Comparative Analysis of Neural Network Models for Foreign Exchange (FX) Forecasting

R, 42438 * **G, 50919** * **K, 40923** * **T, 46572** * **L, 41315** *

Abstract

The USD/JPY currency pair, one of the most liquid and widely traded financial instruments, presents forecasting challenges due to its sensitivity to both macroeconomic factors and short-term market microstructure effects. This study conducts a comparative analysis of deep learning architectures—including Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRUs), Transformer model (Encoding-Only), and a novel hybrid CNN-Transformer approach for multihorizon USD/JPY price prediction. Using historical data (2006–2024) augmented with technical indicators (e.g., Bollinger Bands, RSI), we evaluate each model's ability to generate actionable forecasts of direction at the 10-day horizons. The CNN-Transformer outperforms others, achieving a test AUC of 64% and validation accuracy of 65.9% at the 10-day horizon, leveraging local feature extraction and global dependency modelling. All models struggle with short-term (1-day and 5-day) forecasts due to market noise, with longer horizons yielding better performance. These findings highlight the efficacy of hybrid architectures for financial forecasting and suggest directions for improving short-term prediction performance.

1. Introduction

In the financial domain, time series analysis plays a crucial role in tasks such as stock price prediction, volatility modelling, and exchange rate forecasting. Although traditional statistical models such as ARIMA and GARCH have demonstrated their reliability in modelling short-term dependencies in financial time series, they often struggle to capture long-term dependencies or nonlinear relationships inherent in complex financial data (Zeng et al., 2023). To address these limitations, recent research has increasingly turned to neural network (NN) models, which offer greater flexibility in learning complex temporal patterns from data. Early developments in this area employed Recurrent Neural Networks (RNNs), along with their variants Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), which are designed to capture short and long-term dependencies in sequential data (Foumani et al., 2024). More recently, convolutional neural networks (CNNs) and Transformer-based architectures have been introduced to model both local patterns and long-range temporal relationships more effectively (Zeng et al., 2023).

Foreign exchange (FX) markets are among the most liquid and volatile financial markets, driven by macroeconomic factors, market microstructure effects, and short-term noise (Evans, 2011). Accurate forecasting of FX movements can spell the difference between profitable trades and costly missteps, helping investors and traders time their entry and exit points, yet it remains challenging due to the nonlinear and noisy nature of financial time series. Traditional models, such as autoregressive approaches, often struggle to capture complex temporal dependencies, while deep learning offers promising alternatives for modelling both short- and long-term patterns.

As mentioned before, recent advancements in deep learning, including Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRUs), and Transformer models, have shown potential in financial forecasting (Sezer et al., 2020). However, their comparative performance in FX markets, particularly for the USD/JPY pair, remains underexplored. Moreover, hybrid architectures combining convolutional neural networks (CNNs) with Transformers have emerged as powerful tools for time-series tasks but have not been extensively evaluated in FX contexts (Zeng et al., 2023). In this study, we'll address this gap by conducting a comparative analysis of LSTM, GRU, Transformer, and a novel CNN-Transformer hybrid for predicting directional movements of the USD/JPY exchange rate at 1-day, 5-day, and 10-day horizons. Through training and comparing the modern deep learning models, our goal is to identify which techniques give the clearest signals for making better trading decisions. Our methodology emphasises three innovations:

- **Architecture Benchmarking:** Direct performance comparison of temporal (LSTM/GRU) and attention-based (Transformer) models, isolating their strengths in capturing USD/JPY's price patterns.
- **Model Innovation:** Introduction of a CNN-

Transformer model that combines local feature extraction (via convolutional layers) with global dependency modelling to address both short-term noise and long-term trends.

- **Evaluation:** Metrics include training and test accuracy, together with AUC-ROC to assess economic viability.

Our results reveal that all models generalised better on the 10-day forward return horizon compared to the 1-day and 5-day horizons. The CNN-Transformer emerged as the top performer, achieving a validation accuracy of 65.9% and a test AUC of 64.0% on the 10-day forecast. This highlights the benefit of combining convolutional and attention mechanisms for FX time-series forecasting.

2. Dataset And Data Preprocessing

2.1. Dateset

The dataset, extracted from Yahoo Finance Python API, contains historical USD/JPY exchange rate data with daily records of Open, High, Low, Close, and Volume, spanning from 17 May 2006 to 30 December 2024. It was split into 80% for training, 10% for validation, and 10% for testing to ensure robust evaluation. The target variables is defined as binary indicators (1 for price increase, 0 for decrease) at each prediction horizon, allowing the models to capture temporal patterns in the USD/JPY exchange rate.

2.2. Feature Engineering

To enhance the model's predictive power, extensive feature engineering was applied, resulting in 30 features that capture various aspects of market behavior. Price-based features include lagged percentage returns (ranging from lags of 2 to 21 days) to reflect historical trends and raw price data (Open, High, Low, Close) for direct market signals. Technical indicators were computed to capture momentum and volatility, including the Relative Strength Index (RSI) for overbought/oversold conditions, the Moving Average Convergence Divergence (MACD) for trend shifts, the Commodity Channel Index (CCI) for cyclical patterns, the Simple Moving Average (SMA) over 10 days for trend smoothing, the Exponential Moving Average (EMA) over 10 days for more responsive trend tracking, and the Rate of Change (ROC) over 10 days to measure price momentum.

In addition, volatility and range indicators were incorporated to assess price fluctuations and potential breakouts, consisting of Bollinger Bands (upper and lower bands based on a 20-day moving average and 2 standard deviations) to identify volatility-driven price boundaries, the Average True Range (ATR) over 14 days to gauge average price movement, Keltner Channels (upper and lower bands based on a 20-day EMA and 2 times ATR) for trend-following range analysis, and Donchian Channels (highest high and lowest low over 20 days) to capture breakout levels. Additional risk and momentum metrics, including the Ulcer Index for drawdown risk by measuring the depth and duration of price declines, the Bollinger Band Width (difference between upper and lower bands) to assess volatility contraction or expansion, and the Keltner Channel Width (difference between upper and lower bands) were incorporated to provide a comprehensive view of market dynamics. Volume-based features were derived to evaluate buying and selling pressure, including On-Balance Volume (OBV) to track cumulative volume based on price direction, the Money Flow Index (MFI) over 14 days to measure money flow strength, the Chaikin Money Flow (CMF) over 20 days to assess volume-weighted buying/selling pressure, and the Volume Price Trend (VPT) to capture volume-driven price trends, all leveraging the Volume data where available.

2.3. Data Preprocessing

Due to activation functions and vanishing gradients, the RNN architectures such as LSTM and GRU are particularly sensitive to input scale (Pascanu et al., 2013). While the Transformer model can mitigate scale sensitivity through the use of Layer Normalisation (Vaswani et al., 2017), the pre-scaling inputs remains important for stabilising training and preventing exploding/vanishing gradients. Thus, considering the influence of scales of covariates contained from the section **Feature Engineering**, rather than directly fitting the four models, we employ the `MinMaxScaler` from the `sklearn.preprocessing` module. This scaling technique transforms each feature individually such that it lies within a specified range, typically $[0, 1]$. The transformation is given by:

$$\mathrm{x}_{\text{scaled}} = \frac{\mathrm{x} - \mathrm{x}_{\min}}{\mathrm{x}_{\max} - \mathrm{x}_{\min}}$$

where x is the original value, and $\mathrm{x}_{\min}$ and $\mathrm{x}_{\max}$ are the minimum and maximum values of the feature in the training data. This ensures that all features contribute equally during model training and prevents features with large numeric ranges from dominating the learning process.

3. LSTM Model

3.1. Literature Review

Financial time series forecasting has leveraged Long Short-Term Memory (LSTM) networks for directional exchange rate prediction, combining technical and macroeconomic indicators to improve accuracy. Yildirim et al. (2021) in "Forecasting directional movement of Forex data using LSTM with technical and macroeconomic indicators" propose a hybrid LSTM model that processes both data types separately, demonstrating superior performance in Forex directional forecasting—aligning with this work's focus on technical

indicators for USD/JPY. Their findings underscore the value of multi-source feature integration, though our preliminary results of using only technical indicators (AUC: 57.2% for 10-day predictions) suggest medium-term horizons remain more tractable than short-term ones. Similarly, Bao et al. (2017) in "A deep learning framework for financial time series" note LSTMs' limitations in noisy short-term FX forecasting, supporting our observed performance gap across horizons. Further, Sezer & Ozbayoglu (2018) in "Algorithmic financial trading with deep learning" highlight that technical indicators alone can achieve competitive results with LSTMs, motivating this study's streamlined approach despite omitting macroeconomic factors.

3.2. Model Architecture

The LSTM-based model takes in sequences of 3D tensors shaped as an input sequence of shape (500,30). 500 time steps were chosen for our LSTM model to recognize deeper patterns, cyclical behaviors, and volatility clusters that shorter sequences might miss. To improve generalization during training, a Gaussian Noise layer with a standard deviation of 0.01 is added. The architecture includes two Bidirectional LSTM layers: the first with 32 units preserving outputs at each timestep, and the second with 16 units that further compress the data for hierarchical feature extraction. A dropout rate of 0.3 is applied after each LSTM layer to mitigate overfitting.

To enhance the model's focus on relevant input portions, an attention mechanism is employed. This mechanism involves a `tanh`-activated dense scoring layer, which assigns importance weights across timesteps. These are normalized using a softmax function. The final context vector is computed as a weighted sum of LSTM outputs, effectively concentrating the model's capacity on salient patterns within the sequence.

The output layer is a single dense node with sigmoid activation, designed for binary classification of the directional return (up/down).

3.3. Training Methods

Training uses the Adam optimizer with a learning rate of 1×10^{-4}, along with gradient clipping to prevent exploding gradients. The loss function is binary cross-entropy, and evaluation metrics include accuracy and AUC—AUC being prioritized due to class imbalance in the dataset.

Early stopping is applied to halt training if the validation AUC does not improve after 30 epochs. Additionally, a learning rate scheduler reduces the learning rate by 50% if validation loss stagnates for two epochs, with a lower bound set at 1×10^{-6}. The training loop is capped at 50 epochs (subject to early stopping), with a batch size of 64. Class weights are incorporated to address label imbalance and penalize misclassification of rare events more heavily.

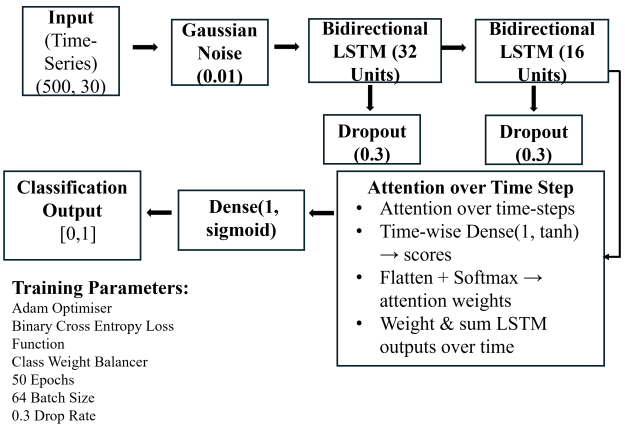

Figure 1. Architecture of the LSTM model

4. Gated Recurrent Unit (GRU) Model

4.1. Literature Review

Financial time series forecasting has seen significant advancements with deep learning, particularly through Gated Recurrent Units (GRUs). Chung et al. (2014) in "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling" introduced GRUs, showing they outperform LSTMs in sequence tasks with fewer parameters, making them ideal for capturing temporal dependencies in FX data. Sezer et al. (2020) in "Deep Learning for Financial Time Series Forecasting: A Review" highlight GRUs' success in financial forecasting when paired with technical indicators like RSI and MACD, supporting the approach of using features such as Bollinger Bands and ATR. In addition, Hossain et al. (2018) in "A Hybrid GRU-CNN Model for Financial Market Prediction" combined GRUs with CNNs, noting a 5% accuracy improvement by extracting spatial features from indicators before temporal modeling, inspiring this work's focus on robust feature engineering while maintaining a simpler GRU architecture for FX directional prediction.

4.2. Model Architecture

The model is a time series classification Gated Recurrent Unit (GRU) neural network designed to predict the binary classification (up/down) of future forward returns of the USD/JPY exchange rate at horizons of 1, 5, and 10 days. It leverages the GRU's ability to capture temporal dependencies in sequential data and processes a time series input of 10 timesteps, which is optimal for short-term forecasting. Each timestep includes 30 features (e.g., price data, technical in-

dicators), resulting in an input shape of (batch_size, seq_len, num_features). The input first passes through a GRU layer with 64 units that processes the sequence and outputs a sequence of the same length (return_sequences=True) to capture temporal patterns across all timesteps, with L2 regularization (coefficient 0.001) applied to mitigate overfitting by penalizing large weights. A Dropout layer with a 10% rate is then applied to further prevent overfitting.

The sequence then enters a second GRU layer with 32 units, which outputs a single vector (return_sequences=False) to summarize the temporal dependencies into a fixed-size representation, again with L2 regularization (coefficient 0.001) for stability. This is followed by another Dropout layer with a 10% rate and a Batch Normalization layer to normalize activations and stabilize training. A Dense layer with 16 units and ReLU activation is then applied to introduce non-linearity and learn complex patterns, with additional L2 regularization (coefficient 0.001) and a third Dropout layer (10% rate) for further regularization. Finally, a Dense output layer with a single unit and sigmoid activation produces a probability between 0 and 1, indicating whether the price will increase (1) or decrease (0). Overall, the GRU layers efficiently handle temporal dependencies with fewer parameters than LSTMs, while dropout, regularization, and batch normalization work together to prevent overfitting and ensure robust training on the USD/JPY time series data.

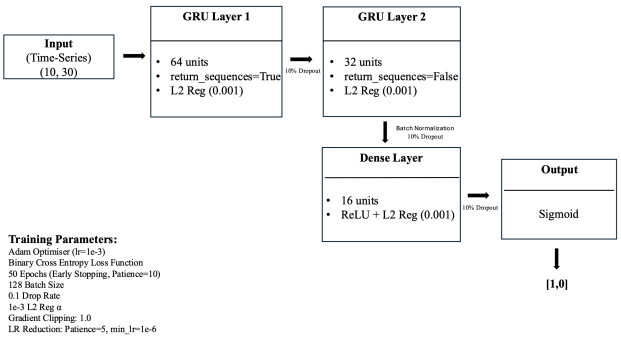

Figure 2. Architecture of the GRU model

4.3. Training Methods

The GRU model was trained to optimize binary classification performance for USD/JPY forward returns using the Adam optimizer with a custom learning rate schedule designed to enhance convergence stability. The learning rate schedule, inspired by adaptive strategies for recurrent neural networks, is defined as:

$$\text{lr}(t) = \frac{k}{\sqrt{d_{\text{hidden}}}} \cdot \min\left(t^{-0.5}, \frac{t}{\text{warmup\_steps}^{1.5}}\right)$$

where t is the training step, d_{hidden} is the hidden dimension of the first GRU layer (64 units), k is a scaling factor set to 0.1, and warmup_steps is set to 1000. This schedule incorporates an initial warm-up phase, where the learning rate increases linearly, followed by a decay phase to prevent overshooting and stabilize training.

Gradient clipping with a norm of 1.0 was applied to mitigate exploding gradients, a common issue in recurrent neural networks. The loss function was binary cross-entropy, suitable for the binary classification task, with performance monitored via accuracy and AUC to assess both correctness and discriminative power across thresholds. Training used a batch size of 128 to balance memory efficiency and gradient stability, spanning up to 50 epochs. Early stopping was implemented, halting training if validation accuracy did not improve for 10 epochs (patience=10), with the best model weights restored to prevent overfitting.

A ReduceLROnPlateau callback further adjusted the learning rate, halving it if validation loss stagnated for 5 epochs, with a minimum learning rate of 1×10^{-6}. Regularization included a 10% dropout rate after major layers and L2 weight penalties (coefficient 0.001) to reduce model variance, while batch normalization stabilized training by normalizing intermediate activations.

5. Transformation-Based Model

5.1. Transformer (Encoding-Only)

5.1.1. Literature Review

The Transformer model, introduced by Vaswani et al. (2017) in their seminal paper "Attention Is All You Need", revolutionised natural language processing (NLP) and machine learning. Unlike previous models relying on recurrent neural networks (RNNs) and convolutional neural networks (CNNs), which usually struggle with long-term dependencies (due to vanishing gradient problem) and limited receptive field (due to filter size), the Transformer discards sequential processing in favour of attention mechanisms. This shift significantly improves computational efficiency and scalability, enabling the processing of data in parallel and capturing long-range dependencies effectively. Besides producing major improvements in translation quality, it provides a new architecture for many other tasks, like time series forecasting. Notably, Transformer-based models such as the Informer (Zhou et al., 2021) and Autoformer (Wu et al., 2021) have extended the standard Transformer to specifically address the challenges of long sequence forecasting. These architectures incorporate mechanisms like sparse attention, decomposition blocks, and series-level representations to reduce complexity and improve forecasting accuracy. Overall, the wide applications of these models indicate the feasibility of time series forecasting using the

Transformer.

5.1.2. Model Architecture:

This model is a time series classification Transformer, which inherits the transformer architecture introduced by the paper "Attention Is All You Need" (Vaswani et al., 2017). Compared with the original Transformer architecture, the model only uses three Transformer Encoder blocks (6 encoder blocks in the original paper) consisting of Multi-Head Attention and a Feedforward Neural Network (FFN) with residual connections and normalisation, which is the same as the Encoder blocks used in the original. Considering our aim is to predict the binary classification (up/down) of the future forward return of the next day, we use only the encoder part of the original transformer (i.e. sequences to one model).

The model expects a time series input of 500 time steps, each with 30 features, i.e. of shape `(batch_size, seq_len, num_features)`. And the input will pass the dense layer that projects each timestep's features into a higher-dimensional space (i.e. `embed_dim`). Like the original Transformer, the Positional Encoding layer will add information about the position (order) of the sequence to the projected inputs (as self-attention treats inputs as a set, not a sequence). This layer implements fixed sinusoidal positional encoding for input sequences, i.e., the encoding for each position pos and embedding dimension i is defined as $PE_{\text{pos},2i} = \sin\left(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$ and $PE_{\text{pos},2i+1} = \cos\left(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$ where pos is the position (e.g., time step), i is the dimension index, and d_{model} is the embedding dimension (Vaswani et al., 2017).

Then the embedding sequences (together with the positional encoding) will cross the stacks of three Transformer Encoder blocks (one after the other). For each Transformer Encoder Block, it will start applying multi-head self-attention to the sequence. In this step, the model lets each position in the input refer to other positions, helping it understand relationships across time steps. Here, the self-attention of our model uses 4 heads, which means it splits the attention into 4 parallel subspaces before combining them. After the attention layer, the block passes the result will be fed into a Feed-Forward Network (FFN). The FFN consists of two kinds of dense layers. The first one expands the dimension to 128 units and applies a non-linear ReLU (rectified linear unit) activation, while the second projects it back to the original embedding size.

Throughout the blocks, residual connections are employed, which allow input to the attention and FFN sub-layers to be added back to their outputs, which helps the model avoid vanishing gradients. After each residual addition, the Normalisation Layer is applied to stabilise and speed up training (Rush, 2018). The GlobalAveragePooling1D layer will collapse the sequence dimension by averaging across the 500 time steps, resulting in a shape of `(batch_size, 30)`. After passing the Dropout layer (avoid overfitting), these sequences will match the input shape of the output layer, which uses the sigmoid function as the activation function, to forecast the forward days return label (i.e up/down).

5.1.3. Training Method

This model is trained to perform binary classification of forward returns using the Adam optimiser with a custom learning rate schedule inspired by the original Transformer architecture. The learning rate schedule is inspired by the Noam scheme (Vaswani et al., 2017), which is defined as:

$$\text{lr}(t) = d_{\text{model}}^{-0.5} \cdot \min(t^{-0.5}, t \cdot \text{warmup\_steps}^{-1.5})$$

where t is the training step, d_{model} is the embedding dimension, and `warmup_steps` is set to 4000. This schedule allows for an initial warm-up (speed up) period followed by gradual decay, helping stabilise training and prevent divergence in early epochs.

To address the class imbalance inherent in financial return classification data, we incorporated class weights during training. Additionally, rather than simply employ the `binary_crossentropy` as the loss function during the training, we customize a binary label smoothing loss function, where target labels were adjusted slightly away from 0 and 1 (e.g., 0.05 and 0.95), reducing the model's tendency to become overconfident and thereby improving generalisation.

The model was trained for up to 100 epochs with a batch size of 64, using early stopping with a patience of 13 epochs to prevent overfitting. The validation performance was monitored during training, and the best weights were restored at the end of training. Regularisation techniques such as L2 kernel penalties (with $1e^{-4}$ of regularisation factor) and dropout (rate $= 0.3$ used in the Dropout layer & rate $= 0.1$ inside the Encoding block) were also employed in the dense layers to further combat overfitting. These methods collectively enhanced training stability and improved the model's ability to generalise across unseen test data.

5.2. CNN-Transformer

5.2.1. Literature Review

Recent advances in financial time series forecasting have increasingly focused on combining deep learning architectures to capture both short- and long-term dependencies. The paper "Financial Time Series Forecasting using CNN and Transformer" (Zeng et al., 2023) proposes a hybrid model that leverages CNNs to extract local, short-term patterns

and Transformers to model long-range temporal dependencies in intraday stock price movements. This combination outperforms traditional statistical models like ARIMA and EMA, as well as deep learning baselines such as DeepAR, demonstrating improved directional prediction accuracy on US stocks. Similar CNN-Transformer hybrids have been explored successfully in fields like medical imaging, heart disease prediction, and environmental sensing, confirming the generalisability of this architecture across domains. However, despite their strong performance, these hybrid models face challenge like reduced interpretability, particularly important factors for deployment in sensitive areas like finance.

5.2.2. Model Architecture

For the CTTS model, it receives an input sequence of shape (500,30), where 500 is the temporal length equating to roughly 2-years of trading period and 30 representing the features which includes the technical indicators and historical price attributes, as mentioned earlier in the data segment. The CTTS architecture is particularly well-suited for financial applications due to its ability to model complex sequential dependencies and adapt to variable-length trends in time series data.

The model used in our study is a modified version of the architecture proposed in the JP. Morgan AI Research Paper (Zeng et al., 2023). Several adjustments were made to better suit the characteristics of the dataset and improve model performance. These include increasing the batch size from 64 to 128, adding L2 kernel regularisation to reduce overfitting, and incorporating early stopping and learning rate reduction callbacks to stabilise training and improve convergence.

The model begins with a 1D convolutional layer that extracts local temporal patterns from the input time series. This layer applies multiple filters, equal to the Transformer embedding dimension, with a specified kernel size and stride, allowing the model to downsample the sequence while learning high-level representations. Batch normalisation is applied afterwards to stabilise and speed up training. Next, positional embeddings are added to the convolutional output to inject information about the position of tokens within the sequence, which is essential for the Transformer to process order-dependent data.

Following the embedding stage, the core of the model consists of a stack of Transformer encoder layers. Each encoder layer incorporates multi-head self-attention to learn dependencies across different time steps in the sequence, followed by a position-wise feed-forward network (FFN). Residual connections and layer normalisation are used throughout to preserve information and ensure stable gradients. Dropout is applied in attention and FFN layers to mitigate overfitting. This structure enables the model to capture both short-term and long-term dependencies, which could be vital in financial time-series forecasting.

The output of the final Transformer layer is globally averaged across the time dimension, compressing the temporal information into a single fixed-size vector. This representation is passed through a dense layer with ReLU activation, followed by dropout for regularisation. Finally, a sigmoid-activated output layer predicts the binary class label—indicating the direction of stock movement. L2 regularisation is applied to the dense layers to further reduce the risk of overfitting.

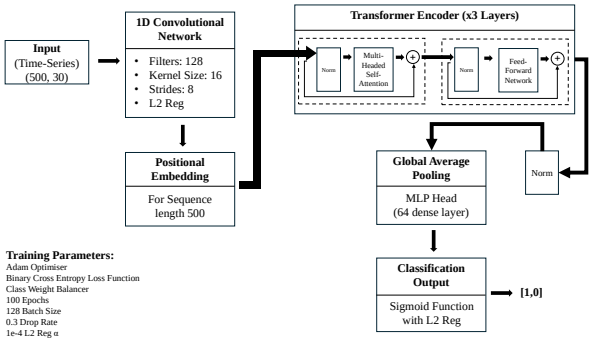

Figure 3. Architecture of the CNN-Transformer model

5.2.3. Training Method

Compared with the original CNN-transformer model, the optimiser was changed from AdamW to the standard Adam optimiser, which provided comparable results with a simpler configuration. To avoid the influence of unbalanced binary classification labels, we account for class imbalance by computing and applying class weights dynamically. During training, the model utilises callbacks like early stopping and learning rate reduction to enhance convergence and generalisation:

We employ an `EarlyStopping` callback. Training stops if the validation loss does not improve for 20 consecutive epochs. Furthermore, the model automatically restores the weights from the epoch with the best validation loss.

We also use a learning rate scheduler based on the `ReduceLROnPlateau` callback in Keras. The learning rate is initially set to 0.001, and will be reduced by a factor of 0.5 if the validation loss does not improve for 5 consecutive epochs (`patience = 5`). The minimum learning rate is set to 1×10^{-6}. This strategy helps the model converge more smoothly when the training plateaus.

Considering the high-dimensionality of the covariates space (i.e. up to 30 features), the model may face the risk of overfitting. Thus, L2 regularisation is also utilised as described in the section **Model Architecture** of CNN-Transformer.

6. Results

The objective of the numerical evaluation was to compare predictive capabilities of the various models employed on binary classification tasks for FX time-series forecasting, particularly on the USD/JPY exchange rate (1-day, 5-day, and 10-day forward binary returns). The models were assessed based on their ability to generalise across unseen data and to distinguish directional market movements. In our study, three key evaluation metrics were utilised:

- **Training and Validation Accuracy**: Measure how well each model fit the training data and how well it generalises to the validation data.
- **Test AUC**: Measure a threshold-independent performance measure, focusing on the model's discriminative power.

The quantitative model results are summarized in Table 1, and the 10-day forward return results are plotted in Figure 4, based on all models generalising better at this horizon. For 10-day forward returns, we see that CNN-Transformer is the top performer with 65.9% validation accuracy and 64.0% test AUC, followed by the Transformer (62.9% validation accuracy, 61.0% test AUC). The GRU model outperformed the LSTM, with a validation accuracy of 53.0% and test AUC of 56.1%, while the LSTM showed the lowest performance (36.8% validation accuracy, 57.2% test AUC). The CNN-Transformer's superior performance demonstrates the benefit of combining CNNs and transformers for time series forecasting.

Notably, for 1-day returns, the GRU showed the strongest validation accuracy (55.7%), while the CNN-Transformer achieved the highest test AUC (52.0%). For 5-day returns, the Transformer maintained 50.0% validation accuracy, with the GRU delivering the best test AUC (56.9%).

These results highlight that while the CNN-Transformer combination provides the most robust performance across time horizons, traditional RNN architectures like the GRU can remain competitive for very short-term forecasting tasks. The LSTM's consistently weaker performance suggests it may be less suitable for this specific forecasting application compared to other architectures tested.

7. Discussion

Across all models, predictive performance on the 10-day forward-return classification task mostly exceeded that on the 1-day and 5-day horizons: validation accuracy and test AUC were uniformly higher for the 10-day forecast.

This can possibly be explained by persistent trends driven by carry trades, momentum effects or evolving interest-rate expectations have time to assert themselves and are less likely to be reversed by short-term oscillations. A 10-day window also aligns more closely with technical indicators like moving-average crossovers or volatility breakouts, which accumulate more reliably over multiple trading days. The feature set, which includes indicators like the 10-day Rate of Change and 20-day Bollinger Bands, may be better suited for the 10-day horizon, as these indicators are designed to capture medium-term trends and momentum. Shorter horizons (1-day and 5-day) might not provide enough data points for these indicators to generate reliable signals, contributing to the observed performance gap.

Furthermore, the 10-day price movements typically exhibit a more balanced distribution of up and down moves and larger absolute swings, making the classification boundary less skewed and easier for sequential models to separate—thus boosting both AUC and accuracy. Additionally, this trend could be that short-term returns (1-day and 5-day) tend

Table 1. The results of the training models. The bold numbers represent the maximum values of the metric for these models.

	Train Acc	Val Acc	Test AUC
Forward 1-Day Returns			
LSTM	50.3%	53.2%	46.6%
GRU	52.0%	**55.7%**	50.0%
Transformer	50.0%	50.0%	50.0%
CNN-Transformer	**53.7%**	51.9%	**52.0%**
Forward 5-Day Returns			
LSTM	50.5%	40.5%	55.6%
GRU	**51.0%**	42.3%	**56.9%**
Transformer	50.0%	**50.0%**	50.0%
CNN-Transformer	50.3%	48.2%	54.0%
Forward 10-Day Returns			
LSTM	52.5%	36.8%	57.2%
GRU	56.4%	53.0%	56.1%
Transformer	65.9%	62.9%	61.0%
CNN-Transformer	**73.4%**	**65.9%**	**64.0%**

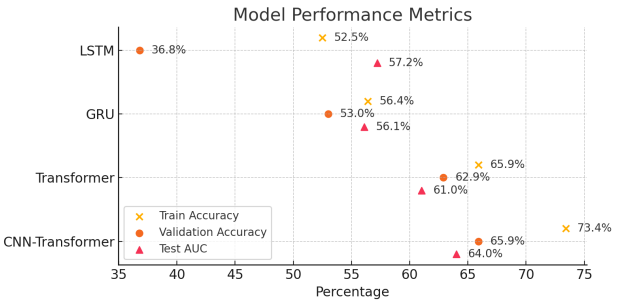

Figure 4. The model performance for forecasting the Forward 10-Day Returns.

to be much noisier, heavily influenced by microstructure effects, random price perturbation, and short-term market sentiments shifting. On this note, it could be extremely difficult for machine learning models to predict reliably. On the other hand, longer-term returns (10-day) may better reflect underlying macroeconomic forces, broader market trends, and technical indicators, which are considerably more stable and thus, learnable over time by sequential models deployed. To add on, longer forecast horizons also smooth out the random walk characteristics of price changes, making directional movements easier to classify.

While binary classification simplifies the forecasting task, it overlooks the magnitude of price movements, which is critical for practical applications like trading. The 10-day horizon's larger absolute swings make the binary labels more meaningful, but for shorter horizons, the distinction between small and large movements may be more pronounced, leading to less informative predictions.

8. Interpretations

The potential success of the CNN-Transformer model can be attributed to its ability to integrate local feature extraction (via convolutional layers) and global temporal dependencies (via multi-head self-attention), thus being able to model both short-term and long-term patterns.

The classic Transformer demonstrated that even with a reduced number of encoder layers, self-attention mechanisms are effective in capturing sequence-wide relationships without relying on complex structures.

The GRU model's modest performance reinforces the notion that simpler recurrent architectures, when modified with careful regularisation techniques, can still be effective. The results reflect its ability to capture some temporal dependencies through its gating mechanisms, but also reveal instability and sensitivity to changing market conditions. While the model performs better in trending environments, it indicates that short-term predictions remain challenging, likely due to the noisy and stochastic nature of FX data.

Lastly, the LSTM model showed a large gap between training and validation accuracy (52.5 % vs 36.8%) at the 10-day horizon, indicating overfitting and an inability to generalize noisy FX data. Despite its theoretical advantages for long-sequence memory retention, can struggle in practice, possibly due to model complexity, sensitivity to noisy financial data difficulty in tuning it such that it reliably filters out high-frequency noise but still retains multi-day or multi-week signals. Figure 5 shows the distinct characteristics of the four models in terms of training stability and predictive performance.

Model	Pros	Cons
CNN-Transformer	• Achieved the highest training and validation accuracy, and the best test AUC • Effectively captured both local patterns (via the CNN layer) and long-range dependence (via self-attention) • Regularization terms like dropout and L2 penalty helped to mitigate overfitting despite its complexity	• There is a noticeable gap between training and validation accuracy, suggesting mild overfitting
Transformer	• Good generalization with close training and validation accuracy values • Demonstrated that a light-weight Transformer (only three encoder blocks) can effectively model sequential relationships found in FX time-series	• Lacked the convolutional feature extraction power as seen in the CNN-Transformer model, slightly limiting performance • Slightly lower AUC compared to the CNN-Transformer, indicating room for improvement
GRU	• Simpler architecture and efficient training, supported by effective regularization strategies (dropout, batch normalization, L2 regularization).	• May underperform in highly volatile or deeply non-linear regimes compared to models with more flexible attention mechanisms.
LSTM	• Theoretically well-suited for long-sequence modeling due to its memory gates and attention integration.	• Poor validation performance and low test AUC, indicating generalization issues • Susceptible to overfitting or getting trapped in local minima when trained on noisy data • More parameter-heavy than GRU, making it less stable • Difficulty in tuning parameters to address underfitting/ overfitting issues

Figure 5. The interpretations of the training models

9. Conclusion

In this paper, we conducted a time series classification forecasting task of USD/JPY exchange rate, performing a comparative analysis of four different deep learning models, including LTSM, GRU, original Transformer and CNN-Transformer. The models were evaluated for their ability to capture both short-term and long-term dependencies in the time series data, with forecasting horizons of 1-day, 5-day, and 10-day. In our experiments, we evaluated each model's ability to predict price movements. The CNN-Transformer performed the best among the four models, with a test AUC of 64.0% and validation accuracy of 65.9% at the 10-day horizon. The results suggest that deep learning models, particularly those that combine convolutional and attention mechanisms like the CNN-Transformer, show strong potential for financial time-series forecasting. Despite that, the challenges remain in modelling extremely short-term returns due to inherent noise, as evident by the poor model accuracy on 1-day and 5-day forward returns. Future work could explore hybrid modelling approaches, data augmentation strategies, and enhanced regularisation to prevent overfitting to improve robustness of model and enhance predictive accuracy. In addition, testing these architectures on other currency pairs or asset classes could broaden their applicability. These results highlight the potential of hybrid deep learning models for financial forecasting and provide a foundation for developing more accurate and economically viable trading strategies.

References

Bao, W., Yue, J., and Rao, Y. A deep learning framework for financial time series using stacked autoencoders and long short-term memory. *Expert Systems with Applications*, 80:76–93, 2017.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Evans, M. D. D. *Exchange-Rate Dynamics*. Princeton University Press, 2011.

Foumani, N. M., Miller, L., Tan, C. W., Webb, G. I., Forestier, G., and Salehi, M. Deep learning for time series classification and extrinsic regression: A current survey. *ACM Computing Surveys*, 56(9):1–45, 2024. doi: 10.1145/3649448. URL https://doi.org/10.1145/3649448.

Hossain, M. A. M., Karim, R., Thulasiram, R., and Bruce, N. D. A hybrid gru-cnn model for financial market prediction. In *Proceedings of the 2018 International Conference on Computational Science*, pp. 123–135, 2018.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. *International Conference on Machine Learning (ICML)*, pp. 1310–1318, 2013.

Rush, A. The annotated transformer. https://nlp.seas.harvard.edu/2018/04/03/attention.html, 2018. Accessed: 2025-04-28.

Sezer, O. B. and Ozbayoglu, A. M. Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. *Applied Soft Computing*, 70:525–538, 2018.

Sezer, O. B., Gudelek, M. U., and Ozbayoglu, A. M. Deep learning for financial time series forecasting: A review. *Neural Computing and Applications*, 32(14):8975–8992, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008. Curran Associates, Inc., 2017.

Wu, H., Xu, Y., Wang, J., Long, G., Jiang, C., Zhang, S., and Yao, L. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/92ccfe0a7a661c8f1196f3462e0b0f0c-Abstract.html.

Yildirim, S., Yildirim, S., Ucuncu, C., Karakose, M., and Akin, E. Forecasting directional movement of forex data using lstm with technical and macroeconomic indicators. *Financial Innovation*, 7(1):1–16, 2021.

Zeng, Z. et al. Financial time series forecasting using cnn and transformer. *arXiv preprint arXiv:2304.04912*, 2023.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of AAAI Conference on Artificial Intelligence*, 35(12):11106–11115, 2021.

Statement about Individual Contributions

Guanhua Chen - Worked on Transformers Model

Shervin Tan - Worked on Transformers Model

Lim En Quan - Worked on LSTM Model

Ryan Lim - Worked on LSTM Model

Kris Cheung - Worked on GRU Model

Team - Collectively worked on report