The best score is determined and the model is saved in an "h5" file format. The model's performance is then evaluated and predictions are made. If the results are satisfactory, the research is concluded. If not, hyperparameter tuning is carried out to achieve the desired outcome.

*b. 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[13]. The simple RNN architecture can be seen on Fig 5 [14].

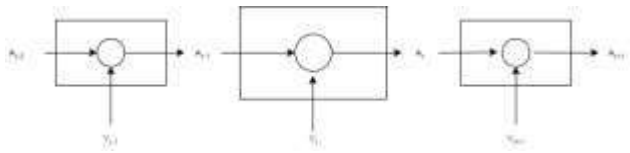


Fig. 5. Simple RNN Architecture

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 6 [15].

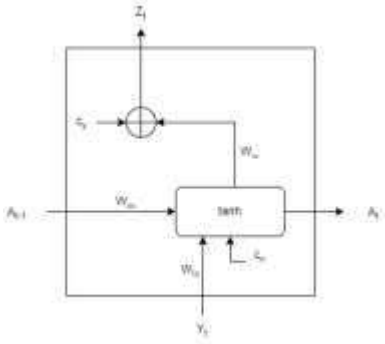



Fig. 6. RNN unit Architecture

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 (3).

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

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 (4).

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

The range of the hyperbolic tangent function is from -1 to 1. The output value $Z_t$ is calculated using the following formula (5).

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

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

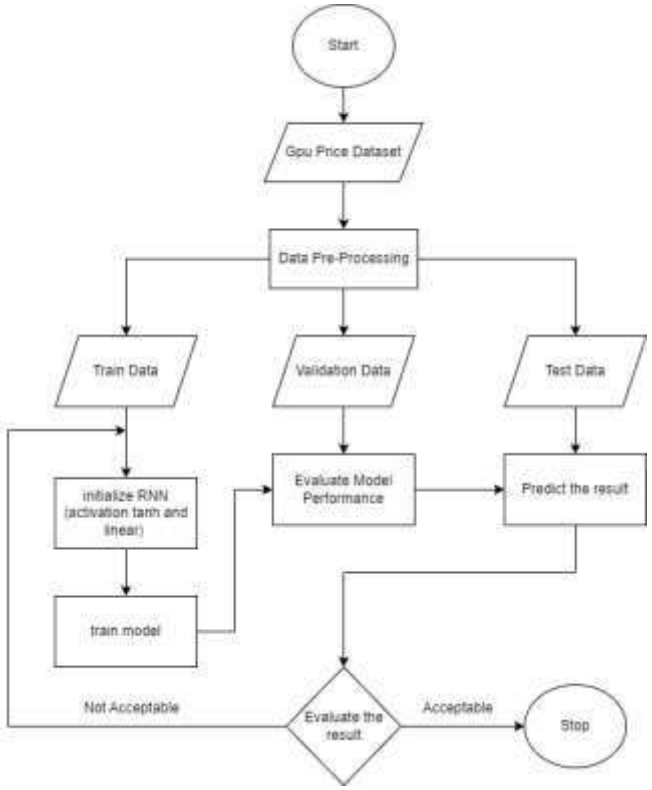



Fig. 7. RNN flowchart

From Fig 7, 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.

*c. Long Short-Term Memory.*

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

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v12i1.1569, Copyright ©2023
Submitted : January 17, 2023, Revised : March 6, 2023, Accepted : March 17, 2023, Published : March 27, 2023

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 [16].The architecture itself will be shown by Fig 8.

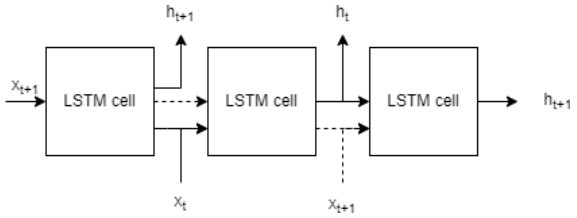



Fig. 8. LSTM architecture

Fig 8 describe that the inputs will go through LSTM cell and every cell has their own structure that will be show on Fig 9 [14].

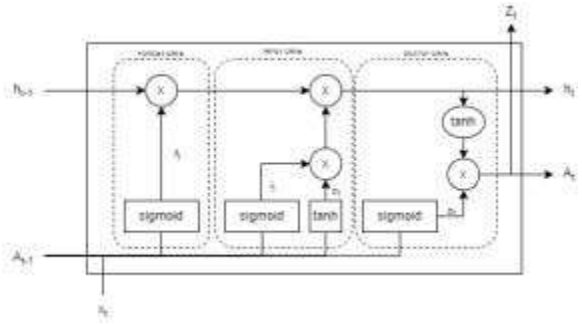



Fig. 9. LSTM cell architecture

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 [17].

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.

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

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

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

$$\tilde{c}_t = \tanh(W_{cx}X_t + W_{cs}A_{t-1} + b_c) \quad (8)$$

$$C_t = f_t \times c_{t-1} + i_t \times \tilde{c}_t \quad (9)$$

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

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

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

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 10

The Fig 10 above, showing the flowchart of our LSTM model. It is quite similar to the RNN flowchart. The main different is on the activatio layer. In LSTM we have 3 activation layer where here we have reccurent activation layer of sigmoid as what you can see from the architecture above. After the model is trained we can evalute the result with the validation data and making the prediction. If the result is nor accepatable we need to repeat the stress. Thus the research is stoped.

*D. 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 [18].

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v12i1.1569, Copyright ©2023
Submitted : January 17, 2023, Revised : March 6, 2023, Accepted : March 17, 2023, Published : March 27, 2023

TABLE II. 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 |

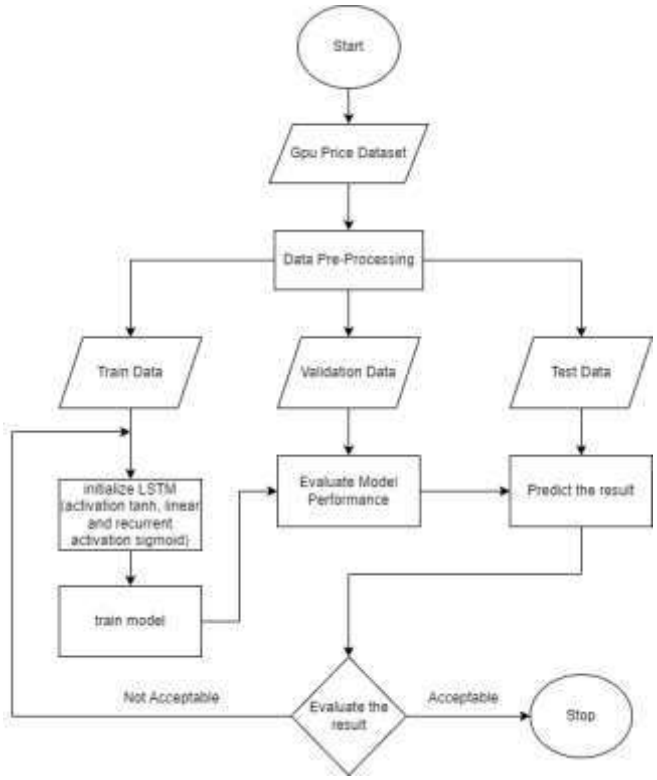



Fig. 10. LSTM flowchart.

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.

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

From the equation there is $\hat{y}$ 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 [19].

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.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} \frac{|A_i - F_i|}{|A_i|} \quad (13)$$

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 [20].

## III. RESULT AND ANALYSIS

The daily data of GPU prices that is used on this research is from the first time the GPU is launch which is September 2020 until November 2022. The data then splitted, where later if we want to forecast for 8, 16, and 30 days. We are just using8, 16, and 30 data from the splitted data test.

Before we train the data, we check the hyper parameters so that the test would be fair and equal for all models. After we do some analysis for the optimum hyper parameters. Here we try to change the epoch so we can find the optimum epoch and batch size. The optimum epoch for this study is 300 where the batch size is 16. The Table II will show the analysis of the epoch and batch size. After we find the optimum epochs and batch size, we then try to define the best dropout rate. Table III.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v12i1.1569, Copyright ©2023
Submitted : January 17, 2023, Revised : March 6, 2023, Accepted : March 17, 2023, Published : March 27, 2023

TABLE III. 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. 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.

TABLE IV. 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 V. 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 VI. 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.

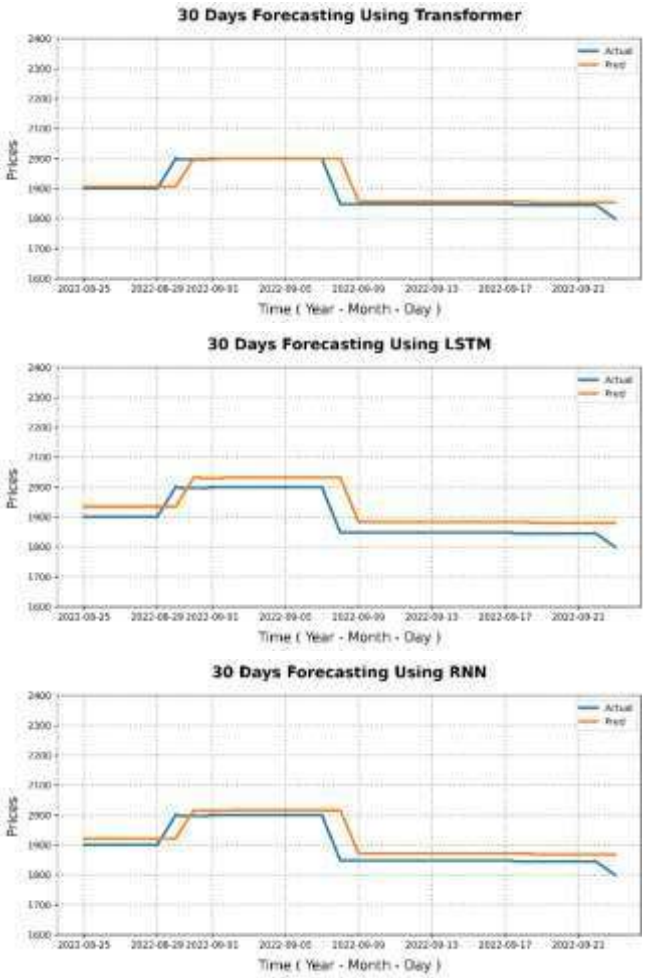



Fig. 11. Forecasting visualization for 30 days of transformer, LSTM, and RNN respectively

According to the visualization results shown in Fig 11 where is shown the comparison the actual and predicted results of different models, showing that the transformer model has a much smaller margin of error compared to the RNN and LSTM models. This suggests that using transformers for forecasting is more accurate and effective than using RNN or LSTM models and by looking at the numerical comparison in Table IV Table

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v12i1.1569, Copyright ©2023
Submitted : January 17, 2023, Revised : March 6, 2023, Accepted : March 17, 2023, Published : March 27, 2023

V and Table VI the Transformer model appears to be more accurate than the LSTM and RNN models in predicting the target variable. This is happened, due to its optimized layer structure. Moreover, the self-attention mechanism which allow transformer to give ever input tokens a weight and will output the best weight across several input tokens. Overall, these results indicate 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.

## IV. CONCLUSION

The main goal of this paper is to investigate the ability of the Transformer model to predict daily GPU Price over periods of 8, 16, and 30 days. To do this, we used a dataset of daily prices of NVIDIA RTX 3090 Founders Edition, and applied the Transformer, RNN, and LSTM models to predict the prices based on the past 2 years of data. Our results showed that the Transformer model was the most accurate in predicting the prices of the GPU, with higher CC, lower RMSE, and lower MAPE values 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. For further research, 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. Lastly, by changing other parameters can lead to a better result.

## REFERENCES


[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] S. S. Pal and S. Kar, "Time series forecasting using fuzzy transformation and neural network with back propagation learning," *Journal of Intelligent and Fuzzy Systems*, vol. 33, no. 1, pp. 467–477, 2017, doi: 10.3233/JIFS-161767.

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

[12] A. Vaswani *et al.*, "Attention Is All You Need," Jun. 2017.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] S. R. Venna, A. Tavanaei, R. N. Gottumukkala, V. v. Raghavan, A. S. Maida, and S. Nichols, "A Novel Data-


p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v12i1.1569, Copyright ©2023
Submitted : January 17, 2023, Revised : March 6, 2023, Accepted : March 17, 2023, Published : March 27, 2023

Driven Model for Real-Time Influenza Forecasting," *IEEE Access*, vol. 7, pp. 7691–7701, 2019, doi: 10.1109/ACCESS.2018.2888585.

[18] 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.

[19] 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.

[20] 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.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v12i1.1569, Copyright ©2023
Submitted : January 17, 2023, Revised : March 6, 2023, Accepted : March 17, 2023, Published : March 27, 2023