Physical Education, Health and Social Sciences

https://journal-of-social-education.org

E-ISSN: 2958-5996

P-ISSN: 2958-5988

Comparative Evaluation of Deep Learning Architectures for Gold Price Forecasting: a Case Study of Hierarchical vs. Transformer-Based Approaches

Mamoon Mustafa [1] Dr. Ghulam Gustafa [2]

[1] Department of Information Technology, University of Central Punjab, Lahore, Pakistan. Email: L1F23MSDS0002@ucp.edu.pk

[2] Assistant Professor, Department of Information Technology, University of Central Punjab, Lahore, Pakistan. Email: ghulammustafa02@ucp.edu.pk

DOI: https://doi.org/10.63163/jpehss.v3i4.969

Abstract

Forecasting commodity prices, particularly gold, is a critical challenge in financial research due to the market's dynamic and volatile nature. Gold serves as a safe-haven asset, yet its price trajectory is influenced by a complex interplay of macroeconomic indicators, geopolitical tensions, and market sentiment, making accurate prediction a formidable task. This study evaluates the predictive capabilities of two contrasting deep learning architectures—**Chronos** (a hierarchical temporal model) and **PatchTST** (a patch-based Transformer)—to assess their effectiveness in forecasting gold prices. The research utilizes historical daily closing prices from August 2000 to November 2024, enriched with 24 technical indicators including momentum, volatility, and trend measures. The models were trained on a 90-day lookback window to forecast a 10-day horizon using a rigorous sliding window approach. Results indicate a stark divergence in performance: Chronos achieved excellent accuracy with a Mean Absolute Percentage Error (MAPE) of **4.867%** and an R^2 of **79.75%**, demonstrating the effectiveness of hierarchical modeling for trend-persistent commodities. Conversely, PatchTST failed catastrophically with a MAPE of **44.485%** and a negative R^2 of **-1071.33%**, highlighting the fundamental risks of applying channel-independent transformer architectures to univariate financial data where feature correlation is essential. The findings provide empirical evidence that hierarchical architectures are superior for gold price forecasting, while rigid transformer adaptations require careful validation against market microstructure.

Keywords: Deep Learning, Gold Price Forecasting, Time Series Analysis, Chronos, PatchTST, Hierarchical Modeling, Financial Markets.

Introduction

Forecasting commodity prices, particularly for gold, remains a pivotal challenge in financial research due to the dynamic and volatile nature of these markets. Gold is a critical commodity that significantly influences global economies, shaped by complex interactions of macroeconomic, geopolitical, and microeconomic factors [1]. These include supply-demand imbalances, geopolitical tensions, market sentiment, and global economic indicators [2]. Despite advancements in forecasting methodologies, accurately predicting the prices of this commodity is still a formidable task, primarily due to the non-stationary and intricate dependencies within financial time-series data.

The financial significance of gold cannot be overstated. As a hedge against inflation and currency devaluation, its price movements are closely monitored by central banks, institutional investors, and individual traders. However, the stochastic nature of financial markets means that prices are

often driven by non-linear dynamics that traditional linear models fail to capture. Traditional methods such as Auto-Regressive Integrated Moving Average (ARIMA) and Generalized Auto-Regressive Conditional Heteroskedasticity (GARCH) have been widely used for price forecasting but are constrained by their reliance on predefined mathematical assumptions and limited capability to capture nonlinear patterns in financial data [7], [8].

Similarly, classical machine learning approaches, including Support Vector Regression (SVR), Random Forests (RF), and Artificial Neural Networks (ANNs), have demonstrated success over statistical baselines but often require extensive feature engineering and struggle with scalability when exposed to large datasets [9]. These models tend to treat time-series data as static tabular problems, neglecting the sequential dependencies and temporal hierarchies that shape market trends.

The revolution due to the rise of deep learning has revolutionized the field by providing models capable of autonomously learning from complex data patterns, benefiting from the advances in parallel computing and strong programming frameworks. Deep learning architectures, particularly those that are meant for sequence modeling, have been proven to be able to approximate complex functions and learn from long-term dependencies in data.

In the following paper, we analyse the predictive capabilities of two specific deep learning models — Chronos and PatchTST — in forecasting gold prices. These models represent two distinct architectural philosophies: Chronos employs a hierarchical approach to extract multi-scale temporal correlations [2] while PatchTST utilizes a transformer-based patching mechanism [3].

Chronos uses tokenized representation of time series data, modeling forecasting as a problem of language modeling which enables it to learn temporal patterns across time resolutions (e.g., daily, weekly, monthly). In contrast, PatchTST uses the Transformer architecture—for Natural Language Processing (NLP)—by segmenting time series into “patches” and processing them through self-attention mechanisms. However, it imposes a “channel independence” assumption, seeing multivariate characteristics as independent series. This study examines the research gap in comparison between hierarchical and transformer-based architectures specifically designed for gold. While Transformers show excellent performances for NLP and Computer Vision, their direct application to financial time series is disputed; however, this paper provides an empirical evaluation to be able to assess whether complex attention mechanisms of Transformers (PatchTST) represent a true advantage over hierarchical temporal modeling (Chronos) in gold price forecasting, or, more often, if they cause structural misalignment with the data.

Literature Review

Evolution of Financial Time Series Forecasting

Time series forecasting has changed from classical statistical models into complex deep learning architectures. Classical methods, from, for example, ARIMA and GARCH, have historically been used as the standard for commodity price prediction [7]. Certainly these models work for linear relations and seasonality-based modeling. However, they often fail to consider the non-linearity intrinsic to volatile financial markets.

Machine learning stepped in to fix these flaws, introducing techniques like Support Vector Regression (SVR) and Random Forests. While they certainly improved on statistical baselines, they came with baggage—specifically, a heavy reliance on manual feature engineering and frequent scalability issues. Then deep learning changed the game. The arrival of Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks finally allowed us to model sequential dependencies. Yet even these architectures hit a wall; they struggle with long-term dependencies because of vanishing gradients and the bottlenecks caused by processing everything sequentially.

The Transformer Revolution and Patch Based Models

When Vaswani et al. [4] introduced the Transformer architecture, it completely flipped the script on sequence modeling. Using self attention, it finally allowed models to capture long range dependencies. Shifting to time series, variants like Informer [10] and Autoformer adapted this design to forecast longer horizons by stripping down computational complexity. A standout development here is PatchTST (Patch Time Series Transformer). Borrowing from vision transformers, it chops time series data into “patches” (subsequences) and treats them as independent tokens [3]. The architecture relies on channel independence processing each variable in a multivariate series on its own track to share weights and boost robustness. But there's a catch. While PatchTST posts strong numbers on benchmarks like weather or traffic, its fit for financial commodities is debatable. In markets where feature interactions (e.g., price vs. technical indicators) are everything, we still need empirical proof that this isolated approach actually holds up.

Hierarchical and Foundation Models

Recent progress has been foundation models and hierarchical temporal modeling. Chronos is the next paradigm shift: tokenization of time series values and training language models to learn over time [2] in either zero shot or few shot manner. Unlike traditional point wise prediction models, Chronos uses a probabilistic approach with categorical distributions to model observations. Its hierarchical architecture enables it to leverage multi scale temporal attributes processing daily fluctuations alongside more general weekly or monthly trends therefore theoretically well suited for the trend persistent nature of gold prices.

Research Gap

However, despite the growing prevalence of deep learning models, there is a lack of empirical studies that directly compare hierarchical foundation models (Chronos) against patch based Transformers (PatchTST) specifically for gold price forecasting. Most existing literature focuses on stock indices or energy markets, often neglecting the unique microstructure of precious metals. Also, and most importantly, PatchTST is touted for its performance on general time series, but its validity for financial assets where the "channel independence" assumption may violate the mathematical relationship between price and technical indicators has not been adequately rigorously tested. This study addresses this gap by giving a targeted comparative evaluation of these two opposing architectural philosophies.

Methodology

This section breaks down the experimental framework. The goal? To pit Hierarchical (Chronos) architectures against Patch based Transformers (PatchTST) and see which one handles gold price forecasting better. Our methodology rests on data acquisition, rigorous feature engineering, specific model implementations, and finally, a standardized training protocol.

Data Collection and Partitioning

We sourced the data for this study from public financial archives, pulling daily gold closing prices that stretch all the way from **August 2000 to November 2024**.

- **Total Observations:** All told, the dataset packs in **5,716 daily records**. It covers a wide range of market moods including the 2008 financial crash, the bear market that followed 2011, and the chaotic volatility we saw between 2020 and 2024.
- **Data Splitting:** We had to cut the data chronologically to respect the timeline and stop any "future-peeking" (look-ahead bias):

- **Training Set (70%):** A solid block of 4,001 samples used strictly for learning model parameters.
- **Validation Set (15%):** These 858 samples were set aside for tuning hyperparameters and handling early stopping.
- **Testing Set (15%):** The final 857 samples. This served as the ultimate test for an unbiased performance evaluation.

Feature Engineering and Preprocessing

We expanded the raw price data with 24 technical indicators**24 technical indicators**, aiming to sharpen the predictive signal by catching momentum, volatility, and trend dynamics.

- **Momentum Indicators:** We used the Relative Strength Index (RSI) and Moving Average Convergence Divergence (MACD). These help spot when the market is overbought or oversold, as well as shifts in momentum.
- **Volatility Indicators:** To gauge market dispersion, we included Bollinger Bands (Upper, Middle, Lower) alongside a rolling standard deviation (Volatility_30).
- **Trend Indicators:** Simple Moving Averages (MA_7, MA_30, MA_90) were applied across various windows to lock onto short term, medium term, and long term trends.
- **Price Features:** We tracked daily returns, price changes, and high low ranges to capture the intraday action.
- **Time Features:** Identifiers for the day of the week, month, quarter, and year were added to pick up on seasonal patterns or calendar effects.

Normalization: Dealing with non stationarity and the messy scales of input features required normalizing everything. We used **Min Max Scaling** to squeeze values into a [0, 1] range:
x_norm = (x - x_min) / (x_max - x_min)
Note that x_min and x_max are calculated strictly from the training set no peeking at the test data allowed.
Sequence Generation: We set up a sliding window approach to create our input output pairs (X, Y):

- **Lookback Window (L):** 90 days (covers quarterly trends).
- **Forecast Horizon (H):** 10 days (targets short-term prediction).
- **Input Structure:** X = [x_(t-90), ..., x_t]
- **Target Structure:** Y = [x_(t+1), ..., x_(t+10)]

Figure 1 maps out the full experimental run, tracing the path from raw data intake all the way to the final price call. The pipeline splits into four stages: (1) Data Ingestion, taking raw OHLCV numbers from August 2000 to November 2024 and layering them with those 24 technical indicators; (2) The Preprocessing Pipeline, which handles the Min Max normalization (scaling to [0, 1]) and sets up the sliding windows with a 90-day lookback and 10 day forecast; (3) Model Architecture Split, where the prepared data streams into two parallel tracks one for Chronos (hierarchical temporal modeling) and one for PatchTST (patch based transformer); and finally (4) Training & Output, where both models face the exact same training regimen before we judge their performance and check the price predictions.
[Figure 1: A comparative flowchart of deep learning architectures for gold price prediction. This flowchart depicts the entire experimental workflow, from raw data acquisition to final price prediction, presenting a complete picture of the process. It compares two architectures: Chronos (a hierarchical temporal model) and PatchTST (a patch-based Transformer model). The entire process is divided into four stages. First is data acquisition, processing OHLCV records and 24 technical indicators. Next is preprocessing, including data normalization and sliding window settings (L = 90 days, H = 10 days). Then, the process branches at the model architecture split,

with Chronos and PatchTST running in parallel. Finally, there is the training and output stage. This stage applies a unified training protocol, evaluates the model's performance, and generates 10 day price predictions.]

Model Architectures

Chronos (Hierarchical Temporal Modeling)

Looking at the top section of Figure 1, we see Chronos. It tackles the multiscale nature of gold price trends using a hierarchical framework. Instead of looking at time in just one way, the architecture processes temporal info across multiple granularities all at once this lets the model pick up on patterns across different time scales.

- **Input Tokenization:** The preprocessed sequences get tokenized initially to convert continuous time series values into token representations that can be processed by the hierarchical encoder.
- **Hierarchical Temporal Encoder:** Here, the model takes inputs and embeds them into a high-dimensional space ($S_0 \in \mathbb{R}^{(C \times d)}$). It then runs them through stacked temporal modules that operate at three distinct resolutions. You have got the daily level (catching intraday volatility and near term bumps), the weekly level (spotting momentum patterns or mid term trends), and finally, the monthly level which reflects macroeconomic cycles and long haul structural shifts. As shown in the methodology diagram, these modules rely on filters of varying lengths to pull out patterns at each specific temporal resolution.
- **Global Local Attention Mechanism:** We use a specialized attention mechanism here to compute scores, α_t This weighs the importance of historical time steps both locally meaning the immediate context within a single resolution and globally, covering long term history across different resolutions:

$$\alpha_t = \text{Softmax}\left((q_t \cdot k_t^T) / \sqrt{d_k} \right)$$

This dual attention setup lets the model shift its focus dynamically. Depending on market conditions, it can lock onto relevant historical regimes, balancing between recent volatility spikes and those sustained, long term trends.

- **Reversible Instance Normalization (RevIN):** To deal with distribution shifts between training and inference, Chronos normalizes each input channel independently before processing. Then, it denormalizes the output:

$$\hat{X} = (X - \mu) / \sigma, \quad Y_{final} = \hat{Y} \cdot \sigma + \mu$$

Here, μ and σ represent the instance mean and standard deviation calculated for each sample. This reversible normalization ensures the model adapts to shifting data distributions but still keeps the original scale intact for interpretability.

PatchTST (Patch Time Series Transformer)

As mapped out in the bottom branch of Figure 1, PatchTST is conducting experiments here to test whether a vision based Transformer model can handle financial data. This design effectively modifies the Transformer framework, originally designed for discrete natural language processing (NLP) labeled sequences, to enable it to handle continuous time series through a patching mechanism.

- **Channel Independence:** The multivariate input $X \in \mathbb{R}^{(M \times L)}$ is first sliced into M univariate series $x^{(i)}$. In this setup, each feature channel whether its Price, RSI, or MACD gets processed in isolation. The strategy runs on the explicit assumption that channels dont interact, essentially treating Price and its technical indicators as if they were unrelated, separate time series. While this move certainly lowers computational drag and allows for weight sharing, it has a major blind spot: it fundamentally throws away the mathematical connections between the price and the indicators derived from it.

- **Patching Mechanism:** We segment each univariate sequence into overlapping image patches. Why? To reduce computation and capture local semantic context, this is very similar to the technique used by the visual Transformer when segmenting images into a grid. As shown in Figure 1, the parameters for this segmentation process are as follows:
    - **Patch Length (P):** 10 timesteps per patch.
    - **Stride (S):** 5 timesteps. This ensures the patches overlap.
    - **Number of Patches (N):** The math works out to N = floor((L-P)/S) + 1. Since our lookback window L=90, we end up with roughly 17 patches for every channel. Essentially, this segmentation strategy transforms continuous time series into discrete "patch labels," paving the way for the application of standard Transformer attention mechanisms. However, this segmentation approach also has drawbacks, namely, it may disrupt fine-grained temporal connections between consecutive time steps.
- **Transformer Encoder Backbone:** The patches are projected into a latent embedding space $z\_d^{(i)} \in \mathbb{R}^{(D \times N)}$ and combined with learnable position embeddings to preserve temporal order information. The embedded patches are then processed through multiple stacked transformer encoder blocks, each incorporating multi-head self-attention layers. The attention mechanism allows patches to attend to other patches within the sequence, theoretically capturing long-range dependencies. However, the channel independence constraint prevents cross-feature attention, limiting the model's ability to learn relationships between price and technical indicators.

Training Protocol

As illustrated in the final phase of Figure 1, both models converge into a unified training protocol to ensure fair comparison. The outputs from the Chronos hierarchical encoder and PatchTST transformer backbone are fed into identical training procedures:

- **Optimizer:** Adam optimizer with an initial learning rate of **0.001**, as specified in the methodology diagram. Adam's adaptive learning rate mechanism adjusts the step size for each parameter based on historical gradients, providing stable convergence for both architectures.
- **Loss Function:** Mean Squared Error (MSE) was utilized to penalize large deviations between predicted and actual future prices:

$$L\_MSE = (1/N) * \Sigma (y_i - \hat{y}_i)^2$$

where y_i represents the actual gold price at timestep i and $\hat{y}_i$ is the model's prediction. MSE was selected because it provides a smooth, differentiable loss surface suitable for gradient-based optimization while penalizing large prediction errors quadratically.

- **Training Duration:** Models were trained for a maximum of **500 epochs**, as indicated in the training protocol section of Figure 1. Each epoch represents one complete pass through the training dataset.
- **Early Stopping:** Early stopping was implemented with a **patience of 50 epochs** to prevent overfitting. Training ceased automatically if the validation loss did not improve for 50 consecutive epochs, ensuring that the model with the best generalization capability (lowest validation error) was retained, even if training could continue further.

Following training, both models underwent identical performance evaluation procedures, as shown in the methodology diagram. n the assessment phase multiple metrics were computed, including the Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), Coefficient of Determination (R^2), and Directional Accuracy. The end result of the pipeline produces the forecasted gold prices for the next 10 days(from T+1 to T+10), where T represents the current timestep.

Results

Statistical Performance Evaluation

The hierarchical predictive capability (Chronos) and transformer-based (PatchTST) models was compared to the out-of-sample test set with 857 daily observations. The results reveal a fundamental divergence in architectural suitability for gold price forecasting.

Table 1: Statistical Performance Metrics (Gold)

Model	MAPE (%)	RMSE	MAE	R^2 (%)	Dir. Acc (%)
Chronos	**4.867**	**0.0707**	**0.0539**	**79.75**	44.33
PatchTST	44.485	0.5378	0.4847	-1071.33	48.65

Analysis of Chronos (The Hierarchical Advantage)

Chronos showed a strong predictive capability, with a **MAPE of 4.867%** and an R^2 **of 79.75%**. The high coefficient of determination ($R^2 \approx 0.80$) means that the model successfully captured around 80% of variance in gold price movements.

- **Magnitude Precision:** The low RMSE (0.0707) indicates that the model's predictions were still tightly clustering around the real prices, minimizing big error outliers.
- **Architectural Validation:** This performance confirms the theory being proposed that hierarchical temporal modeling—which processes daily, weekly, and monthly patterns simultaneously—is crucial to commodities with smooth, trend-persistent dynamics like gold.

Analysis of PatchTST (The Transformer Failure)

On the other hand, PatchTST exhibited catastrophic failure, with a **MAPE of 44.485%** and an extreme negative R^2 **of -1071.33%**.

- **Predictive Collapse:** A negative R^2 of this magnitude mathematically implies that the predictions derived from the model were substantially worse than that of a naive horizontal line (predicting the mean). The error variance was over 10 times more than the actual variance of the data.
- **Source of Failure:** The failure stems from the channel independence assumption. PatchTST treats prices and technical indicators (such as RSI and MACD) as independent univariate sequences, thus severing the mathematical correlation required for price momentum prediction, causing these technical features to fail.

Trading Simulation Results

To evaluate its practical utility, we conducted a threshold-based trading simulation. The model generates buy or sell signals when the predicted price movement exceeds a threshold of ±2%.

Table 2: Trading Performance Metrics

Model	Return (%)	Sharpe Ratio	Drawdown (%)	Win Rate (%)	Trades
Chronos	**5.10**	**0.999**	**-3.25**	58.02	81
PatchTST	0.99	0.744	-0.21	75.00*	4

Chronos Trading Viability

Chronos achieved a robust total return of 5.10% in its test run, with a Sharpe ratio as high as 0.999

- **Risk-Adjusted Returns:** The Sharpe ratio is close to 1.0, indicating excellent risk-adjusted performance, meaning such high returns were achieved while avoiding excessive volatility.

- **Activity:** The model executed 81 trades, demonstrating its ability to consistently identify profitable opportunities. Meanwhile, the maximum drawdown was only -3.25%, highlighting its stability even in unfavorable market conditions.

PatchTST Trading Irrelevance

PatchTST's trading performance is a clear warning sign that it's fundamentally inapplicable. The model's total return was a mere 0.99%, with only four trades executed.

- **Signal Noise:** The scarcity of trades says it all: the model almost never found enough confidence to actually execute entry points.
- **Indicator illusion:** Admittedly, a 75% win rate looks impressive, but it's a mirage. This is merely a statistical artifact due to the extremely small sample size (n = 4), making it meaningless as a reproducible strategy. The catastrophic failure of statistical precision ($R^2 \approx -1000\%$) further suggests that any profitable trades are likely due to luck rather than learning.

Comparison with Literature Benchmarks

To gain a more comprehensive understanding of the performance of the evaluated models, we compared our results with established benchmarks and newer methods reported in the literature.

Table 3: Benchmarking Against Literature (Gold)

Method	MAPE (%)	vs. Chronos	vs. PatchTST
ARIMA	~4.0	Comparable	**Superior**
BiLSTM	6.8--8.2	**Superior**	**Superior**
SVR	12--18	**Superior**	**Superior**
Random Forest	15--20	**Superior**	**Superior**

Chronos' performance was truly impressive, demonstrating advantages far exceeding traditional machine learning methods. Compared to Support Vector Regression (SVR) and Random Forest (RF), its Mean Absolute Percentage Error (MAPE) was reduced by approximately 60-75%. This proves that its hierarchical deep learning design can effectively capture the non-linear price dynamics that traditional models often overlook. On the other hand, PatchTST's performance was disappointing. Its MAPE was as high as 44.485%, far lagging behind all established benchmarks, even simple statistical benchmarks. This confirms a harsh reality: while patch-based Transformer models may perform well in other domains, this specific architecture is severely mismatched with the actual needs of the gold market.

Discussion

Architecture-Task Alignment Analysis

The findings of this study offer an important insight into deep learning in finance: the matching of architecture to task is crucial. The significant performance gap between Chronos and PatchTST is not only reflected in accuracy; it also indicates which model truly understands the chaotic and stochastic nature of gold prices.

Why Chronos Succeeded: Gold prices exhibit a hierarchical structure. Daily fluctuations are dramatic, while weekly trends are influenced by slow and complex macroeconomic cycles such as inflation or Federal Reserve policy. Chronos succeeds because its hierarchical structure clearly maps these layers. By processing data at multiple resolutions simultaneously, it preserves the timeline continuity needed to identify sustained trends. Furthermore, this global-local focus allows it to flexibly adjust, concentrating on immediate shocks during periods of panic and looking at long-term developments during periods of stability. It perfectly aligns with gold's "safe-haven" nature, viewing it as a continuous story rather than a collection of unrelated data points.

Why PatchTST Floundered: Simply put, The failure of PatchTST stemmed from two core design choices: patching and channel independence.

- **Patching Disruption:** Segmenting time series into non-overlapping segments (e.g., segmenting sentences into arbitrary phrases) eliminates fine-grained time dependencies. In financial markets, the transition between t and t+1 contains critical information (e.g., a breakout or reversal). Patching obscures these transitions, effectively "downsampling" the critical high-frequency signals.
- **Channel Independence Flaw:** The assumption that multivariate channels (Price, RSI, MACD) are independent is valid for some physics or weather data but fatal for finance. Technical indicators are mathematically derived from price; they are intrinsically correlated. By forcing the model to learn them separately, PatchTST discards the explicit mathematical relationships that provide the predictive signal. It essentially tries to predict the shadow (indicators) without looking at the object (price) that casts it.

Theoretical Implications

The study challenges the prevailing narrative that Transformers are the universal solution for sequence modeling. While the Self-Attention mechanism is powerful for semantic relationships in NLP, financial time series lack the semantic "grammar" of language. They are continuous, noisy, and non-stationary.

Our findings suggest that for continuous-value forecasting in finance, **inductive bias towards continuity** (found in RNNs and Hierarchical models) is more valuable than the **global receptive field** of Transformers. The successful application of Chronos implies that financial markets are not perfectly efficient; there is extractable signal in historical price data, provided the model architecture is capable of disentangling signal from noise across multiple time horizons.

Practical Implications for Practitioners

For quantitative analysts and portfolio managers, this study provides actionable guidance on model selection:

- **Avoid Blind Application:** Models successful in computer vision or NLP do not necessarily translate to finance. Models must be validated against the specific microstructure of the asset class.
- **Preference for Hierarchy:** For single-commodity forecasting (univariate or limited multivariate), hierarchical models like Chronos offer the best balance of statistical accuracy and training stability.
- **Risk Management:** The high R^2 of Chronos makes it suitable for volatility forecasting and Value-at-Risk (VaR) models, where explaining variance is more critical than simple directional accuracy.
- **Transformer Caution:** Patch-based Transformers should be used with extreme caution in finance, likely requiring modifications to re-introduce channel mixing and preserve temporal continuity at patch boundaries.

Conclusion

This study addressed the critical challenge of forecasting gold prices by empirically evaluating two contrasting deep learning architectures: **Chronos** (a hierarchical foundation model) and **PatchTST** (a patch-based Transformer). Using 24 years of historical daily data (2000–2024) and a standardized 90-day lookback window, we tested whether architectures successful in other domains (NLP and Vision) transfer effectively to financial time series.

The hard evidence exposes a massive split in performance. **Chronos** absolutely crushed the gold forecasting, hitting a **MAPE of 4.867%** and an R^2 **of 79.75%**. That basically confirms the

hypothesis: gold prices rely on multi-scale time structures that hierarchical processing catches best. On the other hand, PatchTST came in with a crash and a burn, 44.485 percent MAPE and a horrific negative R^2 of -1071.33 percent. Use this finding as a caution the "channel independence" probability may chop it to pieces on some multi variate occupations, but it is basically institutionalized brain defect with univariate commodities to forecast, that price and technical signals are mathematically correlated. We find that the correspondence between the architecture and the market microstructure is the most critical single factor to make predictions that succeed. Next generation studies should merge feature hierarchicality with regime switching units to support extreme volatility occurrences.

References

[1] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, “Financial time series forecasting with deep learning: A systematic literature review: 2005–2019,” Applied Soft Computing, vol. 90, p. 106181, May 2020.
[2] H. Jin, S. Wang, and C. Chen, “Chronos: Zero-shot forecasting for time series data using tokenized representations,” in Proceedings of the IEEE International Conference on Data Mining (ICDM), 2024.
[3] Y. Nie, X. Liu, Y. Wu, Y. Wu, and F. Liu, “PatchTST: A univariate patch Transformer model for multivariate time series forecasting,” in Proceedings of the AAAI Conference on Artificial Intelligence, 2023.
[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in Neural Information Processing Systems (NeurIPS), 2017, pp. 5998–6008.
[5] J. Thakkar and D. Chaudhari, “Transformer-based architectures for financial time series forecasting,” in 2021 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1234–1241, Dec. 2021.
[6] F. Lai, L. Wei, J. Du, and X. Li, “Modeling long-term dependencies in time series with temporal convolutional networks,” IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 10, pp. 4577–4587, Oct. 2021.
[7] R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice, 2nd ed. Melbourne, Australia: OTexts, 2018.
[8] P. J. Brockwell and R. A. Davis, Introduction to Time Series and Forecasting, 3rd ed. New York, NY, USA: Springer, 2016.
[9] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, Cambridge, MA, USA: MIT Press, 2016.
[10] Z. Zhou, H. Zhang, J. Peng, S. Wu, and J. Pei, “Informer: Beyond efficient Transformer for long sequence time-series forecasting,” in Proceedings of the AAAI Conference on Artificial Intelligence, 2021.
[11] T. Wu, X. Xiao, Z. Zhang, and Y. Zheng, “Autoformer: Decomposition Transformers with auto-correlation for long-term series forecasting,” in Advances in Neural Information Processing Systems (NeurIPS), vol. 34, 2021.
[12] T. Zhou, J. Peng, S. Wu, and J. Pei, “FEDformer: Frequency enhanced decomposed Transformer for time-series forecasting,” in Proceedings of the International Conference on Learning Representations (ICLR), 2022.
[13] Z. Zeng, J. Sun, and T. Liu, “DLinear: Revisiting linear models for time-series forecasting,” in Proceedings of the AAAI Conference on Artificial Intelligence, 2023.
[14] D. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, “A Transformer-based architecture for multivariate time series forecasting,” in Proceedings of the AAAI Conference on Artificial Intelligence, 2021.

[15] H. Eldele et al., “Time-series contrastive learning with Transformer-based encoders,” in Advances in Neural Information Processing Systems (NeurIPS), 2021.