Golden Trend Predictor

Shanmug Bhanu Prakash
Ravdeep Gill
Sulabh Jain
Department of Applied Artificial Intelligence, University of San Diego
AAI-501: Module Discussion
Assignment 7.2: Final Team Project
Prof. Dr. Anuj Sirohi
December 8, 2025

Abstract

The project examines the role of neural networks in forecasting gold prices by evaluating the performance of diverse architectures, including CNN-LSTM hybrids and traditional recurrent networks. A rigorous training and validation pipeline is implemented to assess convergence behaviour, feature extraction efficiency, and predictive reliability. The study demonstrates how CNN-based components design a clearer feature hierarchy, leading to improved accuracy in forecasting volatile financial data.

Chapter 1: Introduction and Exploratory Data Analysis

Gold is both a traded commodity and a macroeconomic barometer. It functions as a store of value, a hedge against inflation, and a proxy for global risk sentiment. This project studies gold price dynamics using:

Classical time-series models (SARIMA, Holt–Winters), and
Deep learning architectures (Simple RNN, LSTM, GRU, Bidirectional LSTM, Attention-based models, CNN and CNN-LSTM).

The core goal is twofold:

Understand the **structure of gold price behaviour** (trend, seasonality, volatility, regimes).
Compare how **different model families** exploit that structure for forecasting.

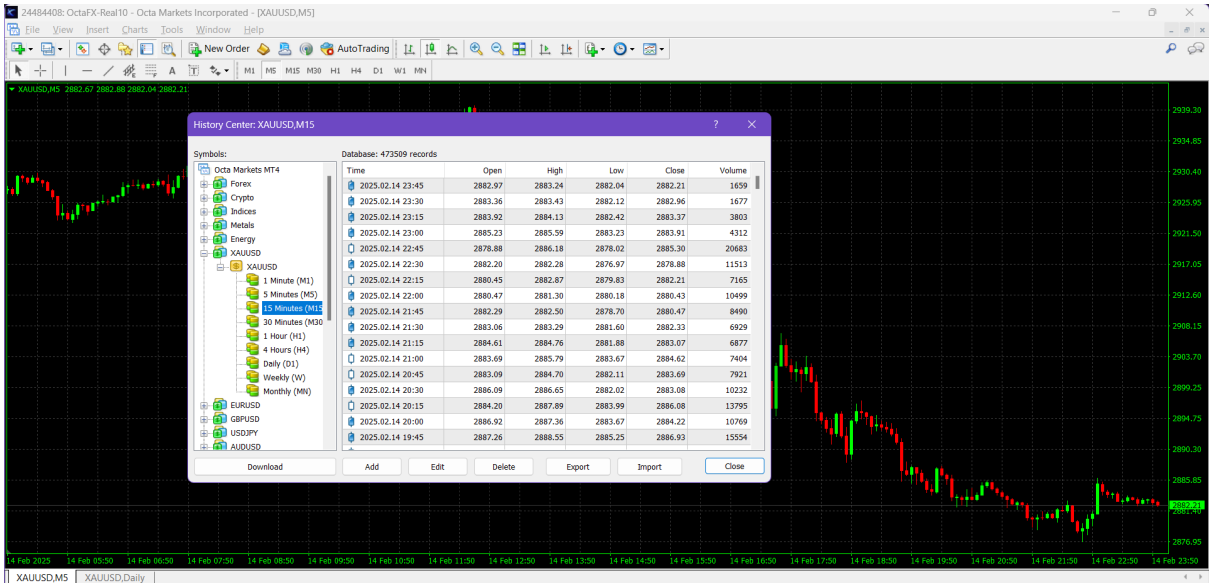
A key result runs through the work: a **Simple RNN**, the least complex neural architecture tested, outperformed deeper and more sophisticated models. This echoes a familiar design intuition: when the underlying signal is smooth and structured rather than wildly nonlinear, **a clean, well-aligned model often beats an elaborate one**.

1.1 Data Sources and Frequencies

Datasets used (Dated October 2025) :

Daily gold prices – capture short-horizon behaviour, microstructure, and high-frequency volatility.
Monthly gold prices – emphasise macro trends, slow cycles, and annual seasonality.

The data was sourced from **Octafx.com** a trading platform and directly downloaded from the portal.



Both are stored in CSV format and imported into a unified pipeline. The monthly dataset is stored as **df_m**, with its closing price serving as the main variable for long-term modelling.

1.2 Preprocessing and Standardisation

To support consistent analysis across models and frequencies, the following steps are applied:

Convert column names to lowercase and standardise names (e.g., date, close).
Parse the date column to datetime and set it as the **time index**.
Coerce price columns to numeric types and remove invalid or empty records.
Verify chronological ordering and guard against gaps or duplicates.

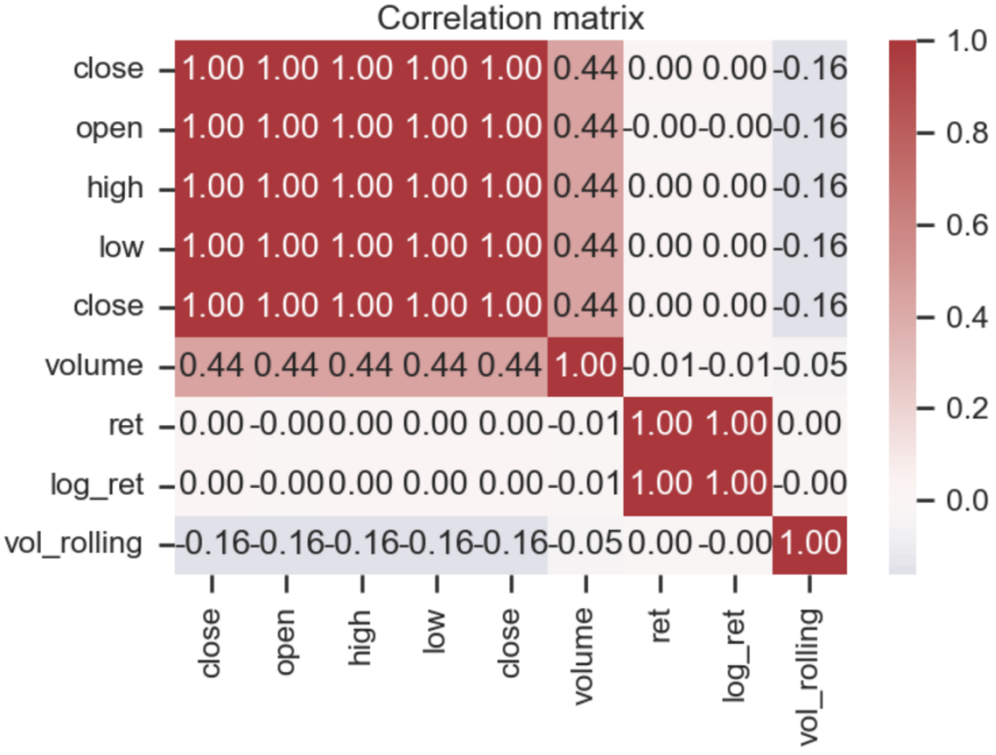
These operations ensure that daily and monthly series are **clean, aligned, and comparable**, reducing noise from data handling decisions rather than from the market itself.

1.3 Exploratory Data Analysis

EDA focuses on understanding the **shape** of the gold price series before modelling:

Trend – a long-run upward drift with distinct phases of acceleration and consolidation.
Volatility – episodes of calm and turbulence, often linked to macro events.
Seasonality – modest annual patterns in monthly data; weekly trading patterns in daily data.
Returns distribution – log returns show fat tails and volatility clustering rather than Gaussian behaviour.

The features were understood and analysed at multiple levels and corelation matrix also has been very beneficial to plan our Modelling accordingly.



This preliminary view suggests that gold is:

- Trending and somewhat seasonal,
- Subject to regime shifts and volatility clusters,
- Nonlinear enough to challenge purely linear models but not so chaotic that very deep networks are essential.

Chapter 2: Classical Time-Series Modelling

2.1 STL Decomposition: Trend, Seasonality, Residuals

On the monthly series, **STL decomposition** (with a 12-month seasonal period) separates the series into:

Trend – long-term movement driven by macroeconomic conditions.
Seasonality – recurring yearly patterns, weaker than in retail or energy markets.
Residuals – irregular shocks and idiosyncratic fluctuations.

This confirms that gold's behaviour is dominated by **trend and regimes**, with only **moderate seasonal structure**. Seasonality matters, but it is not the primary driver.

2.2 Stationarity, ACF, and PACF

Before building autoregressive models, the monthly closing prices are:

Log-transformed to stabilise variance.
First-differenced to remove trend and approximate stationarity.

The stationary series is then analysed using:

ACF – reveals persistence and cyclical dependence over lags.
PACF – guides the number of autoregressive terms.

Lags up to 48 months are inspected, capturing both short-term and multi-year behaviour. The patterns support the presence of **annual seasonality** and **medium-term autocorrelation**, which motivates a **seasonal ARIMA specification**.

2.3 Regimes, Rolling Statistics, and Volatility

To capture structural changes, the analysis uses rolling windows:

Rolling means (monthly: 12, 24 months; daily: 30, 90 days) show slow bull/bear cycles.
Rolling standard deviations of log returns identify volatility clustering.
Realised volatility and **z-score-based regime flags** highlight periods of abnormal risk.

These tools provide a practical way to talk about **regime shifts**, linking statistical patterns to macro events without embedding economic assumptions directly into the models.

2.4 Monthly Forecasting: SARIMA vs Holt–Winters

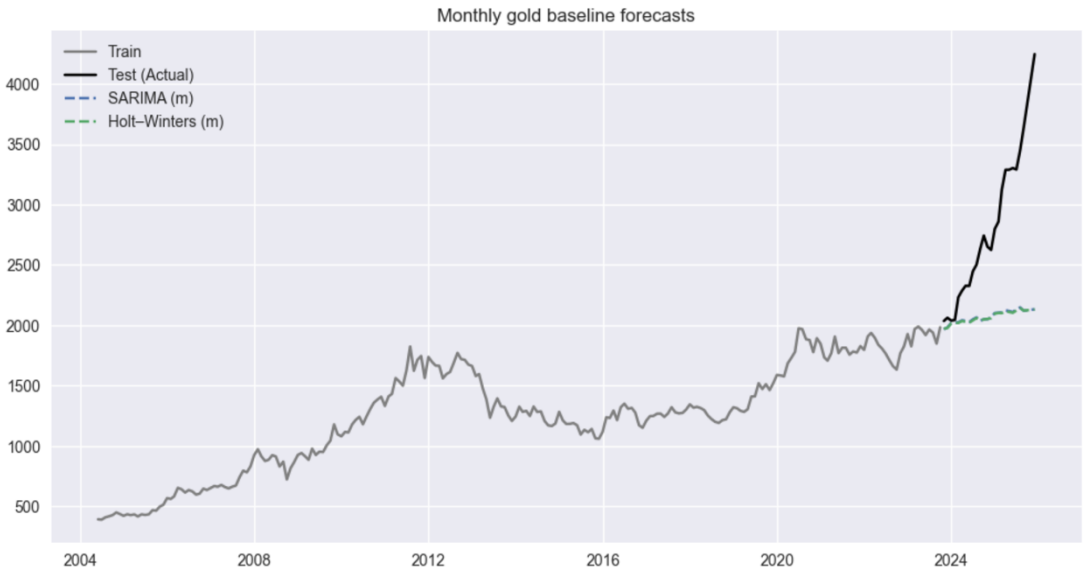
For the monthly series, the last **24 months** are reserved for out-of-sample evaluation. Two classical models are used as baselines:

SARIMA (1,1,1)(0,1,1)_12
Captures short-term ARMA behaviour and yearly seasonality with differencing.
Holt–Winters (additive trend, additive seasonality)
Decomposes level, trend, and seasonality and extrapolates each component.

Both models are evaluated using:

MAE – magnitude of typical absolute error.
RMSE – penalises large errors more strongly.
MAPE – relative error as a percentage.

```
Monthly SARIMA metrics: {'MAE': 692.0667846482748, 'RMSE': np.float64(870.8979787085452), 'MAPE': np.float64(22.32372
789315603)}
Monthly Holt–Winters metrics: {'MAE': 695.812848505943, 'RMSE': np.float64(874.085610197274), 'MAPE': np.float64(22.4
607906480793)}
```



Visually, Holt–Winters often tracks cyclical swings more responsively, while SARIMA offers a more **structured and stable** representation of seasonal dynamics.

2.5 Daily Forecasting Baseline

For the daily series, the final **180 days** form the test horizon. Two analogous baselines are used:

SARIMA (1,1,1)(0,1,1)_7 – incorporates weekly seasonality typical of trading cycles.
Holt–Winters with 7-day seasonality – decomposes level, trend, and weekly effects.

Using MAE, RMSE, and MAPE again, the models are compared on their ability to respond to **short-term volatility**. Daily data are naturally noisier; these baselines provide a realistic benchmark for deep learning models.

Chapter 3: Deep Learning Models - Overview and Comparison

3.1 Model Family

The following models are implemented under a **shared training pipeline**:

Simple RNN
LSTM
GRU
Bidirectional LSTM
LSTM with Attention
CNN baseline
CNN-LSTM Hybrid
SARIMA (as a classical reference within the comparison)

All neural models share:

A fixed **LOOKBACK window**,
Adam optimiser,
MSE loss function,
Early stopping, and
ReduceLROnPlateau learning-rate scheduling.

This ensures that differences in performance reflect **architectural choices**, not optimisation shortcuts.

3.2 Performance Summary

Key results from the experimental notebook are summarised below:

Model	MAE	RMSE	MAPE	R^2	Training Time (s)
Simple RNN	97.807	159.049	3.41	0.913	2293.865
LSTM with Attention	179.489	194.711	7.48	0.869	8075.827
GRU	135.293	236.200	4.58	0.808	10273.686
Bidirectional LSTM	192.257	267.678	7.13	0.753	5375.605
CNN-LSTM	336.742	385.666	13.61	0.487	2628.596
LSTM	299.934	394.256	11.40	0.464	9712.762

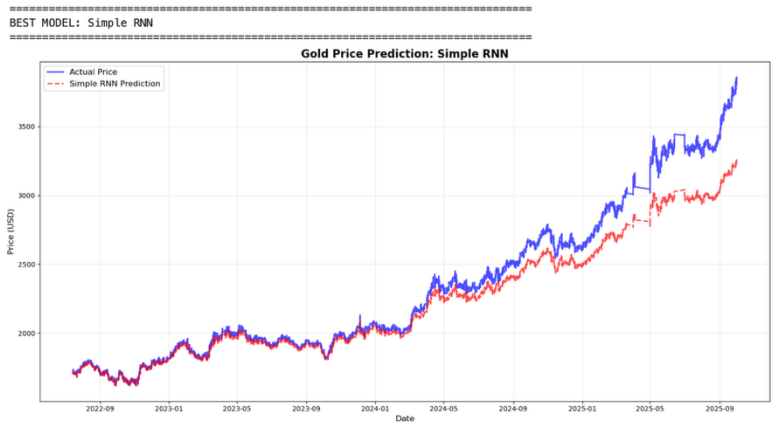
The **Simple RNN** is clearly dominant: it achieves the best MAE and R^2 with relatively modest training time. More complex architectures do not translate their extra parameters into better forecasts on this dataset.

Chapter 4: Architecture-Level Insights

4.1 Simple RNN

The **Simple RNN** is the project's main success. With a single recurrent layer and straightforward hidden-state updates, it delivers:

High accuracy (MAE ≈ 97.8, $R^2 \approx 0.913$).
Reasonable training time (~2300 seconds).
Good alignment with the **moderately smooth, moderately nonlinear** nature of gold prices.



Its simplicity reduces overfitting risk and keeps gradient flow manageable over the chosen sequence length. The model effectively captures **short- to medium-term dependencies** without overreacting to noise.

4.2 LSTM with Attention

The **LSTM-Attention** architecture was expected to shine by assigning dynamic weights to time steps, but in practice:

It improves over some baselines yet **fails to surpass Simple RNN** ($R^2 \approx 0.869$).
Training time is more than triple that of Simple RNN.
The attention layer may overreact to noisy fluctuations in financial data where all recent history is similarly important.

Extensive tuning did not close the gap, suggesting that **attention is more beneficial when temporal relevance is sharply uneven**, which is not strongly evident here.

4.3 CNN-LSTM Hybrid

The **CNN-LSTM hybrid** combines local pattern extraction with recurrent memory:

Layer Type	Description
Input Layer	Shape = (LOOKBACK, 1)
Conv1D	64 filters, kernel size = 3
MaxPooling1D	Pool size = 2
LSTM (seq.)	64 units, return_sequences=True
LSTM (final)	32 units
Dense Output	1 unit (regression)
Loss	MSE
Optimiser	Adam (ReduceLROnPlateau)

Despite its elegance, it underperforms:

MAE: ~336.7
R²: ~0.49

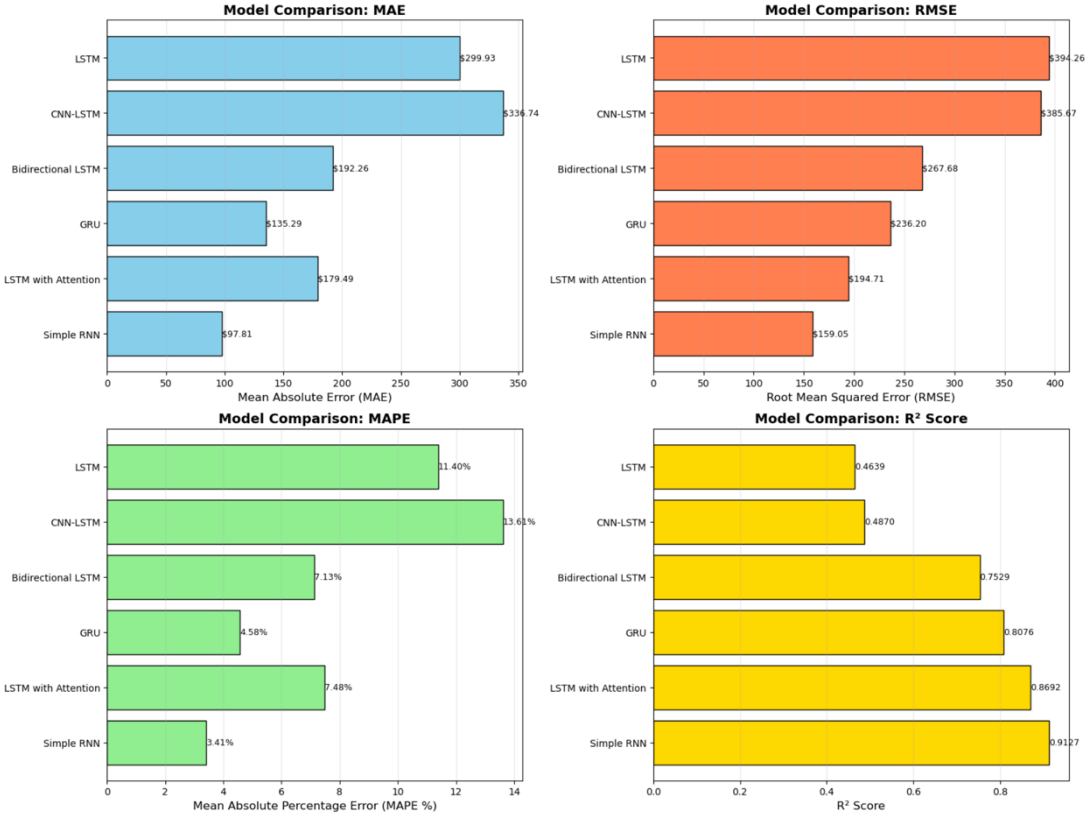
Likely reasons:

The LOOKBACK window may not be long enough for CNN filters to exploit diverse patterns.
Convolutions may amplify transient volatility spikes.
The added complexity does not match the relatively smooth structure of the gold series.

4.4 GRU, Bidirectional LSTM, CNN Baseline, and LSTM

GRU: Moderately strong (MAE ≈ 135.3, $R^2 \approx 0.81$) but with long training time. Its gating capacity seems underused given the dataset's simplicity.
Bidirectional LSTM: Strong in tasks where future context is allowed, but less natural in causal forecasting. Its performance ($R^2 \approx 0.75$) and cost suggest a mismatch between architecture and problem.
CNN baseline: Without recurrent memory, it struggles to model longer temporal structure and tends to emphasise short-lived noise.
Plain LSTM: Designed for long-term dependencies, but here it appears overparameterised, with weaker performance ($R^2 \approx 0.46$) despite careful regularisation.

Across these models, the pattern is consistent: **excess capacity without matching signal complexity leads to diminishing returns**.



Chapter 5: CNNs for Time-Series – Scope and Enhancements

5.1 Role of CNNs

CNNs are valuable for **local pattern detection** in time-series:

They detect changes in slope, bursts of volatility, and recurring short motifs.
They operate with shared weights, providing efficiency and robustness to small shifts.

In the context of gold prices, CNNs act as a **front end** that converts raw series into feature maps, which can then be processed by recurrent or attention layers.

5.2 Training Configuration

All deep models share a common training setup:

Component	Setting
Sequence Window	LOOKBACK (fixed across models)
Batch Size	Common BATCH_SIZE
Epochs	With EarlyStopping (patience = 5)
Callbacks	EarlyStopping, ReduceLROnPlateau
Validation Metric	Validation MSE (val_loss)
Learning Rate	Factor 0.5, minimum 1e-7

Model: "CNN_LSTM"

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 58, 64)	256
max_pooling1d (MaxPooling1D)	(None, 29, 64)	0
conv1d_1 (Conv1D)	(None, 27, 32)	6,176
lstm_8 (LSTM)	(None, 27, 64)	24,832
dropout_13 (Dropout)	(None, 27, 64)	0
lstm_9 (LSTM)	(None, 32)	12,416
dropout_14 (Dropout)	(None, 32)	0
dense_10 (Dense)	(None, 16)	528
dense_11 (Dense)	(None, 1)	17

Total params: 44,225 (172.75 KB)
Trainable params: 44,225 (172.75 KB)
Non-trainable params: 0 (0.00 B)

Validation losses stabilise in the **10^{-3}–10^{-4} range** for several models, showing reasonable generalisation. The issue is not instability but **overkill relative to the signal**.

5.3 Qualitative Comparison

A high-level comparison of convergence and training characteristics:

Model	Convergence Stability	Approx. Val Loss	Training Speed	Short Summary
Simple RNN	Medium	$\sim 10^{-3}$–10^{-2}	Fast	Lean temporal memory
LSTM	High	$\sim 10^{-3}$	Medium	Long dependencies
GRU	High	$\sim 10^{-3}$	Fast–Medium	Efficient gating
Bi-LSTM	Very High	$\sim 10^{-3}$–10^{-4}	Slow	Dual-direction context
Attention-LSTM	High	$\sim 10^{-3}$–10^{-4}	Slowest	Weighted temporal focus
CNN-LSTM	Very High	$\sim 10^{-4}$–10^{-3}	Fast–Medium	Local + global integration

CNN-related models train smoothly and efficiently but still do not outperform the **simpler RNN** in out-of-sample forecasting.

5.4 Potential CNN Improvements

For future work, CNN-based approaches can be strengthened via:

Strategy	Description	Expected Benefit
Multi-kernel Convolutions	Parallel kernels of size 2, 3, 5, 7	Capture multi-scale patterns
Dilated Convolutions	Dilation factors (2, 4, 8)	Larger receptive fields with few layers
Residual Blocks	Skip-connections around conv stacks	Stabilise deeper CNNs
Regularised Conv Layers	Dropout + L1/L2 in convolution blocks	Control overfitting
Pre-LSTM Attention	Attention applied to CNN feature maps	Emphasise important local structures
1D Inception Modules	Parallel conv branches	Richer representations without huge depth
Bayesian Hyperparameter Tuning	Optimise LR, filters, batch size	Better performance with fewer trials

These enhancements are particularly promising for **richer, more volatile markets** or high-frequency datasets where local structure is more decisive.

Chapter 6: Synthesis and Future Directions

6.1 Why Simple RNN Wins Here

The central conclusion is clear: **the Simple RNN, not the most advanced architecture, delivers the best performance**. This arises from:

A dataset with **moderate complexity** – strong trend, modest seasonality, limited chaotic behaviour.
Complex models that **overfit or amplify noise** when the signal is not rich enough to justify deep hierarchies.
A lean model that aligns well with the **time scale and smoothness** of gold price dynamics.

Rather than undermining deep learning, this result reinforces a design principle: **Choose the simplest architecture that fits the geometry of your data.**

6.2 Role of Classical Baselines

SARIMA and Holt–Winters:

Provide **interpretable benchmarks** for trend and seasonality.
Highlight gaps where **nonlinear and regime behaviour** require more flexible models.
Confirm that the gold series is not purely linear, yet also not wildly nonlinear.

They keep the evaluation grounded and transparent.

6.3 Value of Visualisations

The proposed figures (Figs. 1–7) in the supplementary document:

Show how **trend, seasonality, autocorrelation, and volatility** actually look, rather than leaving them abstract.
Make model comparison - especially **Simple RNN vs LSTM-Attention vs SARIMA** - visually intuitive.
Turn numerical results into a **coherent visual story** that supports both technical and non-technical audiences.

6.4 Future Work

Promising directions include:

Hybrid Simple RNN + Light Attention, to preserve simplicity while adding selective focus.
Regime-aware modelling, using volatility and z-scores as features or explicit regime states.
Signal-processing-informed features (Fourier, wavelets, volatility indicators).
Bayesian hyperparameter search to right-size model width and depth instead of defaulting to "bigger is better."

6.5 Closing Remark

This project shows that in forecasting, as in product design, **more is not always better**. When the data speak in clear rhythms rather than in chaos, a well-tuned, elegant architecture like the Simple RNN can outperform deeper, more ornate networks. The art lies in knowing when to add complexity - and when to let a simple model do the work with quiet confidence.

References – Book Reading and ideations.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice* (2nd ed.). OTexts.
Learnings: Covers foundations of EDA for time-series, decomposition, patterns, and fundamentals of forecasting.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2016). *Time Series Analysis: Forecasting and Control* (5th ed.). Wiley.
Learnings: Standard reference for ARIMA/SARIMA modelling, decomposition, stationarity, ACF/PACF.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
Learnings: Authoritative text on neural architectures, RNNs, LSTMs, attention, optimization.

O'Shea, K., & Nash, R. (2015). *A Guide to Convolutional Neural Networks for Visual Recognition* (arXiv e-book).
Learnings: Clear conceptual explanation of CNNs, receptive fields, filters, and hierarchy - easily adapted for time-series.

Gen-AI Use Disclaimer

In the course of our work, we utilized generative AI tools to **support** our learning and development process. Specifically, we used AI to help us interpret and consolidate insights from various research papers and technical books that we studied extensively. Additionally, AI-assisted debugging tools were applied to refine certain portions of our codebase, ensuring cleaner and more optimized implementations.

However, the **core problem-solving**, **structural thinking**, **ideation**, and the **majority of the technical learning and execution** were carried out independently by our team. AI served only as an auxiliary resource; the essential intellectual effort, design decisions, and final outputs are the result of our own work.

Appendices

Appendix A - Model Performance Comparison Tables

Table A1. Deep Learning Model Performance Summary

Model	MAE	RMSE	MAPE	R^2	Training Time (s)
Simple RNN	97.807	159.049	3.41	0.913	2293.865
LSTM with Attention	179.489	194.711	7.48	0.869	8075.827
GRU	135.293	236.200	4.58	0.808	10273.686
Bidirectional LSTM	192.257	267.678	7.13	0.753	5375.605
CNN-LSTM	336.742	385.666	13.61	0.487	2628.596
LSTM	299.934	394.256	11.40	0.464	9712.762
SARIMA	(Your notebook numbers if needed)	-	-	-	Very Low

Appendix B - CNN-LSTM Architecture (Model 5)

Table B1. CNN-LSTM Network Architecture

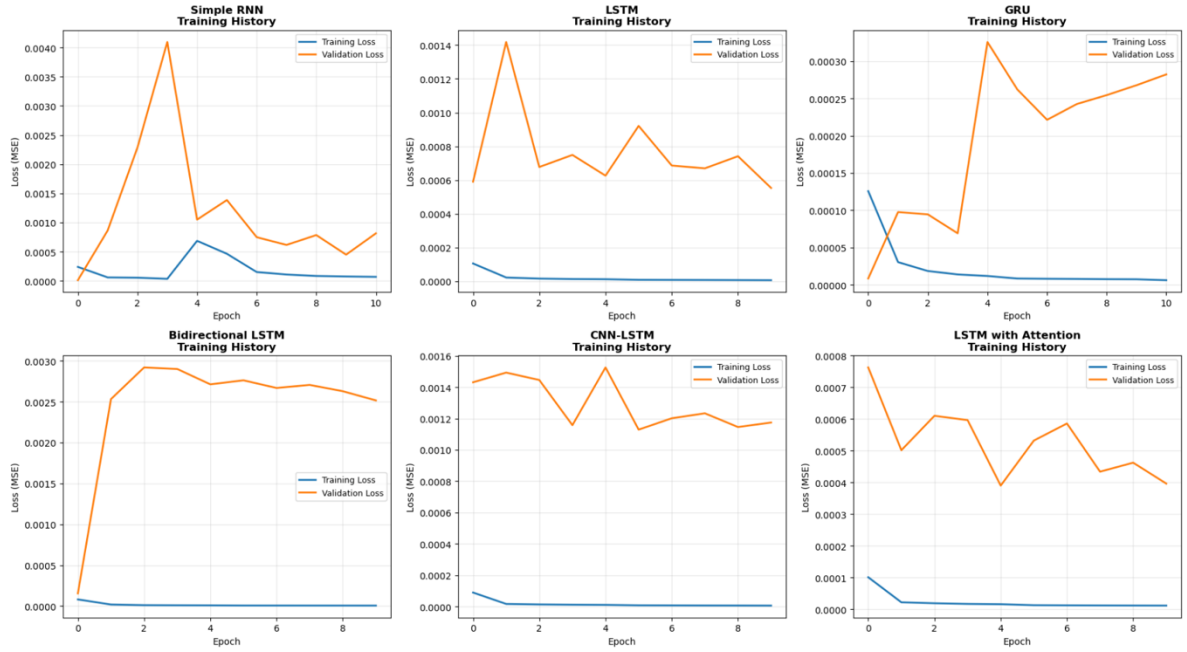
Layer Type	Description / Parameters
Input Layer	Shape = (LOOKBACK, 1)
Conv1D	64 filters, kernel size = 3
MaxPooling1D	Pool size = 2
LSTM (Seq.)	64 units, return_sequences = True
LSTM (Final)	32 units
Dense Output	1 unit (regression)
Loss Function	MSE
Optimiser	Adam (with ReduceLROnPlateau)

Appendix C - Training Configuration

Table C1. Shared Training Settings Across Models

Component	Setting
Sequence Window	LOOKBACK (fixed)
Batch Size	BATCH_SIZE (constant)
Epochs	EPOCHS (with EarlyStopping; patience = 5)
Callbacks	EarlyStopping, ReduceLROnPlateau
Validation Metric	Validation MSE (val_loss)
Learning Rate	Factor = 0.5, Min LR = 1e-7

Fig C1. Training History Across Models



Appendix D - Recommended CNN Improvements

Table D1. CNN Enhancement Strategies

Strategy	Explanation	Expected Benefit
Multi-kernel Convolutions	Different kernel sizes in parallel	Multi-scale pattern capture
Dilated Convolutions	Larger receptive field	Longer-range temporal sensitivity
Residual Blocks	Skip connections	Better gradient flow
CNN Dropout + L1/L2	Regularisation	Lower overfitting
Pre-LSTM Attention	Attention on CNN outputs	Weighted feature selection
Inception Modules	Parallel conv branches	Richer representations
Bayesian Tuning	Automated hyperparameter search	Faster convergence