Trans4Trade: Integrating Structured State Space Models with Transformers for High-Frequency Trading

A Preprint

Wen Gu

November 8, 2024

Abstract

This paper explores the performance of novel neural networks, including transformers, structured state space models (s4), selective state space models (s6), among others, in quantitative finance, especially in mid to high-frequency trading. It compares them to the canonical, highly effective industry-standard Long Short-Term Memory (LSTM) model. A novel hybrid model, Trans4Trade, is introduced for time series forecasting tasks, which incorporates a structured state space model followed by a transformer encoder and a linear decoder, and we assess gated ReLU and novel normalization techniques to boost the model's performance. We then assess their out-of-sample R^2 scores and ultimately find that Trans4Trade has the best ability among all our models to identify profitable trades and achieves higher cumulative Profit and Loss (P&L) than LSTM, Mamba, and other models.

1 Introduction

Quantitative finance has been under the spotlight since James Simons's legendary quantitative trading strategies yielded great results compared to strategies based on fundamental analysis, showing the effectiveness of data-driven strategies in investing and risk management. This has initiated a quantitative revolution in the landscape of finance.

Because of this transition, time series forecasting has become of interest in the financial industry. Forecasting and analyzing Financial Time Series (FTS) data are of special interest to those who wish to make profitable trades in the financial markets. Stochastic methods have been adopted, with classical approaches such as Autoregressive Integrated Moving Average (ARIMA)[1] and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models[2] still used by some in the industry today. However, with the rise of GPUs and machine learning theory, Deep Learning (DL) models[3] have increasingly been recognized and adopted, especially in sectors such as quantitative hedge funds and market making, due to their ability to capture non-linear structures and analyze data in ways that surpass the performance of methods relying on factors and indicators. Methods that include Recurrent Neural Networks (RNNs)[4], especially LSTMs, have been particularly popular for time series prediction due to their supreme effectiveness, which few models can surpass even today. In fact, LSTMs are still the most widely adopted in

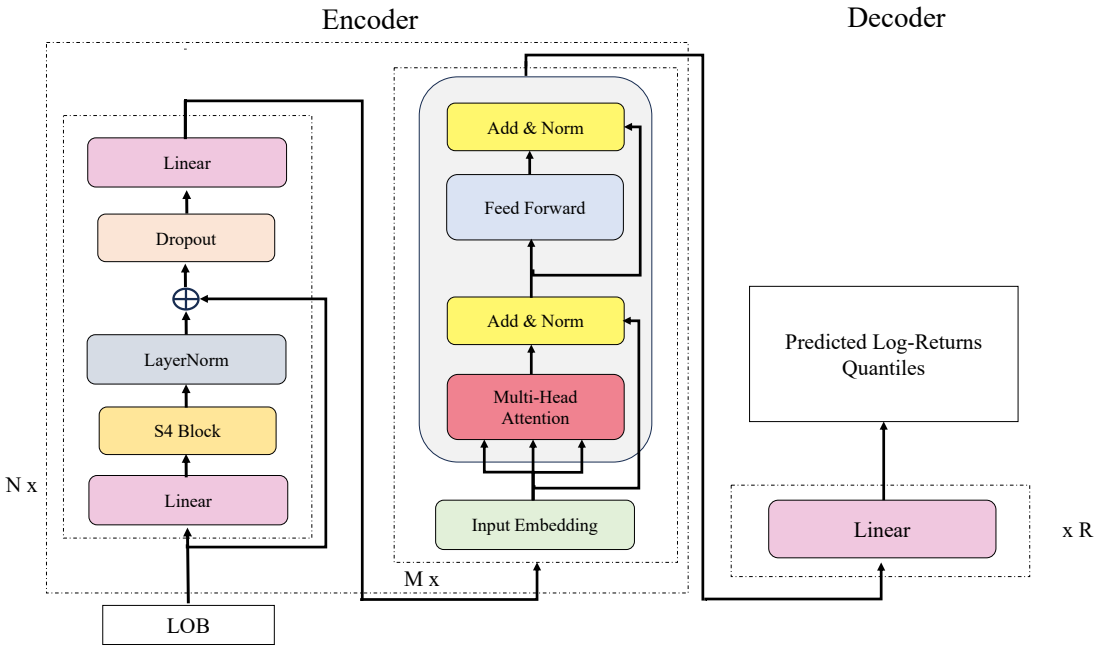

Figure 1: We present Trans4Trade, the first S4-based model for improving long-term memory. Trans4Trade utilizes a stack of S4 blocks and a Transformer Encoder to efficiently capture long-range dependencies between time-series data in a compact latent space. Trans4Trade is a general-purpose framework that is compatible with any parallelizable SSM, including S5 and Mamba.

industries[5]. Recent studies[6] have shown that LSTM[7] remains a surprisingly strong contender in the fields of quantitative finance compared to novel transformer-based models.

Transformers, which incorporate multi-headed attention and have been adapted as replacements for LSTMs in recent years in Natural Language Processing (NLP), have shown great promise[8]. Their adaptation to computer vision tasks and, more recently, to time series forecasting[9] has demonstrated their great versatility. Meanwhile, just as transformers are gaining significant adoption and attention in the Large Language Models (LLMs) industry, an older type of model, state space models, are being refurbished, with noticeable developments in late 2021[10] and 2023[11]. Their characteristics of having continuous states, memory, and convolutional states simultaneously make them stand out. These models have shown comparable performance to transformers in areas where Multilayer Perceptrons (MLP) are used, while attaining a complexity of $O(n)$, compared to $O(n^2)$ for transformers. These two classes of models have better computational parallelizability compared to LSTM, which often has problems with exploding gradients and inefficiency in training[12].

In this paper, we compare the performance of existing models in FTS forecasting. We then introduce a novel hybrid DL model, Trans4Trade. Subsequently, we compare its performance against various high-performance models in the context of FTS forecasting. We focus on collected Limit Order Book (LOB) and aggregated trade (aggtrade) data from cryptocurrency exchanges and process the data before feeding it into the network.

2 Background and Related Work

2.1 FTS

FTS analysis is at the heart of quantitative hedge funds and market makers. Its high noise-to-signal ratio, non-stationarity, trend, drift, auto-correlation, and strong short-term characteristics that decay over time make FTS forecasting a challenging problem[13].

2.2 Traditional Methods for FTS

Stochastic FTS forecasting employs both linear and non-linear methods. Linear models, such as Moving Average (MA), Autoregressive (AR), and ARIMA models, can address both stationary and non-stationary time series. For non-linear models, GARCH and GARCH-based methods are commonly used to forecast stock market volatility, especially in cases of heteroskedasticity[14]. Recently, LSTMs have been successfully applied to high to mid-frequency FTS forecasting, supporting both single-step and multi-step predictions[15]. Hybrid LSTM-CNN architectures[16] are effective for extracting information from LOBs. LSTM-based Additionally, Spiking Neural Networks (SNNs) are employed to predict price spikes, particularly for High-Frequency Trading (HFT) strategies[17].

2.3 Structured State Space Models and Selective State Space Models (Mamba)

Structured state space models and selective state space models (Mamba)[18] have shown comparable performance. The research and use of selective and structured state space models in electronic trading are surprisingly limited, with no available research on topics about structured state space model's performance on financial trading found on Google Scholar. The few studies on selective state space models (Mamba) focus on long-term technical analysis using indicators[19], and no studies have investigated their performance on short-term HFT data.

2.4 Transformer-Based FTS Prediction Models

A significant amount of research on transformers in time series forecasting has been conducted. For FTS prediction and performance evaluation, Transformer-based FTS prediction models are developed and tested by various research entities. Two notable studies by Imperial College London compare the performance of transformer-based models against classic LSTM models as well as novel frameworks like the FEDformer[20]. These studies show that LSTM-based networks remain surprisingly strong, still outperforming transformers and all other architectures by a significant margin in absolute predictive power[21].

3 Data Description

3.1 Orderbook Data

Most of the collected LOB snapshots for BTC-USDT are within a 100 ms interval. We then calculate the weighted midprice, defined as follows:

$$\text{Midprice}_t = \frac{q_a^{1t} \cdot p_a^{1t} + q_b^{1t} \cdot p_b^{1t}}{2} \tag{3.1}$$

where p_a^{1t}, q_a^{1t}, p_b^{1t}, and q_b^{1t} represent the ask price, ask quantity, bid price, and bid quantity at level 1 of the LOB at time t. We process the LOB data by aggregating it by trading volume in an event-based manner, as our analysis focuses on trade time rather than wall clock time. This approach helps to predict future trade prices more accurately and brings the distribution of observed values closer to a normal distribution (see subsection 3.2 for more details). In this paper, "midprice" and "weighted midprice" are used interchangeably.

3.2 Log-Returns and Data Manipulation

FTS data has high variance, so it is essential to make the data stationary.

- **Price Differencing**: $d_{t+1} = p_{t+1} - p_t$
- **Returns**: $r^*_{t+1} = \frac{p_{t+1} - p_t}{p_t}$
- **Log-Returns**: $r_{t+\tau} = \log(p_{t+\tau}) - \log(p_t)$

Each of these transformations helps remove price non-stationarity. However, price differencing retains units and is proportional to the asset's underlying price, which is not ideal for highly volatile assets like Bitcoin. Returns and log-returns are unitless because they represent the price change ratio. Returns indicate asset price changes as ratios or percentages and are widely used in finance.

In this paper, we focus on forecasting log-returns across different time horizons, defined as follows:

$$r_{t+\tau} = \log(p_{t+\tau}) - \log(p_t) = \log\left(\frac{p_{t+\tau}}{p_t}\right) \tag{3.2}$$

When reading the LOB data, we notice that much of the time, graphing the midprice shows that the data is information-sparse and that the midprice is too stationary. To make the data information-dense, one approach is to aggregate the data by dollar volume, grouping the trading data based on the volume traded. Aggregating the data in an event-based manner makes it information-dense, thus more suitable for the model to forecast. Note that the aggregation does not disrupt the bid-ask price levels. Additionally, aggregating by dollar volume helps to understand the true financial impact or liquidity in the market. Higher dollar volumes indicate more capital being moved, which may signal stronger investor interest or conviction compared to periods with lower dollar volume, even if the number of trades is similar. Thus, this data thinning technique is widely used in the industry.

With aggregation, the distribution of returns is closer to a normal distribution.

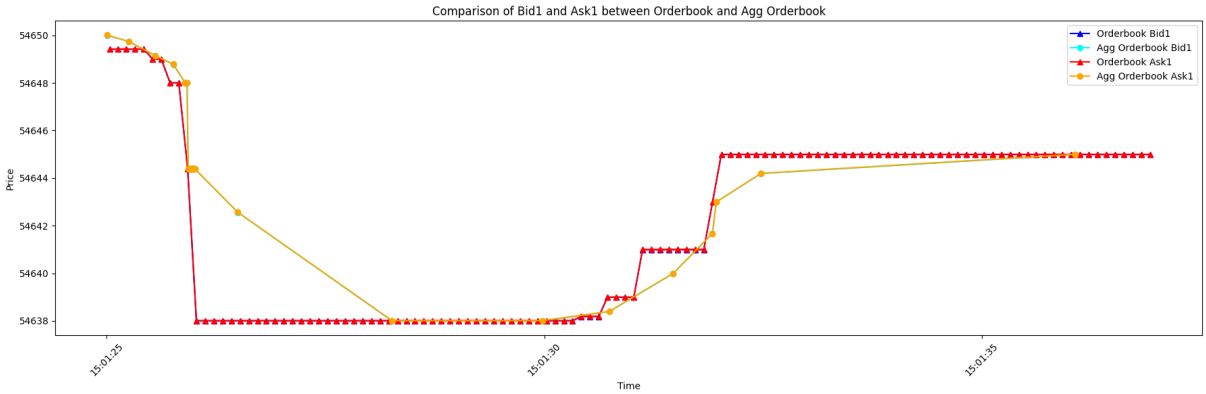

Figure 2: Aggregated LOB vs Original LOB

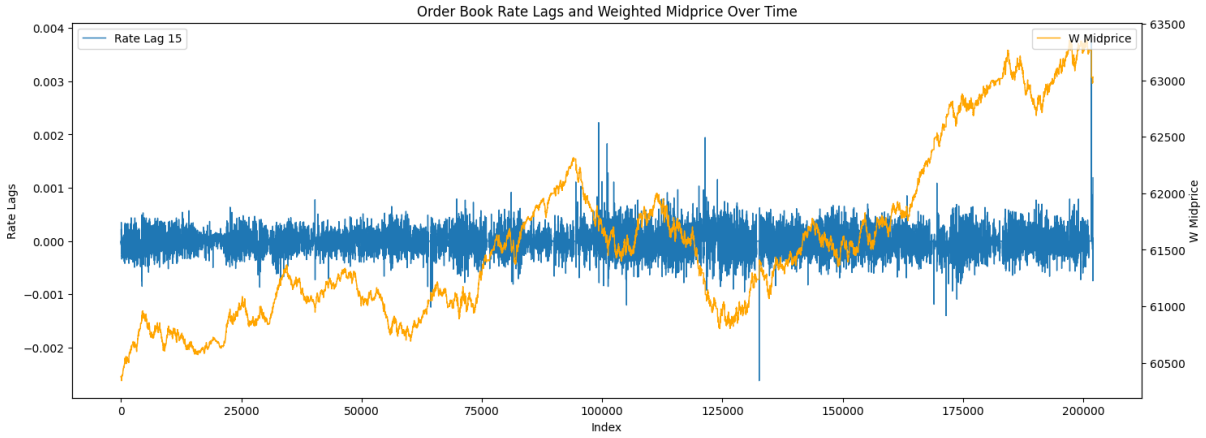

Figure 3: Stationary Mid-price Change VS Non-Stationary Mid-price

4 Model Architecture

4.1 Encoder-Decoder Architecture

The goal is to forecast future values of a target $y_{t+1:t+\tau} = \{y_{t+1}, \ldots, y_{t+\tau}\}$ over the interval from $t+1$ to $t+\tau$, where τ represents the forecast horizon. This prediction is based on past values of the target $y_{t-k:t} = \{y_{t-k}, \ldots, y_t\}$ and other external inputs $x_{t-k:t} = \{x_{t-k}, \ldots, x_t\}$ across a look-back window of length k. This approach, known as sequence-to-sequence (seq2seq), seeks to estimate $\hat{y}_{t+1:t+\tau} = f(y_{t-k:t}, x_{t-k:t})$.

In this encoder-decoder architecture, historical information is encoded into a latent representation Z and subsequently decoded from a subset of its elements to produce the forecast. Specifically, the encoding function $g_{\text{enc}}(y_{t-k:t}, x_{t-k:t})$ maps historical data to the latent space $z_{t-k:t} \subset Z$, while the decoding function $g_{\text{dec}}(z_{t-k:t})$ reconstructs the prediction as $f(y_{t-k:t}, x_{t-k:t}) = g_{\text{dec}}(z_{t-k:t})$, with $g_{\text{enc}}(.)$ and $g_{\text{dec}}(.)$ as the encoder and decoder functions, respectively.

5 Models: Details and Illustrations

In this section, we provide a comprehensive overview of the models evaluated in this study, including their architectural structures, mathematical formulations, and unique

features such as layer normalization. The models discussed are Long Short-Term Memory (LSTM)[5], Transformer[8], DLinear[22], Structured State Space Sequence Model (S4)[10], Structured State Space Sequence Model (S6)[18], and our proposed hybrid model, S4+Transformer (Trans4Trade).

5.1 Long Short-Term Memory (LSTM)

Recurrent Neural Networks (RNNs)[23] are designed to handle sequential data by modeling dependencies of arbitrary length. LSTMs[24], a specialized type of RNN, address the vanishing and exploding gradient problems inherent in traditional RNNs through the use of memory cells and gating mechanisms.

$$h_t = \phi_h \left(W_h h_{t-1} + W_x x_t + b_h \right) \tag{5.1}$$
$$y_t = \phi_y \left(W_y h_t + b_y \right) \tag{5.2}$$

An LSTM unit consists of a cell state and three gates: forget gate, input gate, and output gate. These gates regulate the flow of information, allowing the network to retain or discard information as needed. The mathematical operations within an LSTM unit are defined as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5.3}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{5.4}$$
$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{5.5}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5.6}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{5.7}$$
$$h_t = o_t \odot c_t \tag{5.8}$$

where σ denotes the sigmoid activation function, tanh is the hyperbolic tangent function, and $\odot$ represents element-wise multiplication.

5.2 Transformer

The Transformer architecture[8] has revolutionized various domains, particularly Natural Language Processing (NLP), by leveraging self-attention mechanisms to model dependencies without relying on recurrence.

A standard Transformer model comprises an encoder and a decoder, each consisting of multiple layers. Each layer in the encoder has a multi-head self-attention mechanism followed by a position-wise feedforward network. Layer normalization and residual connections are applied after each sub-layer.

The multi-head self-attention mechanism is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} \right) V \tag{5.9}$$
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) W^O \tag{5.10}$$
$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

The position-wise feedforward network is defined as:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{5.11}$$

Layer normalization is applied to stabilize training:

$$\text{LayerNorm}(x) = \frac{x - \mu}{\sigma}\gamma + \beta \tag{5.12}$$

where μ and σ are the mean and standard deviation of x, and γ and β are learnable parameters.

In this paper, according to previous studies, the full encoder-decoder transformer performs abysmally in this kind of task (negative R^2 scores), but HFformer (a transformer encoder and a linear decoder), easily outperforms transformers, FEDformer, and Autoformer in this kind of task.[cite] So in this paper, we will refer to HFformer as transformer, as it is so close to a transformer.

5.3 DLinear

DLinear[22] is a straightforward yet effective model for time series forecasting that emphasizes linear transformations. It leverages dual linear layers to capture both trend and seasonal components of the data.

The DLinear model consists of two parallel linear layers: one dedicated to modeling the trend component and the other for the seasonal component. The outputs of these layers are then combined to produce the final forecast.

The mathematical operations are defined as:

$$\hat{y}_{\text{trend}} = W_{\text{trend}}x + b_{\text{trend}} \tag{5.13}$$

$$\hat{y}_{\text{seasonal}} = W_{\text{seasonal}}x + b_{\text{seasonal}} \tag{5.14}$$

$$\hat{y} = \hat{y}_{\text{trend}} + \hat{y}_{\text{seasonal}} \tag{5.15}$$

where $W_{\text{trend}}, W_{\text{seasonal}}$ and $b_{\text{trend}}, b_{\text{seasonal}}$ are learnable parameters.

5.4 Structured State Space Sequence Model (S4)

The Structured State Space Sequence Model (S4) is designed to efficiently capture long-range dependencies in sequential data by leveraging state space models (SSMs) with carefully structured state matrices and advanced discretization techniques.

S4 utilizes continuous-time linear state space models with a special parameterization of the state matrix $\mathbf{A}$ using the HiPPO (High-order Polynomial Projection Operator) framework[25]. This allows the model to represent the entire history of the input sequence in a compressed state, enabling efficient modeling of long sequences.

The continuous-time linear time-invariant (LTI) state space model is defined as:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t) \tag{5.16}$$

$$y(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}u(t) \tag{5.17}$$

The HiPPO framework constructs the state matrix $\mathbf{A}$ to project the input function onto a basis of orthogonal polynomials (e.g., Legendre polynomials). The HiPPO-Legendre matrix $\mathbf{A}$ is explicitly defined as:

$$A_{n,k} = \begin{cases} -\sqrt{2n+1} \cdot \sqrt{2k+1}, & \text{if } n > k \\ -(n+1), & \text{if } n = k \\ 0, & \text{if } n < k \end{cases} \tag{5.18}$$

This structure ensures that the state matrix captures the necessary dynamics for projecting the input sequence effectively. However, directly using the HiPPO matrix is computationally intensive. Therefore, S4 parameterizes $\mathbf{A}$ using a diagonal-plus-low-rank (DPLR) structure to enable efficient computation.

To handle discrete-time sequences, the continuous-time SSM is discretized using advanced techniques such as the bilinear (Tustin) transform or rational approximations, resulting in discretized state-space matrices $\bar{\mathbf{A}}, \bar{\mathbf{B}}$.

Layer normalization is applied to stabilize training:

$$\text{LayerNorm}(\mathbf{x}) = \frac{\mathbf{x} - \mu}{\sigma}\gamma + \beta \tag{5.19}$$

It is worth noting that S5, a simplified variant of S4, retains the core benefits while reducing complexity, making it more accessible for practical implementations[**?**].

5.5 Selective State-Space Model (S6)

The **Selective State-Space Sequence Model (S6)**[18] is an advanced adaptation of the Structured State-Space Sequence (S4) model, designed to enhance efficiency in long-sequence modeling through selective mechanisms. Unlike S4, which processes all states uniformly, S6 introduces a selective attention mechanism that dynamically determines which states to update based on their relevance within the sequence.

For an input sequence $x_t \in \mathbb{R}^d$ and hidden state $h_t \in \mathbb{R}^N$, the S4 discrete update rule is modified in S6 to include a selection function $f_{\text{sel}} : \mathbb{R}^N \to \{0,1\}^N$. The update equation becomes:

$$h_{t+1} = f_{\text{sel}}(h_t) \circ (Ah_t + Bx_t) + (1 - f_{\text{sel}}(h_t)) \circ h_t, \tag{5.20}$$

where A and B are state transition matrices, $\circ$ denotes element-wise multiplication, and $f_{\text{sel}}(h_t)$ is a binary mask determining whether each state component is updated (1) or retained from the previous timestep (0).

The output y_t is computed as:

$$y_t = Ch_t, \tag{5.21}$$

where C is the output matrix mapping the selected hidden states to the output space. By selectively updating only relevant components of h_t through f_{sel}, S6 achieves enhanced computational efficiency. This selective updating focuses on salient components, enabling S6 to capture long-range dependencies more effectively than S4 without uniformly processing each state.

5.6 S4+Transformer Hybrid (Trans4Trade)

Trans4Trade is our proposed hybrid model that combines the strengths of the S4 model and Transformer encoders to effectively capture both long-range dependencies and complex temporal patterns in financial time series data.

The Trans4Trade model integrates a stack of S4 layers with a Transformer encoder. The S4 layers efficiently capture long-term dependencies, while the Transformer encoder models intricate temporal relationships through self-attention mechanisms. A linear decoder then produces the final forecasted values.

Input sequences are first processed through S4 layers to capture long-term dependencies, and a Gated ReLU activation function is applied. The resulting representations are then fed into the Transformer encoder, which applies multi-head self-attention to model complex temporal interactions.

$$\text{GatedReLU}(x) = \text{ReLU}(x) \odot \sigma(W_g x + b_g) \tag{5.22}$$

Layer normalization is applied after each sub-layer to stabilize the training process:

$$\mathbf{H} = \text{LayerNorm}(\mathbf{H}) \tag{5.23}$$

Finally, the output from the Transformer encoder is passed through a linear layer to generate the forecast:

$$\hat{y} = W_{\text{dec}}\mathbf{H} + b_{\text{dec}} \tag{5.24}$$

Advantages:

- Efficiently captures both long-term and complex temporal dependencies.
- Combines S4's efficiency with Transformer's expressiveness.
- Enhanced performance through gated activations and normalization.

5.7 Summary of Models

The models evaluated represent a spectrum of approaches to time series forecasting:

- **LSTM**: Captures sequential dependencies with gating mechanisms, currently the canonical standard model used in industry.
- **Transformer**: Uses self-attention to model complex relationships without recurrence.
- **DLinear**: Emphasizes linear transformations for trend and seasonality.
- **S4**: Efficiently models long-range dependencies using state space models and HiPPO matrices.
- **S6**: Further optimizes state space models for enhanced computational efficiency and scalability.

- **Trans4Trade (S4+Transformer)**: Hybrid model combining S4's efficiency and Transformer's expressiveness for superior forecasting.

Each model's unique architecture and mathematical foundations contribute to their strengths and limitations in handling high-frequency trading data and financial time series forecasting tasks.

6 Experiment Setup

6.1 Setup

Our model is based on a combination of Structured State Spaces for Sequence Model (S4) and Transformer encoder. We will use these two parts to form our encoder. The S4 model is responsible for capturing local features in the data latent space, and the Transformer is used to capture long-term dependencies between global data. Finally, a linear layer is used as our decoder to suit the needs of our task. The hyperparameters and model parameters used in training are shown in the Table 1.

Model	Layers	Activation Function	Optimizer	Loss Function	Batch Size
LSTM	2 Layers size 64	PReLU	AdamW with 0.001 learning rate	MSE	1024
Dlinear	1 Series Decomp 1 Moving Average 1 AvgPool1d 3 layers size 100	None	AdamW with 0.1 learning rate	MSE	256
S4	2 S4Block and 2 LayerNorm 2 Dropout1d 1 Layers size 36	GELU	AdamW with 0.1 learning rate	MSE	256
Mamba	2 S4Block and 2 LayerNorm 2 Dropout1d 1 Layers size 36	GELU	AdamW with 0.1 learning rate	MSE	256
Transformer	2 encoders and 2 decoders both of size 64 with 6 Attention heads label length 36	PReLU	AdamW with 0.1 learning rate	MSE	256
Trans4Trade	3 S4 blocks and 3 Transformerencoders as encoders 1 Layers size 64 as decoders label length 36	Spiking ReLU, ReLU or Gated ReLU	AdamW with 0.1 learning rate	MSE	256

Table 1: Model Configurations and Hyperparameters

To compare the performance of different architectures, we use a subset of 350000 LOB snapshots shown in Fig 4, from step 2024-08-27 14:49:00.473 to 2024-08-29 16:28:02.758, with an 80:20 split of training and validation data. The back testing uses a subset of 200k LOB snapshots.

6.2 Data

The input data fed into the models consists of 38 features:

- 9 bid prices and their associated quantities
- 9 ask prices and their associated quantities
- Historical lagged log-return variable $\log\left(\frac{\text{Midprice}_t}{\text{Midprice}_{t-\tau}}\right)$, where τ is the forecast horizon
- The weighted midprice

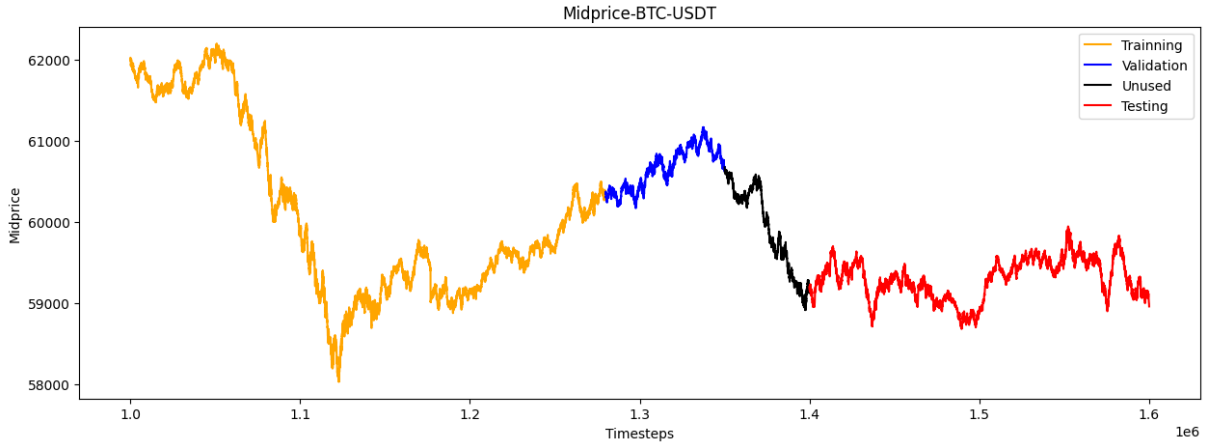

Figure 4: Subset of LOB snapshots used to assess model performance - Midprice.

The model's output is the log-return for a given forecast horizon $\tau \in \{1, \ldots, 30\}$. Here we will focus on horizons of 5 to 15.

The training process utilizes the AdamW optimizer, an adaptive moment estimation method with weight decay. AdamW estimates the first and second moments of the gradient and adjusts the learning rate for each feature, while its decoupled weight decay aids in achieving optimal convergence[26].

Both Mean Squared Error (MSE) and Mean Absolute Error (MAE) loss functions were evaluated across the LSTM[24], Transformer(HFformer)[8], Dlinear[22], Trans4trade, Mamba and S4[10] models. Although MSE is more sensitive to outliers than MAE, models trained with MSE loss achieved higher out-of-sample R^2 scores.

6.3 Evaluation Metric

Out-of-sample R^2 explains the proportion of variance that is explained by the model and is widely used in financial modeling.

Mathematically,

$$R^2_{\text{out-of-sample}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{6.1}$$

where y_i are the true values, $\hat{y}_i$ are the model's predictions, and $\bar{y}$ is the mean of the observed values in the test set. R^2 has the advantage of being interpretable relative to a baseline score. However, R^2 is sensitive to outliers. Finally, after each training epoch, the model's performance is evaluated using training and validation sets to avoid overfitting.

7 Results

7.1 Out-of-Sample R^2 Score Results

We see that the Trans4Trade series achieve first-class out-of-sample R^2 scores. Canonical LSTM remains unsurprisingly strong. Surprisingly, pure state space models don't work well. Transformer (HFformer) is outperformed by LSTM on all horizons, though it is relatively close. The Trans4Trade model has a diverse R^2 score range, indicating potential

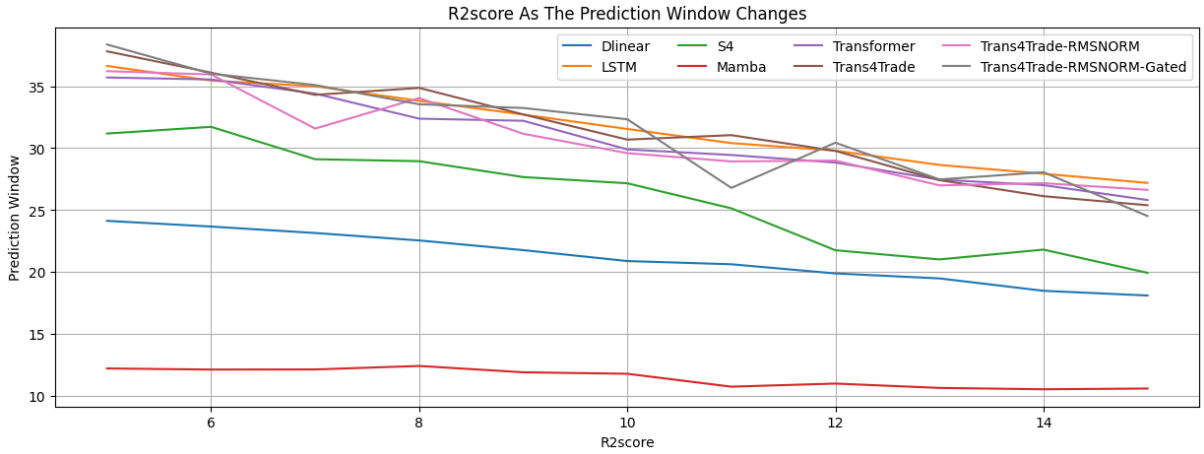

Figure 5: Percentage of out-of-sample R^2 scores

for more diverse trading signals. Later we will show that Trans4Trade has a better P&L score than LSTM, indicating a more balanced trading signal generation.

7.2 Ablation Study accessing R^2 Metric

We see that both the transformer and S4 hold a pivotal role in boosting the model's performance. Adding other components makes the R^2 scores more diverse. A reasonable trading strategy would be to take advantage of this diversity in signal generation and to use diverse models to vote on trades.

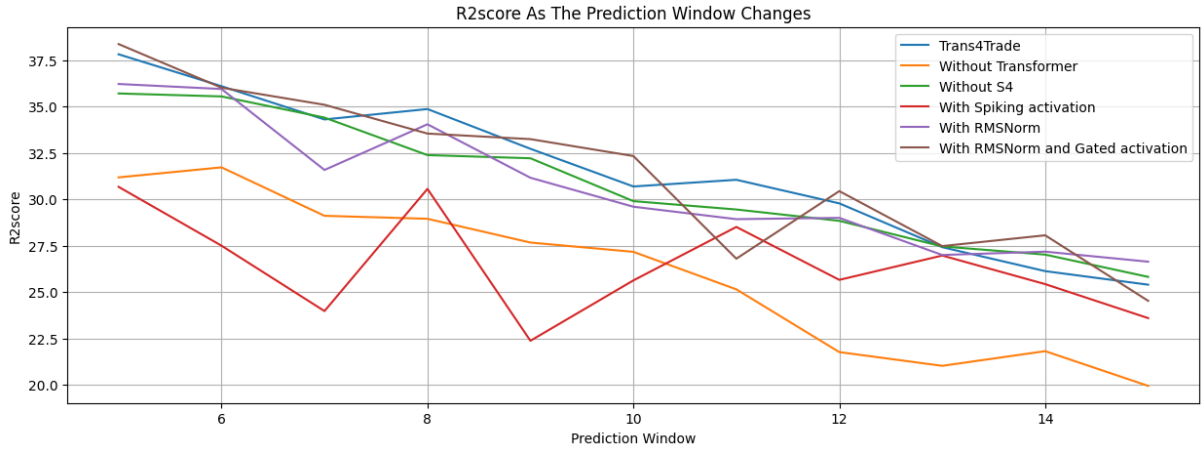

Figure 6: Ablation Study

The out-of-sampleR^2 score of the original Trans4Trade model seems to be most even across all horizons, while the addition of different components will strengthen or weaken some of the out-of-sampleR^2 scores, though it is important to acknowledge the fact that out-of-sample R^2

7.3 Backtesting Results

Backtesting is conducted over 200k LOB snapshots, from 2024-08-29 18:58:41.607 to 2024-08-30 14:44:04.312. If model outputs positive signal over the threshold, open a long

position. If outputs negative signal over the singal threshold, we trigger a short position. A fixed quantity of 0.1 BTC is traded.

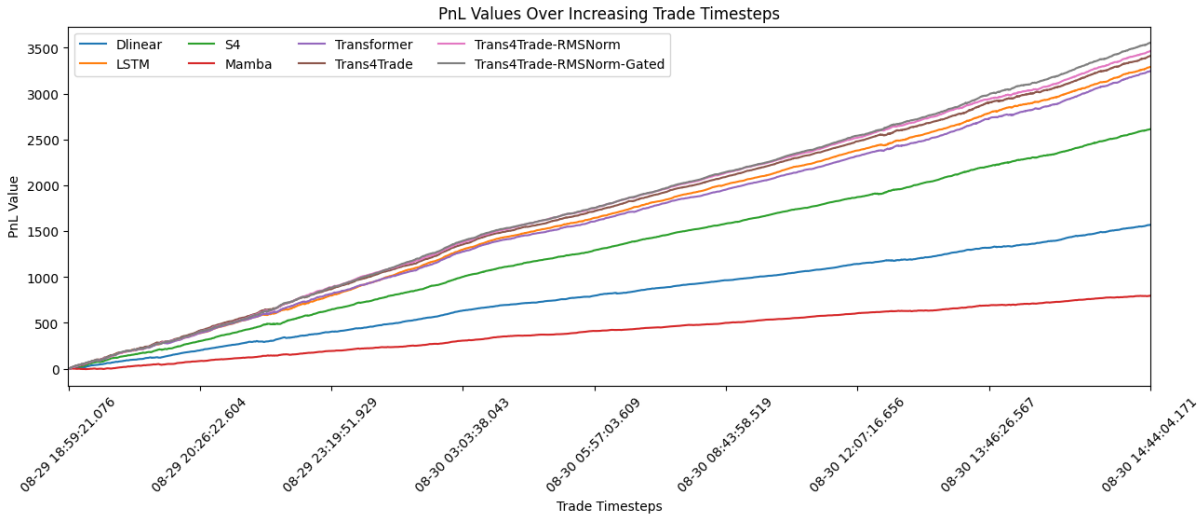

Figure 7: Cumulative P&L

We observe that the Trans4Trade series gains a considerable margin over other models. All variations of Trans4Trade outperformed LSTM and other models. Trans4Trade tends to output a more balanced signal, compared to that of LSTM, and this could account for its advantage in P&L. For a more detailed P&L accumuation in a certain window, refer to appendixA.

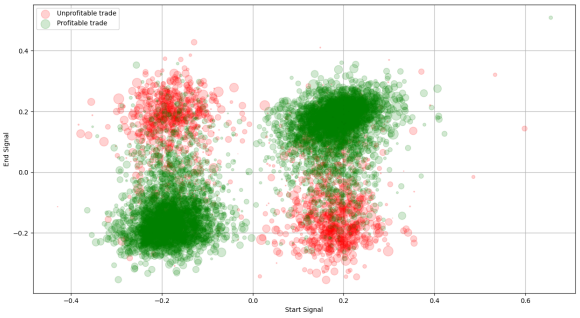

Figure 8: LSTM Signal Output Distribution

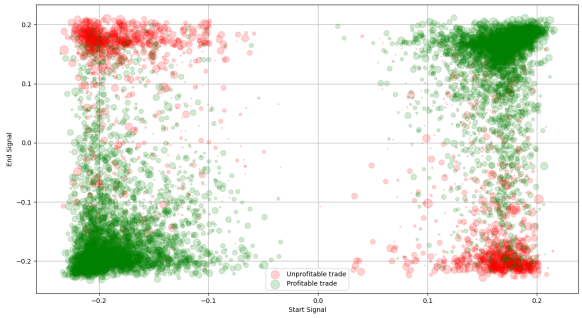

Figure 9: Trans4Trade Signal Output Distribution

LSTM's signal generation is side-heavy, ranging from -0.34 to 0.43 in the end signal ,as shown in Figure 8, while Trans4Trade generates signals in a more balanced and controlled manner, with signal strength ranging from -0.2 to 0.2. This contributes to their overall difference in P&L.

7.4 Summary of the Results

Trans4Trade has demonstrated significant potential in comparison to canonical Long Short-Term Memory (LSTM) models. While LSTM-based models continue to achieve impressive out-of-sample R^2 scores, Trans4Trade, leveraging non-recurrent architectures, not only matches but also surpasses LSTM in certain aspects. This achievement marks

the first instance in which a non-recurrent network has attained performance on par with traditional LSTM standards in financial time series forecasting.

Our experiments reveal that Trans4Trade consistently outperforms LSTM and other baseline models across various forecast horizons, as evidenced by higher cumulative Profit and Loss (P&L) scores and more balanced trading signal distributions. Additionally, the ablation studies indicate that both the Transformer encoder and the Structured State Space (S4) components play pivotal roles in enhancing the model's predictive capabilities. The hybrid nature of Trans4Trade allows it to efficiently capture long-term dependencies and complex temporal patterns, which are critical for high-frequency trading strategies.

In conclusion, Trans4Trade, as the first structured state-space enhanced Financial Time Series (FTS) model, holds considerable promise in replacing canonical LSTMs. Its superior performance and ability to generate more balanced trading signals position it as a strong candidate to become the new industry standard in quantitative finance and high-frequency trading applications.

8 Market Microstructure Analysis

During the training period, we observe that the training MSE decreases rapidly, but the validation MSE increases relatively quickly and the R^2 score diminishes after only a few epochs. According to a study conducted by Imperial College London, the predictive power of these models decreases as the training set increases. This can be explained by the fact that the models tend to learn the short-term noise of the market, and that markets exhibit strong seasonality, even over short periods of time. Human-engineered features could be introduced to address this, as they provide the model with a good guideline to extract patterns rather than being distracted by noise.

9 Limitations of This Study

Binance has recently incorporated a 0.1% fee in its BTC-USDT trading pair, compared to its zero-fee structure in 2022 and 2023. This means that the market conditions are quite different from the 0% fee structure we anticipated during data collection. Nevertheless, it is still sufficient to assess the model's performance on FTS data and the models' ability to identify profitable trades. However, it implies that our predictions of the BTC market are effective up to tens of seconds, demonstrating considerable potential for BTC Exchange-Traded Funds (ETFs) and market making on other BTC trading pairs.

We have made the following assumptions:

1. The impact of the executed order on the market is negligible.
2. The actual execution price is set to the weighted midprice in our model.
3. Trading fees are assumed to be zero.
4. 0.0002% of trading volume is subtracted from P&L to simualte price slippage.
5. The backtesting period is sufficiently long to evaluate model performance.

10 The Following Lines of Research Could Be Conducted in Future

- Evaluating the model's performance on the US stock market, which has a different landscape where markets are more efficient, as fees can be easily reduced for larger trading sizes on Interactive Brokers, as well as on BTC ETFs that have effective fee-controlling mechanisms (also available on Interactive Brokers).
- Assessing its performance on longer (minutes, hours, and days) and shorter (milliseconds) trading tasks.
- Testing its performance in foreign exchange (forex) markets.
- Investigating the impact on performance when time encodings and other human engineered features are added.

11 Conclusion

This study compares the performance of various models in the context of High-Frequency Trading (HFT) and Financial Time Series (FTS) analysis. Through the experiment, we combined the encoder of structured state space models with the encoder of transformers to form Trans4Trade, a hybrid model adapted for HFT. While testing the out-of-sample R^2 scores of various models, we find that Trans4Trade variants and LSTM outperform other models, and that Trans4Trade and its variants gain an overall advantage over LSTM in terms of R^2 scores and predictive accuracy. Finally, Trans4Trade and LSTM are backtested on different strategies, and we show that Trans4Trade generates more balanced and less correlated long and short trading signals than LSTM, resulting in better cumulative P&L. Although these improvements and strategies are trained and backtested on a large set of LOB data collected over two weeks, these methods may yield different results when applied to other pairs, different periods of times, or different financial assets.

References

[1] George EP Box and David A Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association*, 65(332):1509–1526, 1970.

[2] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327, 1986.

[3] Yujie Fang, Juan Chen, and Zhengxuan Xue. Research on quantitative investment strategies based on deep learning. *Algorithms*, 12(2):35, 2019.

[4] Larry R Medsker, Lakhmi Jain, et al. Recurrent neural networks. *Design and Applications*, 5(64-67):2, 2001.

[5] S Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.

[6] Albert Zeyer, Parnia Bahar, Kazuki Irie, Ralf Schlüter, and Hermann Ney. A comparison of transformer and lstm encoder decoder models for asr. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 8–15. IEEE, 2019.

[7] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[8] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[9] Xiyuan Zhang, Ranak Roy Chowdhury, Rajesh K Gupta, and Jingbo Shang. Large language models for time series: A survey. *arXiv preprint arXiv:2402.01801*, 2024.

[10] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[11] Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[13] Patrıcia O Lucas, Omid Orang, Petrônio CL Silva, EMAM Mendes, and Frederico G Guimaraes. A tutorial on fuzzy time series forecasting models: Recent advances and challenges. *Learning and Nonlinear Models*, 19(2):29–50, 2022.

[14] Jorge Caiado and Nuno Crato. A garch-based method for clustering of financial time series: International stock markets evidence. In *Recent advances in stochastic modeling and data analysis*, pages 542–551. World Scientific, 2007.

[15] Mahinda Mailagaha Kumbure, Christoph Lohrmann, Pasi Luukka, and Jari Porras. Machine learning techniques and data for stock market forecasting: A literature review. *Expert Systems with Applications*, 197:116659, 2022.

[16] Wenjie Lu, Jiazheng Li, Yifan Li, Aijun Sun, and Jingyang Wang. A cnn-lstm-based model to forecast stock prices. *Complexity*, 2020(1):6622927, 2020.

[17] Kang Gao, Wayne Luk, and Stephen Weston. High-frequency trading and financial time-series prediction with spiking neural networks. *Wilmott*, 2021(113):18–33, 2021.

[18] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[19] Haohao Qu, Liangbo Ning, Rui An, Wenqi Fan, Tyler Derr, Xin Xu, and Qing Li. A survey of mamba. *arXiv preprint arXiv:2408.01129*, 2024.

[20] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pages 27268–27286. PMLR, 2022.

[21] Putri Utami Rukmana, Hanif Fakhrurroja, et al. Application of lstm, rnn, and transformer in stock price prediction of information technology companies: A comparative analysis. In *2024 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, pages 249–254. IEEE, 2024.

[22] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

[23] Stephen Grossberg. Recurrent neural networks. *Scholarpedia*, 8(2):1888, 2013.

[24] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Interspeech*, volume 2012, pages 194–197, 2012.

[25] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33:1474–1487, 2020.

[26] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

A Appendix

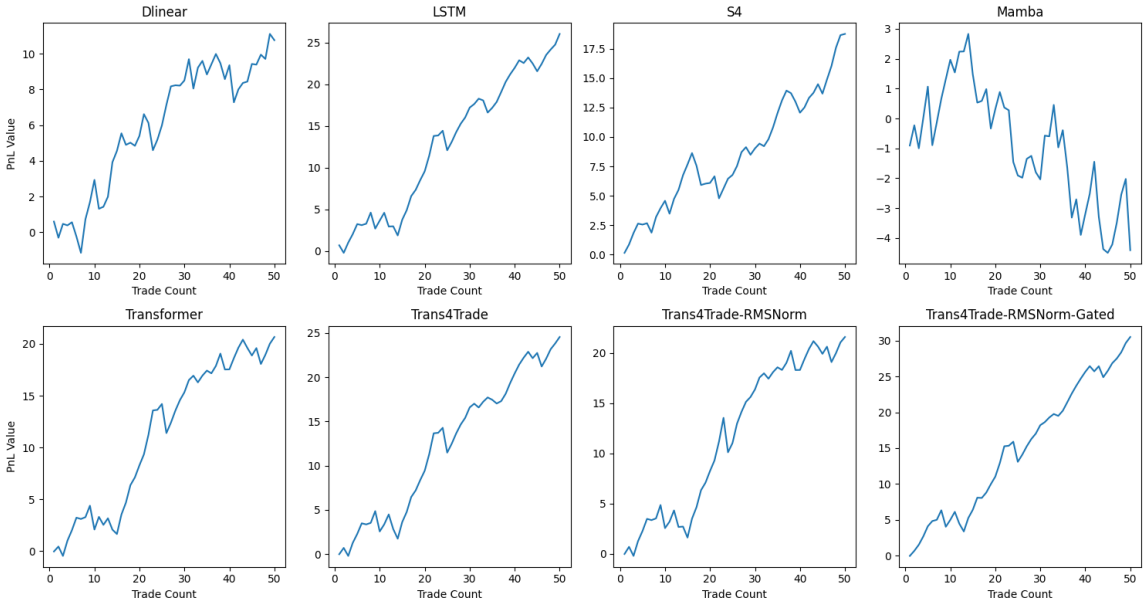

Figure 10: Cumulative of the first 50 transactions P&L

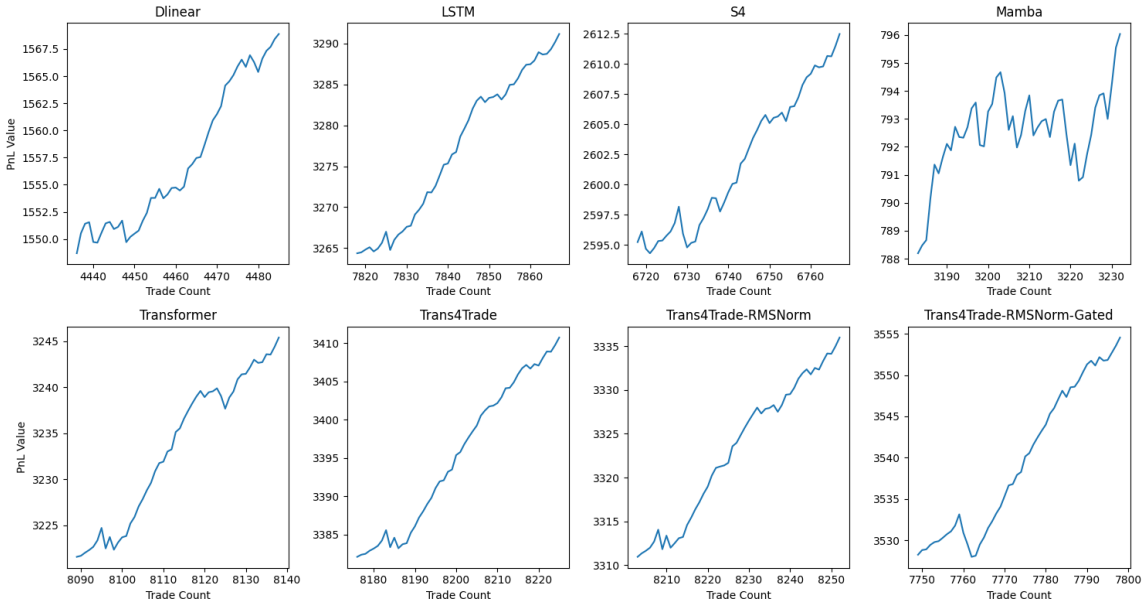

Figure 11: Cumulative of the last 50 transactions P&L