


Sovereign Gold Bond Trend Line Prognostication using Multi-Headed Attention based Transformer

Sakshi Karande

Bachelor of Engg, Dept. of AI & DS
Vidyavardhini's College of Engineering and Technology
MH, India
sakshi.213409202@vcet.edu.in

Raunak Joshi 

Assistant Professor, Dept. of AI & DS
Vidyavardhini's College of Engineering and Technology
MH, India
raunak.joshi@vcet.edu.in

Tej More

Bachelor of Engg, Dept. of AI & DS
Vidyavardhini's College of Engineering and Technology
MH, India
tej.213519101@vcet.edu.in

Neha Raut

Assistant Professor, Dept. of AI & DS
Vidyavardhini's College of Engineering and Technology
MH, India
neha.raut@vcet.edu.in

Abstract—In this paper, a deep learning approach based on transformers is introduced for the purpose of predicting gold prices. The approach utilizes a thorough dataset spanning from 2013 to 2023. The initial stages of the study involve preparing the data through normalization and structuring temporal sequences. The dataset is then divided into training and test sets, with the model being trained on the former and assessed on the latter using metrics such as Mean Absolute Percentage Error (MAPE) and accuracy. The proposed transformer model is specifically tailored to capture the complex temporal relationships present in financial time series data. It consists of multiple layers of self-attention mechanisms and is composed of stacked multi-head self-attention layers, followed by feed-forward neural networks and layer normalization. The model's training is carried out using the Adam optimizer with a batch size of 32 over 50 epochs. The model's robust predictive performance is evidenced by experimental results, with a MAPE of 0.01055 and a test set accuracy of 0.9894. These metrics demonstrate the model's ability to accurately forecast gold prices, surpassing traditional time series forecasting techniques. Additionally, the training process displays convergence, achieving a final Mean Squared Error (MSE) of around 0.0015 on the training data, indicating successful learning. Visual representation of model predictions compared with actual prices and baseline models further confirms the transformer model's capability to capture long-term dependencies and complex patterns in gold price fluctuations. These results highlight its potential usefulness in financial forecasting, providing valuable insights into market trends and investment strategies.

Index Terms—component, formatting, style, styling, insert

I. INTRODUCTION

The sovereign gold structure functions as a crucial economic indicator and a safe-haven and stable asset, which expects precise price forecasting in the financial sector. The gold price prediction model [1] [2] was designed with usability and accessibility in mind, ensuring that researchers and practitioners can easily use and adapt the solution for various forecasting jobs. The modular design of the model's architecture and training process allows for easy customization and adaptation to various datasets and scenarios. The approach encourages code

reusability and maintainability by separating each component into distinct modules, allowing users to add new features or alter current ones without causing the system to malfunction.

Gold price prediction has traditionally relied on a variety of statistical and machine learning models, including ensemble methods, Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, Seasonal ARIMA (SARIMA), Autoregressive Integrated Moving Average (ARIMA), and Convolutional Neural Networks (CNNs). While these models have shown differing degrees of success, they often struggle to accurately capture the intricate patterns and long-term dependencies present in financial time series data. This challenge is compounded by the multifaceted nature of gold prices, which are influenced by economic, geopolitical, and market factors with complex interrelationships over time.

Transformer models, characterized by their use of self-attention mechanisms, represent a significant advancement in deep learning that initially transformed natural language processing tasks. These models excel in processing and understanding sequences of data, making them well-suited for tasks requiring the capture of complex relationships over extended periods. Recently, there has been increasing interest in applying Transformers to time series forecasting, leveraging their scalability and parallel processing capabilities to potentially enhance predictive accuracy and efficiency.

This research focuses on evaluating the effectiveness of Transformer models specifically for forecasting gold prices over a decade-long period spanning from 2013 to 2023. The study employs rigorous comparative analysis, benchmarking Transformer models against traditional deep learning architectures such as RNNs, LSTMs, CNNs, Bi-CNNs, Bi-LSTMs, and established time series models like ARIMA/SARIMA. Evaluation metrics, notably the Mean Absolute Percentage Error (MAPE), are used to objectively assess predictive performance. The findings highlight the Transformer model's potential to outperform traditional methods in capturing the

nuanced dynamics of gold price movements, offering insights into its applicability and advantages in financial forecasting contexts.

II. METHODOLOGY

A. Data collection and Preprocessing

The gold price prediction model was created using a 2584x7 dataset covering data from 2013 to 2023 from Kaggle database. The dataset includes daily gold prices associated with its date in addition to related characteristics like open, high, low, volume (Vol.) and change percentage (Change percentage).

First step being importing the dataset, the dataset goes through a thorough feature engineering process to remove any inconsistencies or anomalies that could jeopardize the quality of analysis. This involves thoroughly reviewing each data item, including identifying and dealing with missing values, as well as identifying and removing redundant or superfluous columns. Initially, the dataset included columns like 'Vol.' and 'Change percent' that were irrelevant to our research and were thus eliminated to simplify the dataset. By removing these unnecessary columns, the dataset is simplified, improving the efficiency of subsequent analyses.

TABLE I
GOLD PRICES DATASET (HEAD)

Date	Price	Open	High	Low
12/30/2022	1,826.20	1,821.80	1,832.40	1,819.80
12/29/2022	1,826.00	1,812.30	1,827.30	1,811.20
12/28/2022	1,815.80	1,822.40	1,822.80	1,804.20
12/27/2022	1,823.10	1,808.20	1,841.90	1,808.00
12/26/2022	1,809.70	1,805.80	1,811.95	1,805.55

Following data cleansing, the temporal dimension of the dataset is highlighted by changing the 'Date' column, which was originally saved as a string, was transformed to date-time format with the `pd.to_datetime(df['Date'])` method from pandas. This translation was required to enable chronological sorting and proper indexing, both of which are critical for retaining the data's temporal order. By describing time in a uniform way, the dataset becomes more suitable for time-based analysis, which improves the efficacy of subsequent modeling efforts. The dataset was sorted in ascending order by the 'Date' column to guarantee that the time series data was properly aligned.

Furthermore, numerical columns holding pricing information go through a vital transformation to ensure numerical consistency and compliance of the upcoming visualisation step. Specifically, the 'Price' column, like other related columns, included values formatted with commas as thousand separators. The commas were eliminated using the 'replace' method with a regular expression viz. a straightforward pattern to remove commas from numeric strings, which is mathematically given as:

$$x' = \sum_{i=1}^n c_i \quad \text{for } c_i \in x \quad \text{where } c_i \neq ','$$

Here, x is the original string, x' is the transformed string, c_i are the characters of the string x , n is the length of the string x , and the summation \sum denotes the concatenation of characters that do not include commas.

After the commas were removed, the columns were transformed to a floating-point data type with ('float64') as type. This conversion guaranteed that the numerical data was properly formatted for future analysis and modeling. Converting to 'float64' allows the data to accommodate a wide range of numerical values with high precision, which is especially important for financial datasets like gold prices, where minute fluctuations can have serious consequences. The conversion to 'float64' reduced potential issues like data type inconsistencies and computational errors, improving the reliability and efficacy of subsequent data analysis tasks.

To prepare the dataset for analysis, we first determined the size of the test set for the year 2022. This involved counting the number of rows in the dataset whose date fell within 2022. The dataset was then divided into two separate sets: the training set, which included data from 2013 to 2021 and 2023, and the test set, which included data from 2022 onward. Each set was divided into DataFrames that included date and gold price information. These DataFrames were then saved as CSV files to ensure reproducibility and data integrity in subsequent stages of modeling. This thorough preprocessing ensured that the dataset was correctly formatted and ready for further analysis and predictive modeling.

B. Model Construction

Our goal in developing the model for predicting gold prices is to create a strong framework capable of capturing various patterns and relationships that exist in the data. Our model is built around the concept of windowing. Unlike RNN and LSTM, we were following step-by-step data capturing, instead here we define the input layer, specifying the shape of input as a window which has a sequential history of previous gold prices, and the matching output reflects the next price in the temporal sequence. By arranging the data in this way, we allow the model to learn from past patterns and make accurate forecasts about future pricing. This entails a methodical approach to creating and deploying a predictive model that uses advanced deep learning techniques specifically tuned to the features of time series data. To begin building the transformer-based model for gold price prediction, the dataset is subjected to data preprocessing processes required for sequential data preparation (section A.). Initially, the numerical features representing gold prices are normalized with MinMaxScaler to ensure that all values are within a standardized range. This normalization step is critical because it prevents a single feature from dominating the model training process due to its size, allowing for more effective learning across all features and is mathematically calculated as,

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (1)$$

where X represents the original data, X_{\min} and X_{\max} denote the minimum and maximum values of X , respectively. This transformation ensures that all features are scaled to a range between 0 and 1.

Following normalization, the dataset is divided into consecutive input-output pairs using a sliding window algorithm. The input sequence 'X train' contains historical gold prices across a window size of 60 days according to Date, with the corresponding goal output 'y train' set to the price point that follows the input sequence. This segmentation prepares the data for efficient training in the transformer architecture, allowing the model to learn from past pricing patterns.

Initially, the model accepts input sequences consisting of 60 consecutive time steps, each representing the gold price at that time step. Positional encoding is applied to these input sequences, incorporating positional information into the input data and enabling the model to retain the sequential order of the time steps, which is crucial for capturing temporal patterns. The transformer model design begins with an embedding layer, which creates a unique representation for each price point. This layer converts each input sequence element to a high-dimensional vector space, called embedding dimensions (d model).

The transformer model is intended to effectively capture long-range relationships in sequential data, making it appropriate for forecasting time series such as gold price thus it begins with initializing important components, including the embedding dimension (d model), which affects the dimensionality of input and output embeddings. This dimension is critical because it defines the space in which the input tokens (previous prices) and positional encodings (time steps) are embedded. Positional encoding, which uses positional information into input tokens. It introduces sinusoidal functions with variable frequencies and phases, which allow the model to distinguish between points in a sequence. The positional encoding function ensures that tokens are embedded with positional information reflecting their order in the sequence, which improves the model's capacity to recognize temporal relationships, mathematically represented as,

$$\text{PosEncoding}(pos, i) = \begin{cases} \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), & \text{if } i \text{ is even,} \\ \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), & \text{if } i \text{ is odd,} \end{cases} \quad (2)$$

where $\text{PosEncoding}(pos, i)$ represents the positional encoding for a position pos and dimension i , and d_{model} denotes the embedding dimension of the transformer model. This encoding scheme introduces sinusoidal functions of different frequencies to encode the positional information of the input sequences. The transformer architecture consists of numerous encoder layers. Each encoder layer has two sub-layers: multi-head self-attention and position-based feedforward networks. Self-attention is a mechanism that calculates attention scores for different points in the input sequence. The self-attention mechanism generates attention scores that reflect relationships

between distinct points in the input sequence. The algorithm uses query (Q), key (K), and value (V) matrices to linearly modify the input embeddings. The attention scores are generated using scaled dot-product attention, which analyzes the similarity between tokens at different places weighted by their relevance.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3)$$

where Q , K , and V are matrices representing the query, key, and value derived from the input sequence, respectively. d_k is the dimensionality of the key vectors.

The softmax function computes the normalized weights across the rows of the scaled dot-product matrix $\frac{QK^T}{\sqrt{d_k}}$, ensuring that the attention weights sum up to 1. This mechanism allows the transformer model to assign weights to each key-value pair based on their relevance to the query, facilitating effective capture of dependencies in the input data. Furthermore, dropout regularization is used after each sublayer (self-attention and feedforward network) to prevent overfitting. Dropout randomly masks a subset of input units during training, causing the model to acquire more robust features by preventing it from depending too much on certain tokens or positions.

$$\text{Dropout}(x) = \begin{cases} \frac{x}{1-p}, & \text{with probability } 1-p \\ 0, & \text{with probability } p \end{cases} \quad (4)$$

where x denotes the input to the dropout layer and p is the dropout rate, typically set to 0.1 in transformer models as seen in the code (`tf.keras.layers.Dropout(0.1)`).

This technique helps improve generalization by preventing the model from relying too heavily on specific input features during training, thereby enhancing its ability to generalize to new data. Subsequently, layer normalization is applied to the outputs of the attention layer. This normalization ensures that the outputs have a mean of zero and a standard deviation of one, stabilizing the training process and maintaining consistent gradient flow through the network. The normalized outputs are then fed into a second multi-head attention layer, which refines the sequence representation by focusing on additional dependencies in the data. Another round of dropout and layer normalization follows, ensuring the stability of the training process and further reducing the risk of overfitting.

Finally, the output from the last normalization layer is passed through a dense layer with a single unit. This dense layer, which applies a linear transformation, processes the refined sequence representation to generate the predicted gold price for the subsequent time step. Throughout this process, from positional encoding to the final dense layer, the model effectively captures and leverages the temporal dependencies in the gold price data, enabling it to make accurate predictions based on historical trends.

C. Transformer Framework:

In this paper, the transformer framework is used to forecast gold prices, exploiting its ability to handle sequential

input and capture complex relationships over time. The self-attention mechanism is crucial to the transformer design, since it computes attention scores to determine the value of each token in a sequence in comparison to others. This method allows the model to successfully learn from past price trends and incorporate pertinent information into future projections. Multi-head attention improves the model's capacity to focus on many components of the input sequence at once, allowing for richer representations.

To handle the sequential nature of time series data, positional encoding is used. This approach uses sinusoidal functions to embed positional information directly into the input embeddings, guaranteeing that the model can understand the order and context of each price observation. Each transformer encoder layer has two major sub-layers: multi-head self-attention and feedforward neural networks (FFN). The FFN consists of two fully connected layers with a ReLU activation function, allowing the model to learn complex patterns in the input. Dropout regularization and layer normalization are built into each sublayer to improve generalization and training stability. Overall, the transformer model is trained to minimize the Mean Squared Error (MSE) loss between projected and real gold prices, which is optimized using the Adam algorithm. This comprehensive methodology enables the model to accurately anticipate gold prices using historical patterns and current market conditions.

D. Model Training

The model is compiled with the Adam optimizer and the Mean Squared Error (MSE) loss function. The Adam optimizer was chosen because of its ability to efficiently handle sparse gradients and alter learning rates adaptively, which is essential for training deep learning models with vast parameter spaces. The MSE loss function is used to calculate the average squared differences between projected and real gold prices, so giving a clear statistic for directing the optimization process and assessing prediction errors, the focus is on iteratively modifying the model's parameters to reduce the difference between predicted and real values.. This stage guarantees that the model is optimized to learn successfully from the training data by reducing prediction errors.

Training proceeds for 50 epochs, with each epoch representing a full pass over the training dataset. As training advances, the model's performance usually increases, as evidenced by a reduction in the loss function. During model training, input sequence is handled in batch size of 32, each with 60 time steps (window size).

This batch processing enables efficient computation and gradient updates. For each batch, the model executes a forward pass (feed forward network), which involves passing the input sequences through the transformer model's multiple layers, as shown in Figure.1 to generate predictions. It begins with the compilation of the neural network model. This compilation phase initiates the learning process by specifying how the model will learn from the data. The training data is made up of windowed sequences of time series, with each sequence

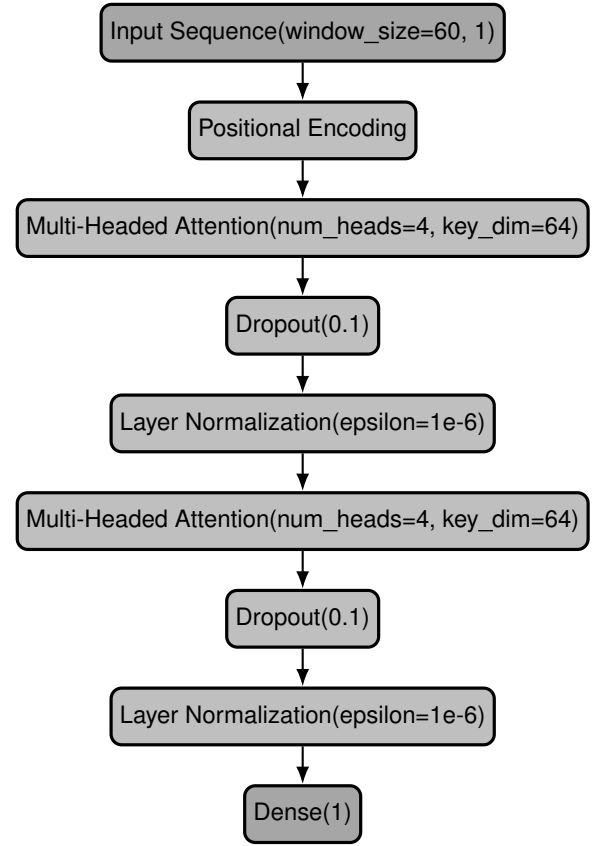


Fig. 1. Transformer Model Architecture

serving as an input feature and the next value in the time series representing the target label. These sequences are divided into X-train and Y-train, which allow the model to learn patterns from the data. The MSE loss is then derived by comparing these forecasts to actual gold prices for the relevant time steps, hence determining the model's prediction accuracy.

$$\text{MSE}(\hat{y}, y) = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad (5)$$

where \hat{y} represents the predicted values, y denotes the true values, and n is the number of samples. In the provided code, the model is trained using MSE loss with the Adam optimizer:

`model.compile(optimizer='adam', loss='mse')` This loss function quantifies the average squared difference between predicted and true values, providing a measure of the model's performance during training. It is minimized during training to improve the model's ability to accurately predict gold prices.

Following the loss computation, backpropagation is utilized to calculate the gradients of the loss in relation to the model parameters. These gradients show how much each parameter should be modified to minimize the loss. The Adam optimizer uses these gradients to update model parameters, specifically weights and biases, in a way that minimizes loss. This optimization stage is critical since it refines the model to better fit the training data.

This procedure is carried out for each batch in the training set to ensure that every sample in the dataset contributes to the learning process. The entire dataset is run through 50 epochs, with each epoch representing a complete pass through the training data. This iterative approach enables the model to learn and generalize from past data, hence enhancing its capacity to properly anticipate future gold prices.

III. RESULTS

This section of the paper describes the comparison and impacts of each model utilized previously, as well as how the transformer-based architecture outperforms them.

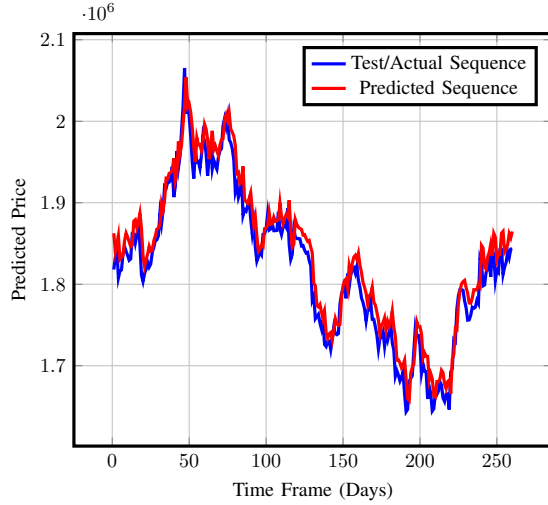


Fig. 2. Actual and Predicted Sequence Trend Visualization

The graph compares actual (Test/Actual Sequence) and predicted (Predicted Sequence) gold prices over time, evaluating a transformer-based model's forecasting accuracy and its implications for financial strategies.

A. Model Evaluation

The transformer-based model's performance was assessed using a gold price dataset spanning from 2013 to 2023 which underwent preprocessing to eliminate unnecessary columns and ensure proper formatting. Subsequently, it was split into training and test sets, with the test set containing data from 2022 and the rest as train set i.e 2013 to 2021 and 2023.

The primary evaluation metric for the model's predictions was the Mean Absolute Percentage Error (MAPE) because it offers a straightforward, percentage-based assessment of the prediction accuracy, which makes it simpler to understand the model's performance in relation to the actual values. MAPE is independent of scale, enabling easy comparison between various datasets and models. The model attained a MAPE of 0.01055, signifying an average error of slightly over 1 percent between the predicted and actual gold prices, representing highly accurate forecasting. The corresponding test accuracy, computed as 1MAPE, was 0.98945 i.e 98.945 percent, indicating the model's predictions were highly accurate and dependable.

1) *Training and Validation accuracy:* During the training process, the transformer model was trained for 50 epochs using a batch size of 32. The model's design consisted of several attention heads and layers, allowing it to capture complex relationships in the time series data. The chosen loss function was the Mean Squared Error (MSE), which is appropriate for regression tasks.

Throughout the training period, the model consistently reduced its loss, demonstrating its effective learning from the training data. The initial training loss of around 0.1852 decreased to 0.0015 by the final epoch. Simultaneously, the validation loss, which assesses the model's performance on unseen data, decreased from 0.0092 to 0.0005. The decrease in both training and validation losses indicates that the model not only captured the patterns in the training data but also generalized effectively to new data.

B. Model Validation and Comparison

In order to validate the model, we assess the predicted values on the test set by comparing them with the actual gold prices and then created a visualization to effectively showcase the model's performance. The visualization displays two distinct lines: the actual test data (blue), and the predicted test data (red). The remarkably close correspondence between the red and blue lines illustrates that the transformer's predictions closely mirrored the actual prices, indicating its high level of accuracy.

The performance surpasses that of traditional models like ARIMA or basic LSTMs, which frequently face challenges in handling long-term dependencies and intricate temporal patterns in financial time series data. The multi-head attention mechanism of the transformer enables it to simultaneously focus on various segments of the time series, effectively capturing subtle patterns and trends. This feature empowers the transformer model to reduce prediction errors and achieve reliable performance. Moreover, conventional models are often constrained by their linear assumptions and basic structures, which constrain their capacity to grasp the complex dynamics of financial markets. The transformer model surpasses these traditional methods by using its advanced architecture to manage complex and long-term relationships in the data of gold prices. The findings highlight the superior capability of the transformer to outperform traditional models significantly, demonstrating its superiority and potential for practical financial forecasting applications.

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

- [1] S. Shafiee and E. Topal, "An overview of global gold market and gold price forecasting," *Resources policy*, vol. 35, no. 3, pp. 178–189, 2010.

- [2] T. Brabenec, P. Suler, J. Horák, and M. Petras, "Prediction of the future development of gold price." *Acta Montanistica Slovaca*, vol. 25, no. 2, 2020.