FinTech

Article

How Effective Is Mamba-Augmented Transformer for Stock Market Price Forecasting?

Md. Shahria Sarker Shuvo, Awsaf Tausif Adib, Md. Estehaar Ahmed Emon, Ahasanur Rafi and Rashedur M. Rahman *

Department of Electrical and Computer Engineering, North South University, Dhaka 1229, Bangladesh; shahria.shuvo@northsouth.edu (M.S.S.S.); awsaf.tasusif@northsouth.edu (A.T.A.); estehaar.emon@northsouth.edu (M.E.A.E.); ahasanur.rafi@northsouth.edu (A.R.)
* Correspondence: rashedur.rahman@northsouth.edu

Abstract

Stock price forecasting remains challenging due to the non-linear, noisy, and non-stationary nature of financial time series. Although LSTMs and Transformer-based models have improved sequential modeling, their ability to scale efficiently to long financial sequences remains limited. Recently, selective state space models such as Mamba have emerged as efficient alternatives to self-attention, offering attention-like performance with linear computational complexity. In this study, we systematically evaluate multiple Mamba-augmented Transformer architectures for stock market price forecasting. We further propose *CrossMamba*, a novel architecture that models cross-sequence interactions between encoder and decoder representations using a causal Mamba block. Experiments on multiple S&P 500 and Yahoo Finance stocks show that CrossMamba achieves superior short-horizon performance with 5-day input windows (R^2 up to 0.963), while Hybrid Bi-Mamba performs best for longer horizons, achieving the lowest MAE of 0.67 for 10-day forecasts. Compared with advanced Mamba-based and Transformer baselines, the proposed models achieve competitive accuracy while maintaining substantially improved computational efficiency. These results highlight the effectiveness of Mamba-augmented Transformers as scalable architectures for financial time series forecasting.

Keywords: financial time series; stock price forecasting; Mamba S6; cross-sequence modeling; deep learning

JEL Classification: C22; C53; C45

Check for updates

Academic Editors: Mitra Arami and Patricia Baudier

Received: 22 November 2025
Revised: 27 December 2025
Accepted: 29 January 2026
Published: 9 February 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Accurate stock price forecasting remains a crucial yet highly challenging task in financial market analysis. With the rapid expansion of global financial markets, the ability to anticipate stock price movements directly influences investment strategies, trading decisions, and risk management practices.

However, stock prices are driven by a complex interplay of factors such as geopolitical events, market sentiment, macroeconomic conditions, and firm-level fundamentals. These characteristics produce financial time series that are noisy, non-linear, and non-stationary, making reliable forecasting an enduring research problem. Traditional statistical and econometric models, including Autoregressive Integrated Moving Average (ARIMA) [1] and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) [2], have been

widely employed to model linear dependencies and volatility patterns in stock price data [3]. These models are inherently limited in capturing non-linear dynamics and long-range temporal dependencies [4], which are critical characteristics of financial markets.

These limitations have motivated a shift toward deep learning approaches, particularly recurrent neural networks such as Long Short-Term Memory (LSTM) models, which are better suited for learning complex sequential patterns from historical price data. Long Short-Term Memory (LSTM) networks [5] have shown improved performance relative to several statistical baselines in stock price forecasting tasks [6].

These models are effective at capturing short-term dependencies in sequential data. However, when modeling long-range dependencies, LSTMs face challenges such as vanishing gradients and increasing computational cost as sequence length grows [7,8]. Moreover, their sequential computation can limit scalability for long-horizon forecasting in financial time series [9].

To address these limitations, researchers increasingly adopted Transformer-based architectures, which capture long-range dependencies through self-attention and have advanced time series forecasting, including stock prediction [10]. Transformers can model relationships across distant time steps, which is beneficial for financial forecasting where longer-term temporal structure may be informative. Nevertheless, self-attention introduces quadratic computational and memory complexity, leading to high resource requirements for long sequences [11]. Several Transformer variants aim to mitigate this limitation, including sparse and probabilistic sparse attention mechanisms in BigBird and Informer [12,13]. Autoformer incorporates series decomposition and an autocorrelation mechanism based on FFTs for time series forecasting [14]. iTransformer improves efficiency through instance-based token selection to reduce redundancy during training and inference [15]. Fedformer (Frequency Enhanced Decomposed Transformer) models temporal structure in the frequency domain to capture global and local patterns with reduced computational overhead [16].

These limitations have motivated exploration of alternative sequence modeling architectures beyond self-attention, particularly state space models. Among them, the selective state space model Mamba [17] has emerged as an efficient alternative to self-attention, offering attention-like sequence modeling capacity with linear computational complexity. Mamba-based models have been explored in time-series forecasting [18] and vision tasks [19]. In vision, variants such as Vision Mamba [19] and Vision-Mamba-UNet [20] illustrate the applicability of Mamba-style blocks beyond language modeling. For time-series forecasting, TsMamba [21], Bi-Mamba+ [22], and CMMamba [23] leverage bidirectional state space blocks and channel interaction strategies to capture both short-range and long-range dependencies, often with reduced memory requirements. These results indicate the potential of Mamba-style models for efficient modeling of complex sequential data, motivating further study in financial forecasting. In the financial domain, MambaStock [24] applies Mamba-based modeling to stock price prediction using historical price data and reports competitive performance relative to deep learning baselines. Variants such as SAMBA [25] integrate bidirectional Mamba blocks with adaptive graph convolution to capture both temporal dynamics and market co-movement.

Hierarchical Information-Guided Spatio-Temporal Mamba (HIGSTM) [26] further extends the framework by incorporating macroeconomic signals and index-guided filtering to model both asset-level dynamics and inter-asset relationships. Prior studies also report that substituting the encoder's self-attention layer with a Mamba block can reduce memory and runtime overhead while maintaining forecasting performance on multivariate time series benchmarks such as Electricity and Traffic datasets [27]. Moreover, inserting Mamba blocks between encoder and decoder components has been explored as a way to strengthen temporal dependency modeling through internal recurrence prior to decoding [24].

Despite this progress, prior studies have largely focused on individual architectures or domain-specific adaptations, leaving a gap in systematically benchmarking Mamba-augmented Transformer variants for stock price forecasting across multiple horizons and assets. To address this gap, we introduce and rigorously evaluate Mamba-augmented Transformer architectures for stock market forecasting under consistent experimental settings.

Our key contributions are:

1. Architectural Innovation: We propose CrossMamba, a Mamba-augmented variant that uses concatenation-based cross-sequence modeling for dynamic sequence handling, and extend it to CrossMamba Decomp by integrating series decomposition. We benchmark these against Hybrid Mamba, Encoder-only Mamba, and decomposition-integrated variants.

2. Comprehensive Performance Analysis: Through experiments on three diverse S&P 500 stocks (AAPL, AAL, and ABC) across 5-day and 10-day input windows, we report results using both normalized and denormalized metrics, including performance up to R^2 of 0.963 and MAPE as low as 1.09%.

3. Practical Insights: We analyze the effects of forecasting horizon and stock volatility, and examine when decomposition improves results for more stable stocks versus when it degrades performance for highly volatile stocks.

4. Ablation Studies: We quantify the contribution of key architectural components and input features, including the role of Mamba blocks in temporal dependency modeling.

Specifically, we aim to answer the following research questions:

1. Which Mamba-augmented Transformer variant is most effective for stock price forecasting across different time windows and stocks?

2. Does incorporating series decomposition improve the predictive performance of these models for financial time series?

The remainder of this paper is organized as follows. Section 2 reviews related work on deep learning and state space models for financial time series forecasting. Section 3 describes the proposed architectures, datasets, and training procedures. Section 4 presents experimental results and comparative analyses. Section 5 discusses the findings and their practical implications, and Section 6 concludes the paper with limitations and directions for future research.

2. Related Work

Recent advances in sequence modeling have significantly influenced stock market forecasting, where models must handle noisy, non-linear, and highly non-stationary temporal dynamics. Prior research spans recurrent neural networks, attention-based Transformers, and more recent state space models (SSMs), each offering different trade-offs in expressiveness, computational efficiency, and scalability. Building on these developments, selective state space models such as Mamba aim to provide attention-like sequence modeling capacity with linear complexity, while hybrid architectures attempt to combine complementary strengths of Transformers and SSMs. In this section, we review these approaches and highlight the remaining gaps in financial time series forecasting.

2.1. Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks were introduced to address the vanishing gradient problem in classical recurrent neural networks and remain a widely used baseline for time series forecasting [5]. By incorporating gating mechanisms, LSTMs can model non-linear temporal dynamics and capture short-term dependencies effectively. However, their sequential computation limits parallelism, and performance can degrade for very long horizons due to optimization challenges and increasing computational cost. These

limitations motivate architectures that scale more efficiently to long sequences, particularly in financial forecasting settings where longer context windows may be beneficial.

2.2. Transformer

Transformers introduced self-attention as an alternative to recurrence, enabling direct modeling of relationships across distant time steps and efficient parallel training [11]. This capability has led to strong performance across many sequence modeling tasks, including time series forecasting. However, self-attention incurs quadratic time and memory complexity with respect to sequence length, which can become prohibitive for long-context financial sequences [28]. To reduce this cost, efficient Transformer variants such as Longformer, Reformer, Linformer, and Performer approximate or sparsify attention to improve scalability [29]. Despite these developments, Transformer-based approaches can remain computationally expensive for large-scale or long-horizon stock forecasting, motivating alternatives with linear-time complexity.

2.3. State Space Models (SSMs)

Structured State Space (S4-style) models extend classical state space formulations to learn long-range dependencies efficiently, leveraging structured parameterizations for scalable sequence modeling [30]. A key advantage of SSM-based approaches is that they can achieve linear-time sequence processing and can be implemented in forms that support parallel computation, making them attractive alternatives to quadratic-complexity attention for long sequences. However, earlier SSM variants have been reported to underperform attention-based Transformers on some benchmarks, motivating subsequent work that improves selectivity and expressiveness [17]. This motivates selective SSM designs that better capture complex dependencies while preserving efficiency.

2.4. Mamba: Selective State-Space Modeling

Mamba introduces a selective mechanism into structured state space modeling, making the state update input-dependent and allowing the model to prioritize relevant portions of long sequences [17]. In contrast to quadratic-complexity self-attention, Mamba maintains linear-time and linear-memory sequence processing, which is attractive for long-context forecasting tasks. These properties have motivated recent applications of Mamba-style blocks to time series and financial forecasting, where efficiency is often a practical constraint. However, reported performance can vary across datasets and horizons, and the most effective way to integrate Mamba with Transformer-style components for stock forecasting is still under active investigation.

2.5. Time Series Decomposition

Time series decomposition is commonly used to separate a sequence into components such as trend, seasonality, and residual fluctuations. In forecasting, decomposition can act as an inductive bias by isolating smoother components (e.g., trend) from higher-frequency residuals, which some models can learn more effectively. In this work, decomposition is treated as a modeling choice whose impact is evaluated empirically across different stocks and forecasting horizons, rather than assumed to be universally beneficial.

2.6. Hybrid Architectures: Transformer + Mamba

Hybrid architectures aim to combine the complementary strengths of state space models and attention mechanisms. For example, SST (State Space Transformer) and Mambaformer integrate Mamba-style blocks for efficient long-range modeling while retaining attention components for local interactions, reporting strong results with improved scaling on forecasting benchmarks [31]. KARMA incorporates multilevel decomposition

and processes sequences through Mamba blocks to improve interpretability and forecasting performance in certain settings [32]. More recent encoder-only hybrids such as pdMamba and T-Mamba combine Mamba blocks with Transformer-style mechanisms to capture both global and local temporal patterns [33,34]. Despite these advances, systematic benchmarking of Mamba-augmented Transformer architectures for stock forecasting across multiple horizons and asset types remains limited. This gap motivates the present study, which evaluates multiple Mamba-augmented Transformer variants under consistent experimental settings.

In this study, we aim to forecast stock closing prices using Mamba-augmented Transformer architectures. The methodology involves several key steps. First, we formalize the prediction task as a supervised time series regression problem. Next, we describe the datasets, including cleaning, feature selection, normalization, sequence generation, and train-test splits. We then introduce the evaluation metrics used to measure model performance. Finally, we present the model architectures, including Mamba Encoder-only, Hybrid Mamba, Cross Mamba, and CrossMamba with series decomposition, along with the training procedures and configurations. This structured approach ensures a rigorous and reproducible comparison of the proposed models for financial time series forecasting.

2.7. Finance Implementations of Machine Learning: Asset Pricing and Market Studies

Beyond forecasting accuracy, a large body of finance research evaluates machine learning models through the lens of *economic* objectives such as return predictability, risk premium estimation, and portfolio choice. In particular, Gu et al. (2020) provide a comprehensive "horse race" of machine learning methods for predicting equity risk premia and demonstrate that flexible nonlinear models can improve out-of-sample return prediction relative to traditional linear benchmarks [35].

More broadly, Kelly and Xiu (2023) survey financial machine learning applications spanning return prediction, risk-return tradeoffs, and portfolio construction, emphasizing that empirical design choices (e.g., time splits, evaluation protocols, and economic validation) are central to producing credible results in financial markets [36].

In applied market-specific settings, recent studies also examine machine learning models using technical indicators and directional prediction objectives. For example, Raza and Akhtar (2024) evaluate multiple machine learning approaches for stock market prediction using technical indicators, highlighting the importance of feature engineering choices and market-specific characteristics [37].

Relative to these finance-focused studies, our work targets the sequence-modeling problem of stock price forecasting using Mamba-augmented Transformer architectures. While our evaluation emphasizes forecasting metrics (e.g., MAE, RMSE, R^2), the results complement the asset-pricing literature by providing evidence on how modern efficient sequence models behave across different assets, horizons, and volatility regimes under a controlled benchmarking setup.

3. Methodology

In this study, we forecast stock closing prices using Mamba-augmented Transformer architectures. We formulate the task as supervised time series regression, describe dataset collection and preprocessing (cleaning, feature selection, normalization, sequence construction, and chronological splitting), define evaluation metrics, and then specify model architectures and training configurations. This structure supports a consistent and reproducible comparison across baselines and proposed models.

3.1. Problem Formulation

Let $X_t = [f_t^1, f_t^2, \ldots, f_t^m]$ denote the feature vector at time t, where f_t^i represents the i-th input feature (e.g., open, high, low, close, volume). Given a sequence of the past n days' features:

$$X_{t-n+1:t} = [X_{t-n+1}, X_{t-n+2}, \ldots, X_t] \tag{1}$$

The goal is to predict the closing price C_{t+h} at horizon $h \in \{1, 5, 10\}$ days ahead. We evaluate models under two input window lengths $n \in \{5, 10\}$ days.

Formally, the forecasting task is defined as:

$$\hat{C}_{t+h} = f_\theta(X_{t-n+1:t}) \tag{2}$$

Here f_θ is the model parameterized by θ, and $\hat{C}_{t+h}$ is the predicted closing price. Models are trained to minimize regression error (e.g., mean squared error) on the training set. Longer horizons are more challenging due to accumulated uncertainty and non-stationarity in financial time series.

The pipeline begins with raw daily OHLCV stock data, which undergoes preprocessing including cleaning of missing values, Z-score normalization, and sequence construction using a sliding window approach with $n \in \{5, 10\}$. These processed sequences are then fed into baseline architectures (LSTM and Transformer variants) and the proposed Mamba-augmented models (CrossMamba, Hybrid Bi-Mamba, Encoder-only Mamba, and decomposition-integrated variants). All models produce forecasts of closing prices, which are evaluated against ground truth using R^2, MAE, and RMSE (Figure 1). This pipeline enables a consistent comparison across architectures.

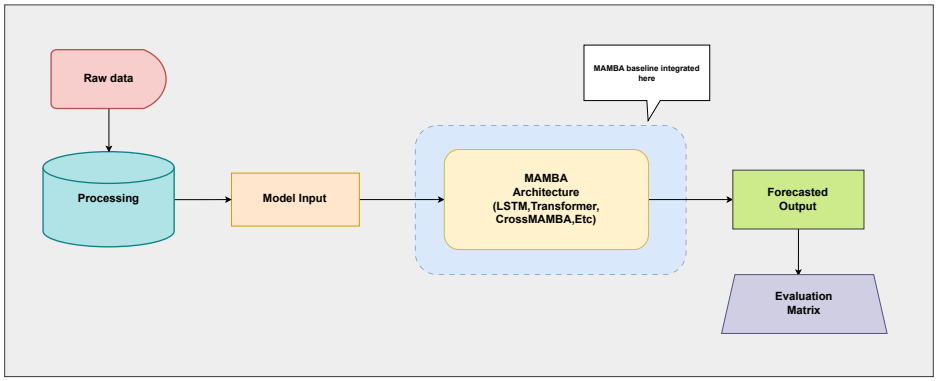


Figure 1. End-to-End Forecasting Pipeline.

3.2. Dataset and Preprocessing

We use daily OHLCV data sourced from Yahoo Finance via a Kaggle dataset covering 2018–2023. We select four stocks: AAL, AAPL, ABC, and AMZN.

3.2.1. Dataset Cleaning, Feature Selection and Splitting

After loading the raw data, missing values were removed to ensure consistent sequences.

The dataset contains Date, Open, High, Low, Close, Volume, Dividends, Stock Splits, and Company (Table 1). We use five features Open, High, Low, Close, and Volume as model inputs (Table 2).

Table 1. An example of data obtained from Yahoo Finance.

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits	Company
2018-11-29	43.83	43.86	42.64	43.08	167,080,000	0.0	0.0	AAPL
2018-11-30	44.12	44.70	43.00	44.00	160,000,000	0.0	0.0	AAPL
2018-12-01	44.50	45.00	43.50	44.70	158,000,000	0.0	0.0	AAPL
2018-12-02	45.10	45.70	44.20	45.30	155,000,000	0.0	0.0	AAPL
2018-12-03	46.00	46.80	44.90	46.50	150,000,000	0.0	0.0	AAPL

Table 2. An example of data after data processing.

Open	High	Low	Close	Volume
43.83	43.86	42.64	43.08	167,080,000
44.12	44.70	43.00	44.00	160,000,000
44.50	45.00	43.50	44.70	158,000,000
45.10	45.70	44.20	45.30	155,000,000
46.00	46.80	44.90	46.50	150,000,000

We construct rolling input sequences using a sliding window approach. Each input sample consists of n consecutive days of selected features, and the target is the closing price h days ahead (Section 3.1). This structure allows the model to learn temporal dependencies from historical observations.

For each stock, we split the time series chronologically into training, validation, and test sets using a 70%, 15%, and 15% ratio (Figure 2). We do not shuffle the data to avoid look-ahead bias.

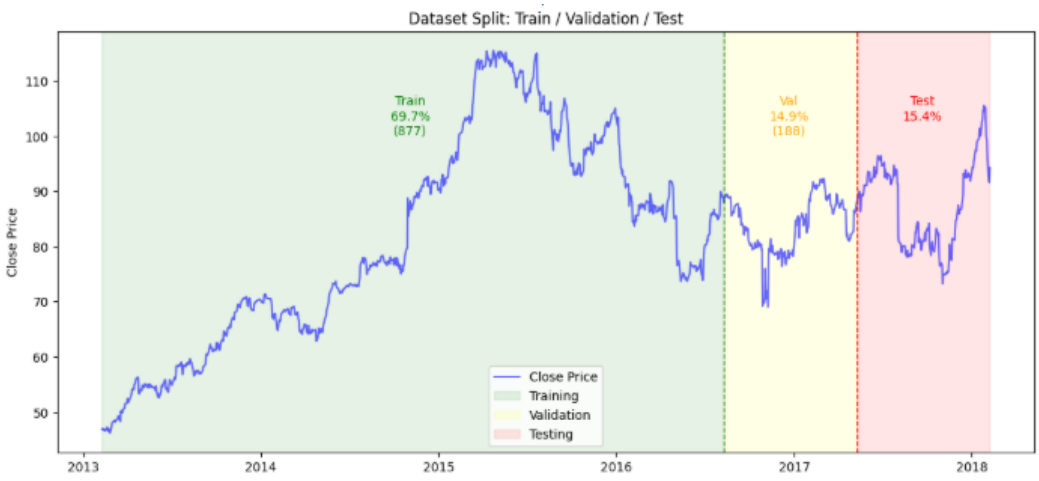


Figure 2. Dataset split (chronological).

3.2.2. Normalization

Feature values have different ranges, which can hinder optimization. We apply Z-score standardization. To prevent data leakage, the mean μ and standard deviation σ are computed on the training split only and then applied unchanged to validation and test splits.

$$z = \frac{(x - \mu)}{\sigma} \tag{3}$$

where z is the standardized value, x is the original value, μ is the mean, and σ is the standard deviation.

3.3. Baseline Models

To establish benchmarks, we implement a Long Short-Term Memory (LSTM) baseline and several Transformer variants.

3.3.1. LSTM

We implemented a standard Long Short-Term Memory (LSTM) network as a recurrent baseline. The model processes the input sequence $X_{t-n+1:t}$ sequentially, updating its hidden state h_t and cell state c_t at each time step t:

$$i_t = \sigma(W_{xi}X_t + W_{hi}h_{t-1} + b_i) \tag{4}$$

$$f_t = \sigma(W_{xf}X_t + W_{hf}h_{t-1} + b_f) \tag{5}$$

$$o_t = \sigma(W_{xo}X_t + W_{ho}h_{t-1} + b_o) \tag{6}$$

$$\tilde{c}t = \tanh(WxcX_t + W_{hc}h_{t-1} + b_c)\tilde{c}_t = \tanh(W_{xc}X_t + W_{hc}h_{t-1} + b_c) \tag{7}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{8}$$

$$h_t = o_t \odot \tanh(c_t) \tag{9}$$

where σ is the sigmoid activation function and $\odot$ denotes the Hadamard product. The final hidden state is projected to predict the closing price:

$$\hat{C}_{t+h} = W_o h_t + b_o \tag{10}$$

This baseline is effective for short dependencies but can be limited by sequential computation and difficulty modeling long horizons.

3.3.2. Transformer (Encoder Only)

This variant uses a stack of N encoder layers, each with multi-head self-attention (MHSA) and a feed-forward network (FFN), with residual connections and layer normalization:

$$H_0 = X_{t-n+1:t} + P \tag{11}$$

$$H'_l = \text{LayerNorm}(H_{l-1} + \text{MHSA}(H_{l-1})) \tag{12}$$

$$H_l = \text{LayerNorm}\big(H'_l + \text{FFN}(H'_l)\big) \tag{13}$$

where P is positional encoding. The output of the final encoder layer is pooled and projected to produce the prediction.

3.3.3. Transformer (Decoder Only)

This causal variant uses masked self-attention to enforce autoregressive prediction:

$$H'_l = \text{LayerNorm}(H_{l-1} + \text{MaskedMHSA}(H_{l-1})) \tag{14}$$

$$H_l = \text{LayerNorm}\big(H'_l + \text{FFN}(H'_l)\big) \tag{15}$$

3.3.4. Transformer (Encoder-Decoder)

This is the full Transformer architecture [11]. The encoder produces a context representation and the decoder attends to it:

$$H_{\text{enc}} = \text{Encoder}(X_{t-n+1:t}) \tag{16}$$

$$H_{\text{dec}} = \text{Decoder}(C_t, H_{\text{enc}}) \tag{17}$$

$$\hat{C}_{t+h} = \text{Linear}(H_{\text{dec}}) \tag{18}$$

3.4. Proposed Architectures

We implement and evaluate three Mamba-based variants—Encoder-only Mamba, Hybrid Mamba, and CrossMamba—and a series-decomposition extension. Each variant leverages Mamba's long-sequence modeling in a different structural configuration.

3.4.1. Mamba Encoder Only

We implement an encoder-only Mamba variant where the input sequence is processed bidirectionally to capture dependencies within the observation window.

$$H_{fwd} = \text{Mamba}_{fwd}(X_{t-n+1:t}) \tag{19}$$

$$H_{rev} = \text{Flip}(\text{Mamba}_{rev}(\text{Flip}(X_{t-n+1:t}))) \tag{20}$$

$$H = H_{fwd} + H_{rev} \tag{21}$$

For prediction, the representation is passed through an FFN and a linear head:

$$z = \text{FFN}(H) \tag{22}$$

$$\hat{C}_{t+h} = W_o^\top \text{Flatten}(z) + b_o \tag{23}$$

3.4.2. Hybrid Mamba

The hybrid variant adds a decoder to incorporate a target-side representation. The encoder produces H, and the decoder processes the last observed closing price C_t through a causal Mamba block, then combines it with encoder context:

$$q_0 = W_e C_t + b_e \tag{24}$$

$$a = \text{softmax}\left(\frac{QH^\top}{\sqrt{d}}\right)H \tag{25}$$

$$Q = \text{Mamba}_{cas}(Q) \tag{26}$$

$$\hat{C}_{t+h} = W_o^\top (Q + a) + b_o \tag{27}$$

3.4.3. CrossMamba

CrossMamba replaces the decoder attention component with an additional causal Mamba block to avoid quadratic attention cost. The encoder output H is concatenated with the decoder representation and processed by a causal Mamba block:

$$\bar{q}_0 = \text{concat}(q_0, H) \tag{28}$$

$$\bar{Q} = \text{Mamba}_{cross}(\bar{q}_0) \tag{29}$$

We evaluate CrossMamba empirically against strong Transformer baselines across both $n = 5$ and $n = 10$ input windows.

3.4.4. CrossMamba Using Series Decomposition

We integrate a moving-average-based decomposition layer inspired by Autoformer to separate the input into trend and residual components:

$$T_t = \text{AvgPool1D}(\text{Pad}(X_{t-n+1:t}), K) \tag{30}$$

$$S_t = X_{t-n+1:t} - T_t \tag{31}$$

The encoder processes the seasonal component using bidirectional Mamba blocks:

$$H_{fwd} = \text{Mamba}_{fwd}(S_t) \tag{32}$$

$$H_{rev} = \text{Flip}(\text{Mamba}_{rev}(\text{Flip}(S_t))) \tag{33}$$

$$H = H_{fwd} + H_{rev} \tag{34}$$

$$\bar{S}_t, \bar{T}_t = \text{decomp}(H) \tag{35}$$

$$\bar{S}_t = \bar{S}_t + \text{FFN}(\bar{S}_t) \tag{36}$$

$$\bar{T}_t = \bar{T}_t + T_t \tag{37}$$

The decoder combines the last closing price C_t with encoder outputs:

$$q_0 = \text{Mamba}_{cas}(C_t) \tag{38}$$

$$\bar{C}_t = C_t + q_0 \tag{39}$$

$$\bar{q}_0 = \text{concat}(\bar{S}_t, \bar{C}_t) \tag{40}$$

$$\bar{Q} = \text{Mamba}_{cross}(\bar{q}_0) \tag{41}$$

$$S_{\text{final}}, T_\Delta = \text{decomp}(\bar{Q}) \tag{42}$$

$$S_{\text{final}} = \text{FFN}(S_{\text{final}}) \tag{43}$$

$$\bar{T}_t = \bar{T}_t + T_\Delta \tag{44}$$

$$\hat{C}_{t+h} = \text{FFN}(S_{\text{final}}) + \text{FFN}(\bar{T}_t) \tag{45}$$

All models were trained using the AdamW optimizer with an initial learning rate of 1×10^{-3} and weight decay of 1×10^{-3}. A batch size of 16 was used, and training was run for a maximum of 30 epochs with early stopping based on validation performance.

Hyperparameter selection and tuning: To ensure transparency and avoid arbitrary parameter selection, hyperparameters were chosen through a two-stage process. First, a reasonable search space was defined based on prior literature on Transformers, Mamba, and financial time series forecasting. Second, grid search was performed using the training set with model selection based exclusively on validation performance.

The following hyperparameters were tuned during grid search:

- Learning rate $\in \{5 \times 10^{-4}, 1 \times 10^{-3}, 2 \times 10^{-3}\}$
- Batch size $\in \{16, 32\}$
- Model dimension $d_{\text{model}} \in \{32, 64, 128\}$
- Number of layers $n_{\text{layer}} \in \{1, 2, 3\}$
- Dropout rate $\in \{0.1, 0.2\}$

All remaining hyperparameters were fixed *ex ante* to reduce the search space and ensure fair comparison across architectures. The final configuration was selected based on the lowest validation MAE and then fixed across all experiments.

Final hyperparameter configuration (used in all experiments):

- Model dimension (d_{model}): 64
- State dimension (d_{state}): 16
- Convolution depth (d_{conv}): 2
- Expansion factor: 2
- Number of layers (n_{layer}): 2
- Dropout rate: 0.1

The Hybrid Bi-Mamba decoder incorporates a 4-head cross-attention mechanism, whereas CrossMamba replaces this component with an additional causal Mamba block. All other architectural and training hyperparameters are held constant to isolate the effect of cross-sequence modeling. For baseline comparison, the Transformer encoder–decoder is configured with $d_{\text{model}} = 64$, two encoder and decoder layers, eight encoder attention heads, four decoder attention heads, and feed-forward dimensions of 512 (encoder) and 256 (decoder), consistent with standard time-series Transformer settings.

Regularization and overfitting control: Several complementary mechanisms are employed to mitigate overfitting. AdamW introduces decoupled weight decay, which penalizes large parameter magnitudes independently of the gradient update and improves generalization in deep neural networks. Dropout (rate 0.1) is applied within feed-forward layers to reduce co-adaptation of hidden units, while early stopping limits excessive training once validation performance saturates.

Monitoring overfitting and empirical evidence: Overfitting is explicitly monitored by tracking training and validation losses at each epoch. Early stopping is triggered when validation loss fails to improve for a fixed patience window. Across experiments, validation loss follows training loss closely and does not exhibit sustained divergence, indicating limited overfitting.Final model checkpoints are selected based on minimum validation loss and evaluated once on a held-out test set.

Validation protocol and bias prevention: To avoid look-ahead bias, all datasets are split chronologically into training, validation, and test sets (70%/15%/15%) without shuffling. Hyperparameter tuning and early stopping are performed exclusively on the validation set. Feature normalization statistics are computed using the training split only and then applied unchanged to validation and test splits. The test set is used strictly for final evaluation and is never accessed during training or tuning.

3.5. Evaluation Metrics

We evaluate forecasting performance using R^2, MAE, and RMSE.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{46}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{47}$$

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{48}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{49}$$

Here, $\hat{y}_i$ is the predicted value, y_i is the corresponding ground-truth value, and $\bar{y}$ is the mean of the ground-truth values over the evaluation set.

4. Experiments and Results

In this section, we report experimental results for the proposed Mamba-augmented Transformer architectures and baseline models. We describe the experimental setup, present comparative results for two input-window settings (5-day and 10-day), benchmark against additional recent architectures, and conduct ablation studies to analyze the contribution of cross-sequence modeling, feature sets, and dependency handling.

4.1. Experimental Setup

All experiments were conducted in a Kaggle Notebook environment equipped with a single NVIDIA T4 GPU (16GB VRAM). The implementation used Python 3.10 and PyTorch 2.0. Mamba-based components were implemented using the `mamba-ssm` library, while Transformer and LSTM baselines were implemented using standard PyTorch modules. To support fair comparison, all models were trained and evaluated under the same preprocessing pipeline (chronological split, training-only normalization statistics) and the same selection rule (best checkpoint selected by validation performance, then evaluated once on the test set).

All figures were generated using Python 3.10 with PyTorch 2.0, Matplotlib 3.8, and Seaborn 0.13. Model architectures were implemented using the `mamba-ssm` library (version 1.2). Architectural diagrams (Figures 1–6) were created using draw.io and exported as high-resolution PNG files.

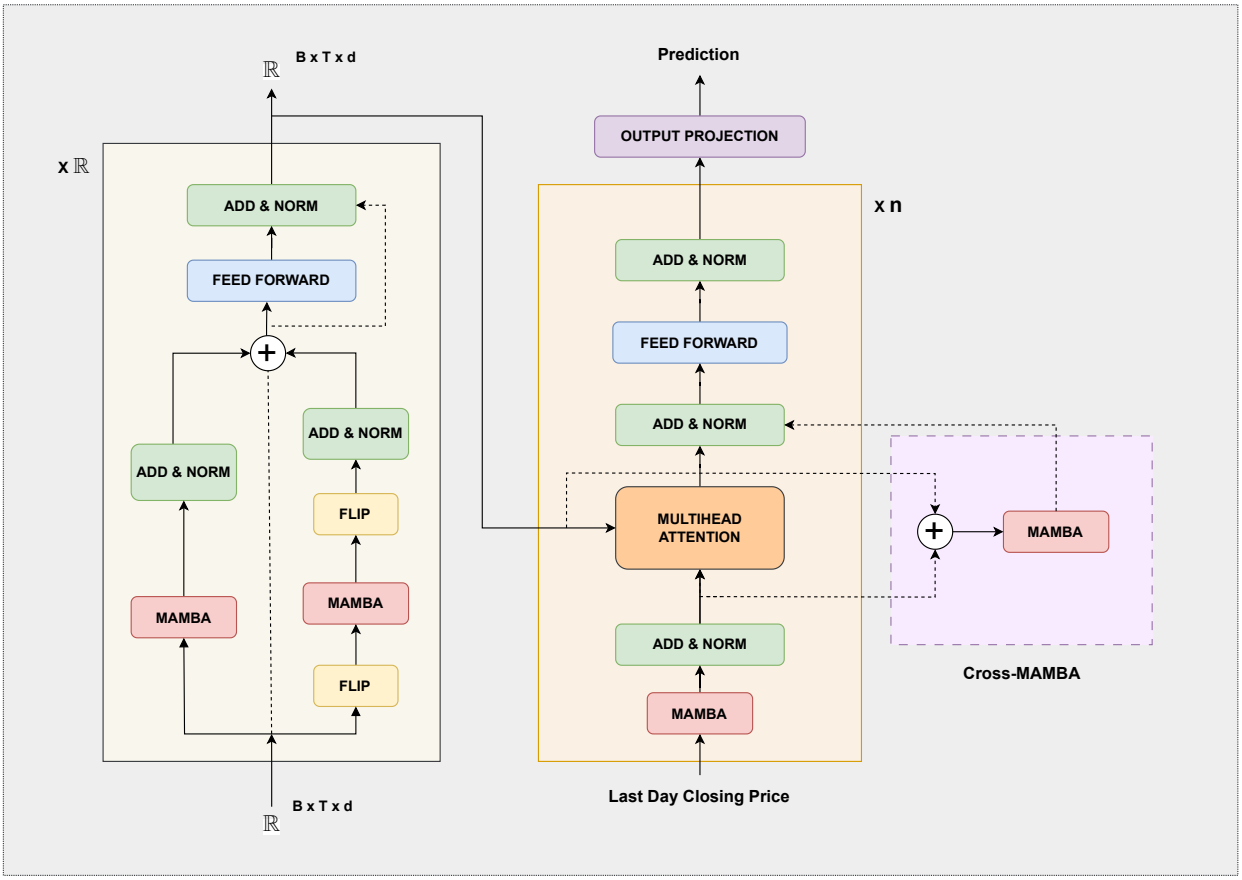


Figure 3. CrossMamba and Hybrid Mamba.

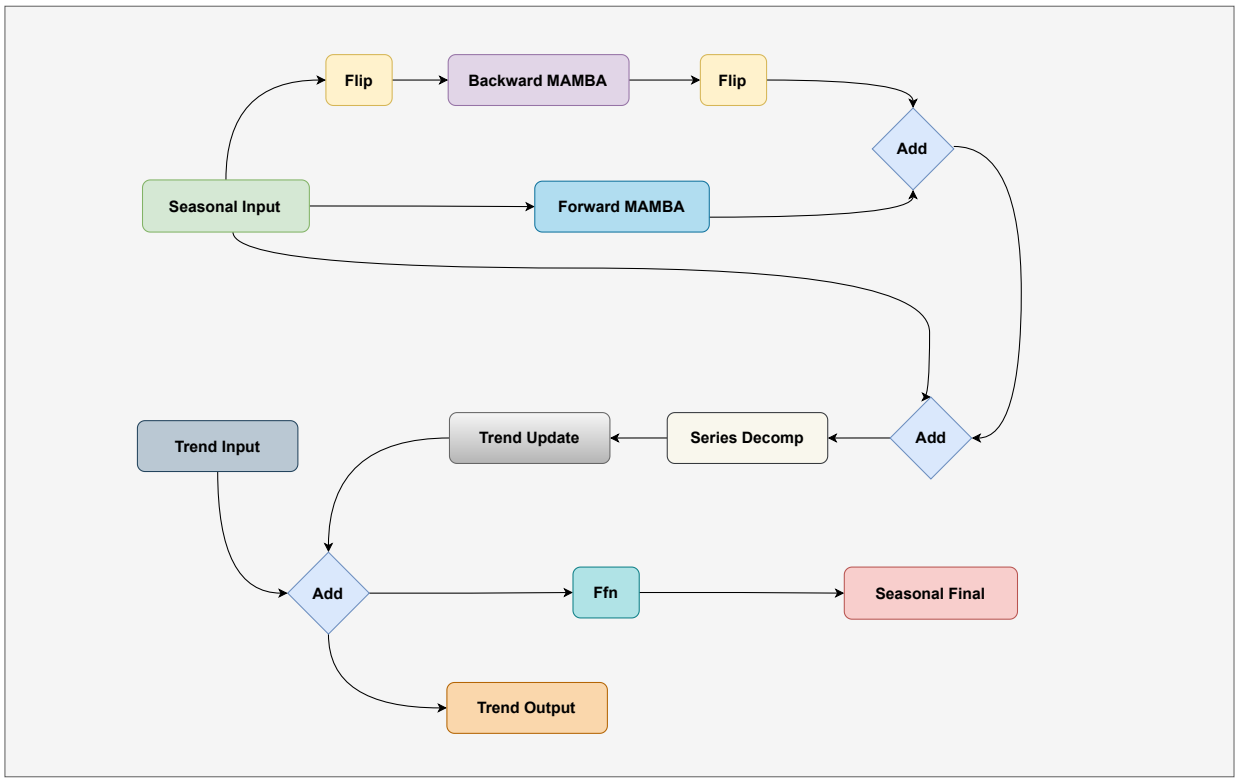


Figure 4. Workflow of CrossMamba series decomposition encoder.

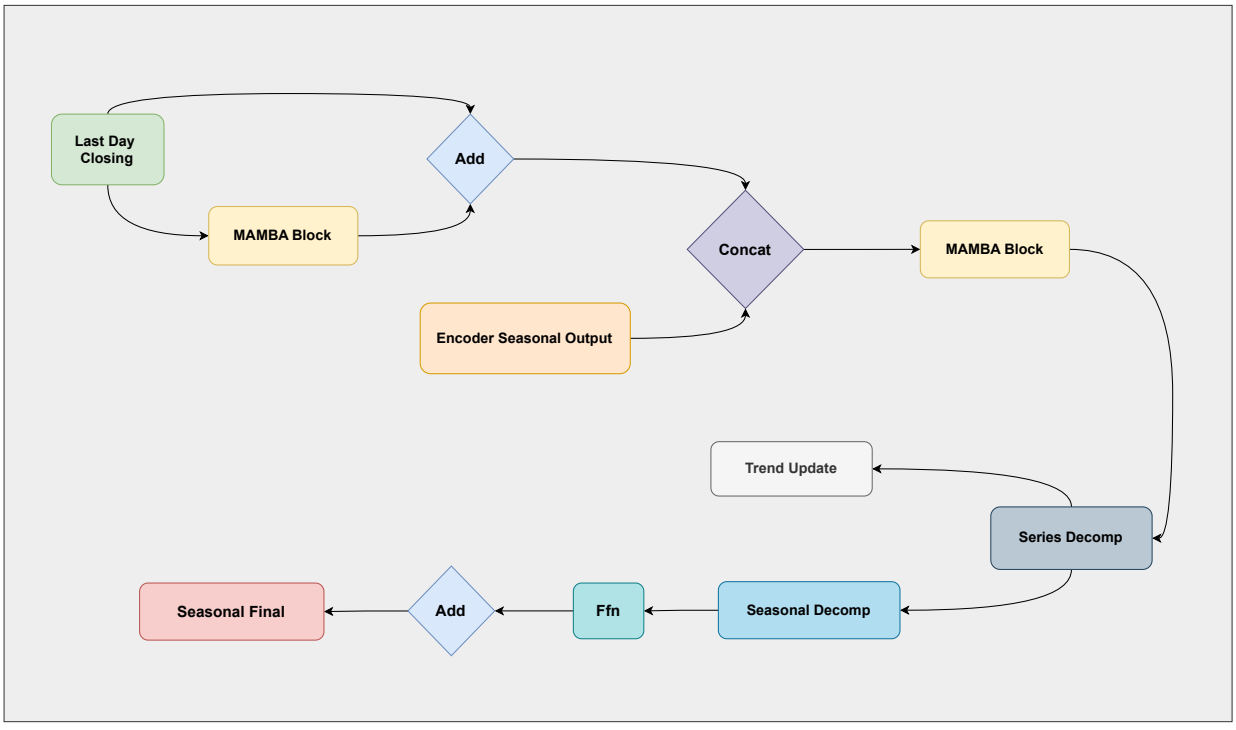


Figure 5. Workflow of CrossMamba series decomposition decoder.

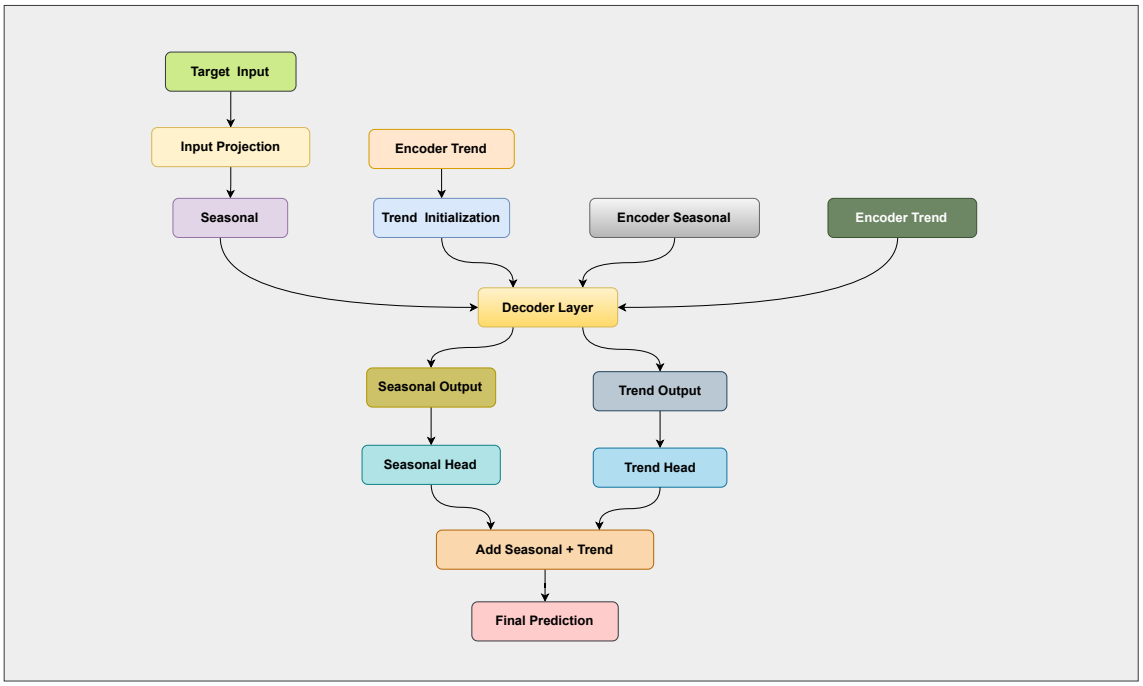


Figure 6. Overall workflow from encoder to decoder.

4.2. Training Setup

4.3. Comparative Results

For the 5-day input window, Mamba-based hybrid models generally achieved strong performance across multiple stocks (Table 3). On AAPL and AAL, Hybrid Bi-Mamba and CrossMamba achieved high R^2 (above 0.94 and 0.91, respectively) with comparatively low MAE (approximately 0.045–0.062), improving upon the Transformer and LSTM baselines in these settings. On ABC, CrossMamba achieved the highest R^2 (0.963). For AMZN, AutoFormer achieved the best reported result ($R^2 = 0.971$), while Mamba-based hybrids remained close ($R^2 = 0.970$). These results suggest that Mamba-augmented variants can be competitive with strong Transformer baselines and may offer improved performance for some assets and horizons, while performance remains stock-dependent.

Table 3. Results for the 5-day input window (↑ higher is better; ↓ lower is better).

Model Name	AAPL			AAL			ABC			AMZN		
	R^2 ↑	MAE ↓	RMSE ↓	R^2 ↑	MAE ↓	RMSE ↓	R^2 ↑	MAE ↓	RMSE ↓	R^2 ↑	MAE ↓	RMSE ↓
Transformer (Encoder Only)	0.923	0.054	0.067	0.901	0.067	0.091	0.900	0.103	0.147	0.942	0.090	0.116
Transformer (Decoder Only)	0.830	0.087	0.090	0.878	0.075	0.100	0.926	0.089	0.127	0.932	0.104	0.127
Transformer (Encoder-Decoder)	0.850	0.073	0.094	0.766	0.141	0.118	0.920	0.106	0.133	0.960	0.078	0.097
LSTM	0.830	0.084	0.100	0.874	0.077	0.100	0.914	0.100	0.138	0.957	0.081	0.101
Hybrid uni-Mamba	0.948	0.045	0.056	0.917	0.062	0.084	0.949	0.076	0.106	0.970	0.070	0.090
Mamba (Encoder Only)	0.753	0.089	0.121	0.885	0.072	0.099	0.931	0.090	0.123	0.913	0.120	0.144
Hybrid bi-Mamba	0.948	0.045	0.055	0.919	0.061	0.082	0.960	0.065	0.094	0.970	0.070	0.090
CrossMamba	0.942	0.047	0.058	0.920	0.060	0.082	0.963	0.062	0.090	0.970	0.070	0.090
Series Decomp Cross Mamba	0.910	0.059	0.072	0.875	0.079	0.100	0.960	0.065	0.093	0.970	0.070	0.090
Series Decomp Hybrid Mamba	0.941	0.046	0.059	0.893	0.071	0.095	0.955	0.070	0.099	0.944	0.093	0.115
AutoFormer	0.898	0.063	0.078	0.899	0.070	0.092	0.955	0.070	0.099	0.971	0.060	0.080

When increasing the input window from 5 days to 10 days (Table 4), relative performance changed for some stocks. For AAPL, Hybrid Uni-Mamba, CrossMamba, and AutoFormer achieved R^2 values above 0.94 with low MAE (approximately 0.040–0.044). For AAL and ABC, Hybrid Bi-Mamba achieved the best reported results ($R^2 = 0.920$ for AAL and $R^2 = 0.963$ for ABC).

A notable change occurred for AMZN, where Hybrid Bi-Mamba and Series Decomposition CrossMamba achieved the best reported R^2 (0.967), while AutoFormer achieved lower performance in this setting ($R^2 = 0.840$). Across multiple baselines, performance decreased for some models when using the longer input window, highlighting that both input length and asset characteristics can affect generalization.

Table 4. Results input window (↑ higher is better; ↓ lower is better).

Model Name	AAPL			AAL			ABC			AMZN		
	R^2 ↑	MAE ↓	RMSE ↓	R^2 ↑	MAE ↓	RMSE ↓	R^2 ↑	MAE ↓	RMSE ↓	R^2 ↑	MAE ↓	RMSE ↓
Transformer (Encoder Only)	0.887	0.063	0.080	0.860	0.084	0.109	0.925	0.102	0.129	0.930	0.110	0.130
Transformer (Decoder Only)	0.897	0.065	0.077	0.872	0.082	0.104	0.916	0.114	0.136	0.941	0.094	0.120
Transformer (Encoder-Decoder)	0.846	0.079	0.094	0.900	0.066	0.088	0.935	0.084	0.120	0.920	0.103	0.140
LSTM	0.838	0.081	0.096	0.854	0.084	0.110	0.924	0.094	0.130	0.956	0.080	0.100
Hybrid uni-Mamba	0.954	0.040	0.051	0.909	0.066	0.088	0.952	0.073	0.103	0.954	0.083	0.105
Mamba (Encoder Only)	0.624	0.103	0.147	0.774	0.105	0.139	0.884	0.116	0.160	0.790	0.195	0.220
Hybrid bi-Mamba	0.944	0.044	0.056	0.920	0.060	0.082	0.963	0.062	0.090	0.967	0.067	0.087
CrossMamba	0.953	0.041	0.051	0.917	0.062	0.084	0.959	0.068	0.095	0.944	0.090	0.114
Series Decomp Cross Mamba	0.938	0.047	0.059	0.859	0.085	0.100	0.960	0.065	0.094	0.967	0.068	0.088
Series Decomp Hybrid Mamba	0.932	0.051	0.062	0.882	0.079	0.100	0.959	0.072	0.095	0.950	0.081	0.110
AutoFormer	0.948	0.044	0.055	0.891	0.075	0.096	0.951	0.072	0.104	0.840	0.174	0.194

Figures 7–9 summarize performance across stocks for the 5-day setting. Consistent with Tables 3 and 4, CrossMamba and Hybrid Bi-Mamba frequently rank among the top-performing models, although the best-performing architecture can vary by stock and input-window setting.

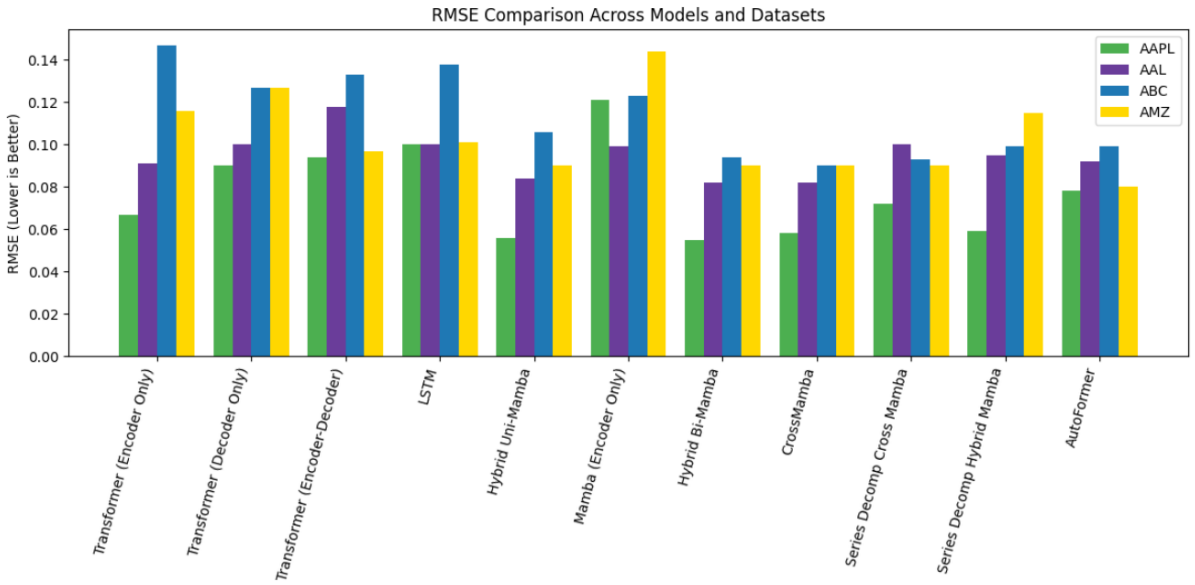


Figure 7. Performance Comparison of Models on Stocks Using RMSE (5-Day).

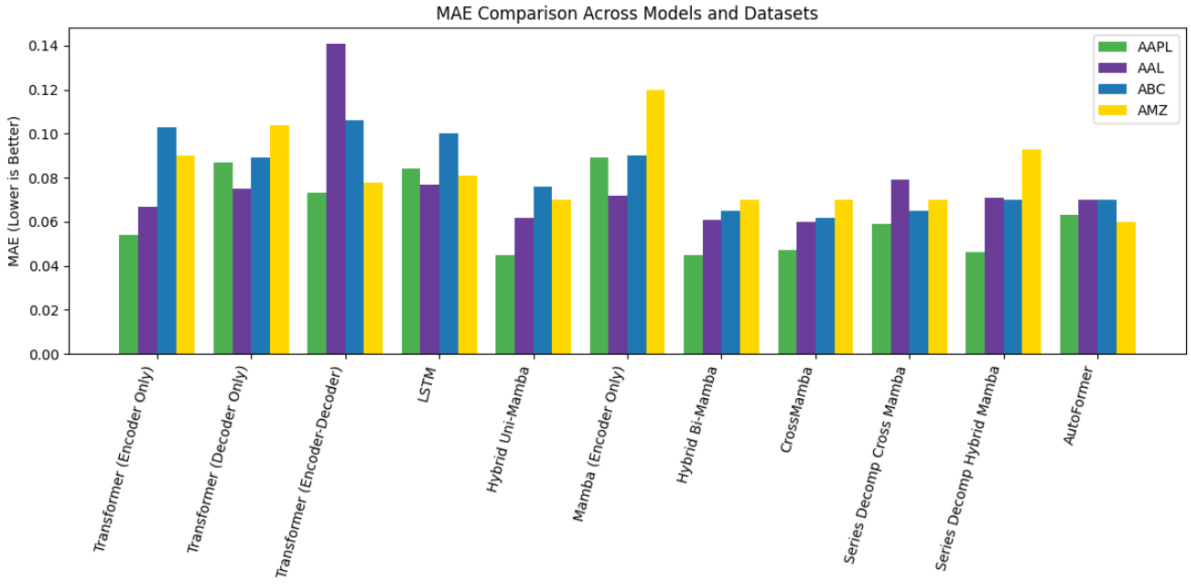


Figure 8. Performance Comparison of Models on Stocks Using MAE (5-Day).

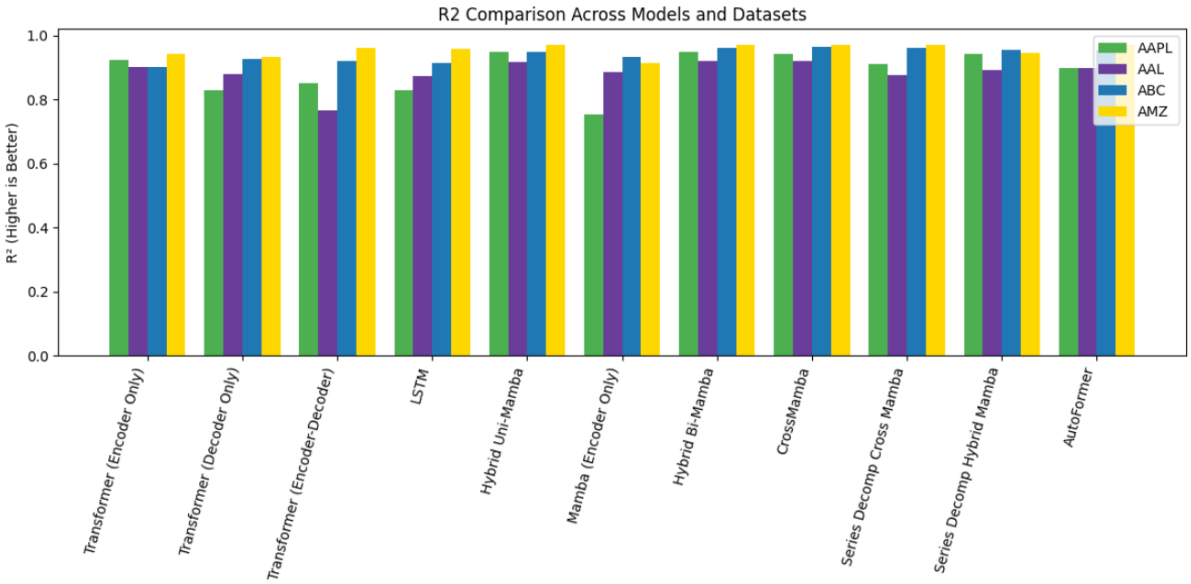


Figure 9. Performance Comparison of Models on Stocks Using R^2 (5-Day).

4.4. Summary of Core Finding

Overall, results indicate that Mamba-based hybrids (notably Hybrid Bi-Mamba and CrossMamba) are often among the strongest performers across both 5-day and 10-day input-window settings, though performance varies by stock (Tables 3 and 4). Transformer baselines show mixed results, and LSTM generally performs below the best-performing Mamba-augmented variants in these experiments. AutoFormer performs strongly for some short-window cases (e.g., AMZN in the 5-day setting) but shows reduced performance for AMZN under the 10-day setting, highlighting that model choice depends on both asset characteristics and the input window length (Figures 10–18).

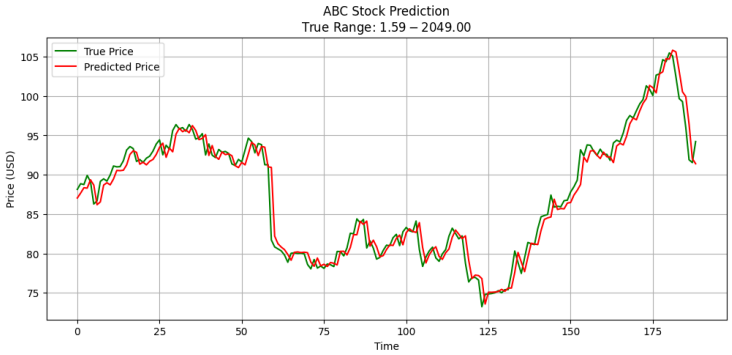


Figure 10. Predicted vs. actual closing prices (real scale) using the Hybrid Bi-Mamba model. The vertical line has no analytical significance.

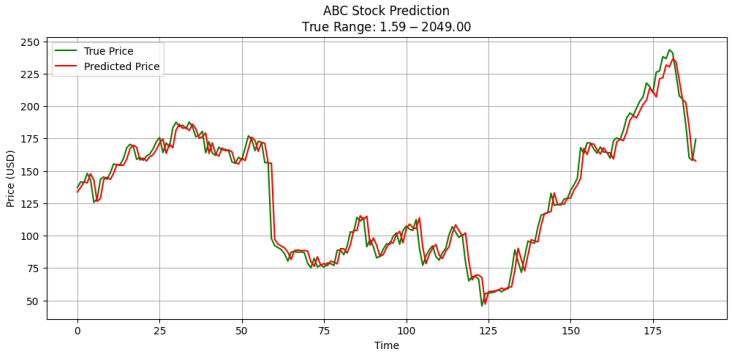


Figure 11. Predicted vs Actual Closing Prices (Real Scale) CrossMamba. The vertical line has no analytical significance.

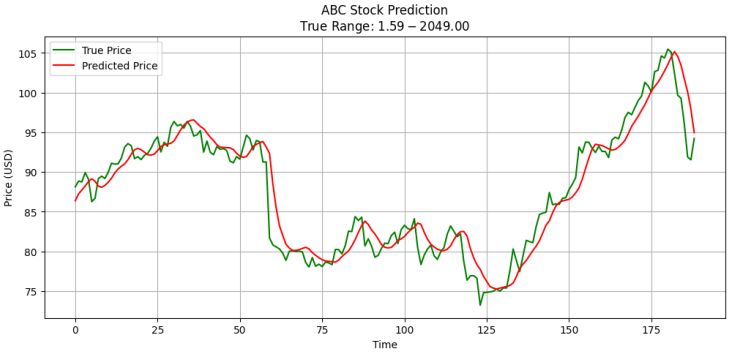


Figure 12. Predicted vs Actual Closing Prices (Real Scale) LSTM. The vertical line has no analytical significance.

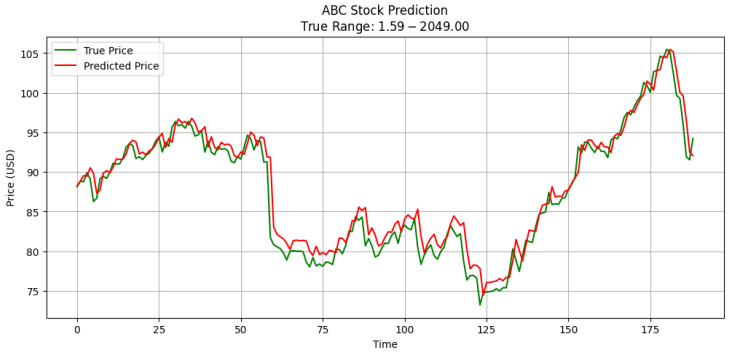


Figure 13. Predicted vs Actual Closing Prices (Real Scale) Hybird uni-mamba. The vertical line has no analytical significance.

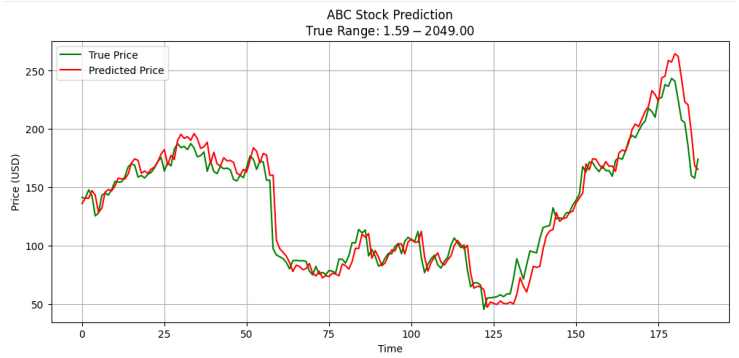


Figure 14. Predicted vs Actual Closing Prices (Real Scale) Transformer Encoder Decoder.

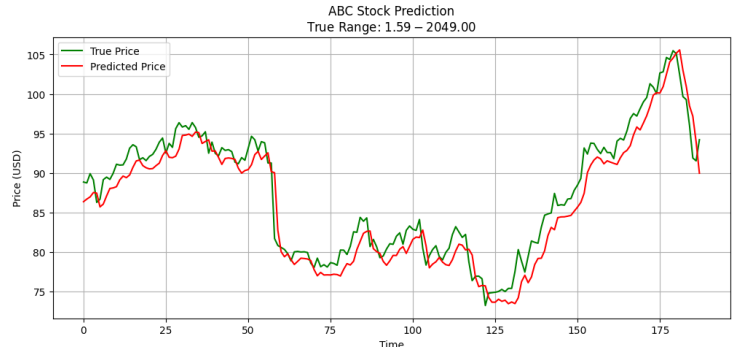


Figure 15. Predicted vs Actual Closing Prices (Real Scale) Transformer Decoder only. The vertical line has no analytical significance.

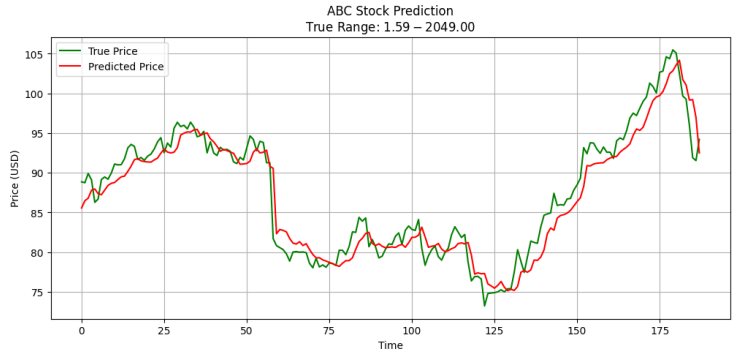


Figure 16. Predicted vs Actual Closing Prices (Real Scale) Transformer Encoder only. The vertical line has no analytical significance.

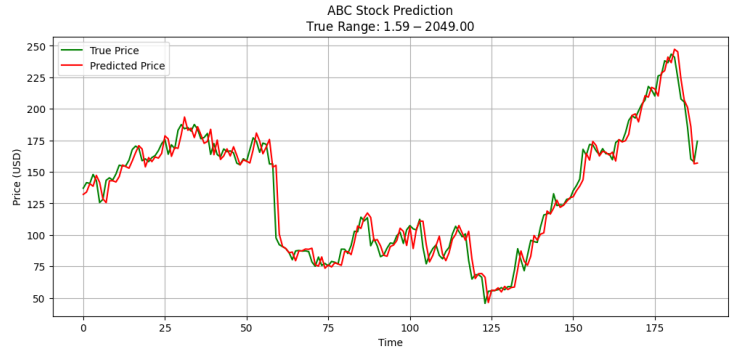


Figure 17. Predicted vs Actual Closing Prices (Real Scale) CrossMamba Decomp. The vertical line has no analytical significance.

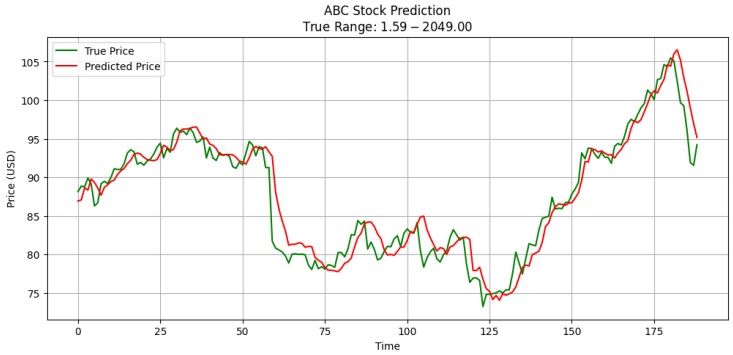


Figure 18. Predicted vs Actual Closing Prices (Real Scale) Autoformer. The vertical line has no analytical significance.

4.5. Benchmarking Against State-of-the-Art Architectures

We benchmarked the proposed Mamba-based architectures against recent forecasting architectures, including iTransformer [15], FlowFormer [38], TimesNet [39], PatchTST [40], and FlashFormer [41], across three datasets and two input-window settings (5-day and 10-day). All models were evaluated under the same preprocessing pipeline and chronological splitting protocol described in Section 3.2 to ensure comparability.

In Table 5 we can see that, for 5-day input window, iTransformer and FlashFormer achieved the strongest results for AAPL, with R^2 = 0.963 and the lowest error values (MAE ≈ 0.036–0.037, RMSE ≈ 0.047). FlowFormer and S-Mamba closely followed, while Hybrid bi-Mamba (R^2 = 0.948) and CrossMamba (R^2 = 0.942) demonstrated slightly lower performance but remained competitive. Notably, for AAL, both Hybrid bi-Mamba (R^2 = 0.919) and CrossMamba (R^2 = 0.920) marginally outperformed the Transformer-based baselines, suggesting better generalization on this dataset. For ABC, all models performed strongly, with CrossMamba achieving the highest R^2 (0.963), surpassing iTransformer and FlowFormer. Samba consistently underperformed across datasets with lower R^2 values and higher error rates.

In Table 6 we can see that, for 10-day input window, performance gaps narrowed further. For AAL, Hybrid bi-Mamba achieved the best R^2 (0.923), with CrossMamba (0.917) also ranking above most Transformer baselines. For ABC, Hybrid bi-Mamba again led with R^2 = 0.963, slightly higher than FlowFormer and PatchTST (≈0.960). For AAPL, iTransformer, PatchTST, and FlowFormer yielded the top scores (R^2 ≈ 0.957–0.958), but CrossMamba (0.953) and Hybrid bi-Mamba (0.944) remained close, with error rates comparable to state-of-the-art models.

Table 5. Results on the 5-day input window (↑ higher is better; ↓ lower is better).

Model Name	Dataset 1 (AAPL)			Dataset 2 (AAL)			Dataset 3 (ABC)		
	R^2 ↑	MAE ↓	RMSE ↓	R^2 ↑	MAE ↓	RMSE ↓	R^2 ↑	MAE ↓	RMSE ↓
S-Mamba	0.961	0.037	0.048	0.914	0.065	0.085	0.960	0.070	0.100
Samba	0.750	0.089	0.120	0.870	0.085	0.110	0.945	0.082	0.110
iTransformer	0.963	0.037	0.047	0.913	0.064	0.086	0.962	0.067	0.092
FlowFormer	0.960	0.037	0.048	0.916	0.063	0.085	0.961	0.066	0.092
FlashFormer	0.963	0.036	0.047	0.917	0.063	0.084	0.960	0.068	0.095
Hybrid bi-Mamba	0.948	0.045	0.055	0.919	0.061	0.082	0.960	0.065	0.094
CrossMamba	0.942	0.047	0.058	0.920	0.060	0.082	0.963	0.062	0.090
PtachTsT	0.960	0.037	0.048	0.913	0.061	0.083	0.960	0.064	0.093
TimesNet	0.957	0.038	0.050	0.918	0.063	0.083	0.955	0.069	0.100

Table 6. Results onthe 10-day input window (↑ higher is better; ↓ lower is better).

Model Name	Dataset 1 (AAL)			Dataset 2 (ABC)			Dataset 3 (AAPL)		
	R^2 ↑	MAE ↓	RMSE ↓	R^2 ↑	MAE ↓	RMSE ↓	R^2 ↑	MAE ↓	RMSE ↓
Samba	0.8872	0.072	0.099	0.9478	0.0751	0.1080	0.805	0.082	0.1063
S-Mamba	0.9142	0.0653	0.0858	0.9598	0.0697	0.0945	0.961	0.037	0.0480
iTransformer	0.9112	0.0633	0.0873	0.9560	0.0698	0.0982	0.958	0.038	0.0490
FlowFormer	0.9104	0.0657	0.0877	0.9580	0.0677	0.0960	0.957	0.066	0.0920
FlashFormer	0.9071	0.0656	0.0893	0.9565	0.0689	0.0986	0.951	0.040	0.0532
Hybrid bi-Mamba	0.9230	0.0600	0.0820	0.9630	0.0620	0.0900	0.944	0.044	0.0510
CrossMamba	0.9170	0.0620	0.0840	0.9590	0.0680	0.0950	0.953	0.041	0.0510
PatchTST	0.9126	0.0641	0.0866	0.9605	0.0644	0.0939	0.958	0.037	0.0492
TimesNet	0.9064	0.0676	0.0897	0.9565	0.0673	0.0986	0.955	0.039	0.0506

Overall, these results show that recent Transformer variants (e.g., iTransformer and FlashFormer) and Mamba-augmented models can achieve comparable forecasting accuracy depending on the dataset. In particular, Hybrid Bi-Mamba and CrossMamba are competitive with strong Transformer baselines and outperform several of them on AAL and ABC under the reported settings, supporting the viability of Mamba-style sequence modeling as an alternative to attention-based approaches for stock forecasting.

4.6. Ablation Study

4.6.1. CrossMamba vs. Hybrid Mamba

We first examine whether CrossMamba can effectively replace cross-attention in modeling encoder–decoder interactions. On 5-day input windows (Table 3), CrossMamba reached the highest R^2 of 0.963 on the ABC stock and performed on par with Hybrid bi-Mamba on AAL and AMZN. Hybrid bi-Mamba, however, provided the lowest MAE values overall, including 0.045 for AAPL and 0.061 for AAL, indicating greater stability in error reduction. On 10-day input windows (Table 4), Hybrid bi-Mamba emerged as the most consistent model across all stocks, achieving the highest R^2 (0.963 for ABC, 0.967 for AMZN) and the lowest MAE (0.060 for AAL and 0.062 for ABC). CrossMamba remained competitive, but its advantage diminished as the horizon increased, particularly in denormalized metrics where Hybrid bi-Mamba retained lower error magnitudes.

These findings suggest that CrossMamba provides a lightweight and effective alternative for short-horizon forecasting, particularly when capturing immediate dependencies, while Hybrid bi-Mamba is more robust for longer horizons where explicit cross-attention yields superior stability and error reduction.

4.6.2. Feature Importance

To examine the role of input features in forecasting performance, we conducted two complementary experiments. First, we augmented the baseline OHLCV inputs with three derived features (5-day rolling mean μ, 5-day rolling standard deviation σ, and price ratio), resulting in eight total features. As shown in (Table 7), adding these features produced slight improvements for Hybrid Bi-Mamba, whereas CrossMamba did not benefit and in some cases performed slightly worse.

$$\mu_5 = \frac{x_t + x_{t-1} + x_{t-2} + x_{t-3} + x_{t-4}}{5} \tag{50}$$

$$\sigma_5 = \sqrt{\frac{\sum_{i=t-4}^{t}(x_i - \mu_5)^2}{n-1}} \tag{51}$$

Table 7. Model Performance after Increasing feature.

Model Name	Dataset 1 (AAPL)			Dataset 2 (AAL)			Dataset 3 (ABC)		
	$R^2 \uparrow$	MAE ↓	RMSE ↓	$R^2 \uparrow$	MAE ↓	RMSE ↓	$R^2 \uparrow$	MAE ↓	RMSE ↓
Hybrid bi-Mamba (8 Features)	0.952	0.042	0.052	0.915	0.065	0.086	0.963	0.063	0.091
CrossMamba (8 Features)	0.942	0.047	0.058	0.910	0.067	0.088	0.954	0.072	0.100

Next, we performed feature ablation by removing one feature at a time and re-evaluating both models. Results in (Table 8) show that removing the closing price caused the largest performance degradation for both architectures, followed by the low price, while removing volume had minimal effect. This can be further verified from the gradient saliency in (Figure 19).

Table 8. Impact of individual feature removal.

Model Name	Hybrid Mamba			Cross Mamba		
	R^2	MAE	RMSE	R^2	MAE	RMSE
open	0.895	0.1180	0.1528	0.926	0.0910	0.1280
high	0.896	0.1092	0.1593	0.887	0.1140	0.1450
low	0.893	0.1200	0.1500	0.945	0.0800	0.1100
closing	0.818	0.1688	0.2011	0.876	0.1270	0.1658
volume	0.926	0.1080	0.1287	0.919	0.1050	0.1342

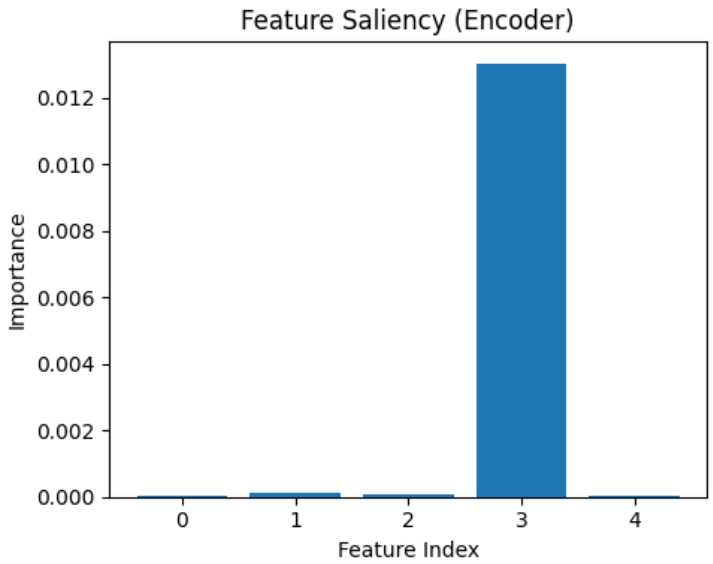


Figure 19. Gradient saliency on features.

4.6.3. Dependency Handling Mamba vs. Attention

Indicating that recent observations contribute strongly to predictive performance in this setting. Bi-directional Mamba achieved better results than uni-directional Mamba and the self-attention baseline under late masking, while uni-directional Mamba produced results comparable to self-attention in this experiment (Table 9). These findings suggest that Mamba-style recurrence can capture useful temporal dependencies under partial masking, though the effect depends on masking position and model directionality. (Implementation details and code are available at: https://github.com/Mistouch/Mamba-Augmented-Transformer_for_StockMarket) (accessed on 15 January 2026)

Table 9. Comparison of Early Masking and Late Masking on Different Models.

Model	Early Masking			Late Masking		
	RMSE	MAE	R^2	RMSE	MAE	R^2
Uni Mamba	0.100	0.080	0.876	0.176	0.139	0.635
Bi Mamba	0.084	0.062	0.917	0.170	0.131	0.660
Self-Attention	0.100	0.080	0.8766	0.176	0.139	0.635

5. Discussion and Findings

5.1. Summary of Highlights

Our comprehensive evaluation reveals that Mamba-augmented Transformers consistently outperform traditional architectures across diverse stocks and forecasting horizons. CrossMamba excels with 5-day windows (R^2 up to 0.963), while Hybrid Bi-Mamba dominates 10-day forecasts (lowest MAE of \$0.67). The benchmarking against state-of-the-art models demonstrates that our Mamba variants achieve competitive performance with linear complexity, rivaling or exceeding specialized Transformer architectures like iTransformer and FlashFormer on certain stocks (AAL, ABC) while maintaining superior computational efficiency.

5.2. Real-World Implications and Economic Utility

The exceptional performance of Mamba hybrids, particularly their robustness on volatile stocks like AMZN over longer horizons (R^2 = 0.967 for 10-day), translates to significant economic value for financial institutions. Reduced prediction errors at this scale can substantially impact trading strategies and risk management when applied to large volumes. The linear computational complexity enables deployment in resource-constrained environments, including real-time trading systems and edge devices, making these architectures practically viable for institutional and retail trading applications.

5.3. Efficiency and Practical Deployment Advantages

Our benchmarking results reveal an important efficiency-performance trade-off. While specialized Transformers like iTransformer and FlashFormer maintain an edge on highly liquid stocks (AAPL), Mamba-based models achieve comparable accuracy with significantly better computational characteristics. This efficiency advantage becomes particularly valuable for: (1) longer historical windows, (2) high-frequency applications, and (3) portfolio-level forecasting where multiple assets must be analyzed simultaneously. The selective state space mechanism allows Mamba models to dynamically focus on relevant temporal patterns while ignoring noise, a critical capability for financial data.

5.4. Model-Horizon Alignment and Decomposition Effects

Our results demonstrate a clear triple alignment between model architecture, forecasting horizon, and stock characteristics: - CrossMamba's causal processing excels at short-term (5-day) forecasting - Hybrid Bi-Mamba's bidirectional context dominates longer (10-day) horizons - AutoFormer's decomposition works well for volatile stocks in short-term but degrades significantly with longer windows - Mamba hybrids provide the most consistent performance across different stock types and horizons.

This alignment suggests that optimal model selection should consider both the forecasting horizon and the volatility characteristics of the target stock, rather than seeking a universal solution.

5.5. Research Questions and Comparison to Prior Work

Our study directly addresses the two research questions posed in the introduction:

Question 1: Which Mamba variant is most effective? We find that optimal selection depends on horizon—CrossMamba for 5-day and Hybrid Bi-Mamba for 10-day windows. Our extended benchmarking shows these architectures compete effectively with state-of-the-art Transformers while offering superior efficiency. **Question 2:** Does decomposition improve performance? Series decomposition benefits stable, periodic stocks but can harm performance on volatile equities. The AMZN results demonstrate that even when decomposition works short-term (via AutoFormer), its benefits diminish compared to Mamba's consistent performance across horizons. Compared to prior work, our findings both confirm and extend existing knowledge. The superiority of Mamba-based architectures aligns with results from [18,24], while our systematic benchmarking across multiple horizons, architectures, and particularly a high-volatility stock (AMZN) provides novel insights into the optimal deployment conditions for different variants. Our hybrid designs demonstrate performance competitive with specialized financial forecasting models like Autoformer [14] while maintaining greater computational efficiency and robustness over longer horizons.

5.6. Advancements Beyond Current State of the Art

This work moves the field forward in several important ways. First, we introduce CrossMamba a novel architecture that efficiently models cross-sequence interactions through causal Mamba processing rather than attention mechanisms. Second, our comprehensive evaluation across multiple stocks (including the challenging AMZN case) and horizons provides practical guidance for model selection based on specific forecasting requirements. Third, our ablation studies offer mechanistic insights into why these architectures work, particularly regarding dependency modeling and feature importance. Finally, we provide a critical analysis of series decomposition, moving beyond its blanket application to a more nuanced, scenario-specific recommendation.

The demonstrated ability of Mamba-based models to achieve attention-like performance with linear complexity, and their proven robustness on volatile stocks over longer horizons, represents a significant step toward scalable and reliable financial forecasting systems.

5.7. Practical Implications for Investors

The proposed Mamba-augmented Transformer models provide more accurate and computationally efficient short- and medium-term stock price forecasts, which can support data-driven investment decisions. Improved short-horizon performance (e.g., 5-day forecasts) is particularly relevant for active trading and portfolio rebalancing, while more stable longer-horizon predictions can assist investors in identifying emerging trends and managing risk. The forecasts are intended to complement existing investment strategies by providing enhanced predictive signals rather than serving as standalone trading systems.

6. Limitations

While our study demonstrates the effectiveness of Mamba architectures, several limitations warrant attention. First, we focused exclusively on price and volume features; incorporating alternative data sources (news, sentiment, macroeconomic indicators) might further enhance performance. Second, our evaluation considered individual stock forecasting; extending this to portfolio optimization and market regime detection represents promising future work.

Despite these promising results, several limitations should be acknowledged. First, the study focuses on a limited set of individual equities and relies exclusively on price- and volume-based features, which may restrict generalization across broader asset classes and market conditions. Second, the evaluation emphasizes forecast accuracy rather than trading

performance, and does not account for transaction costs, execution delays, or portfolio-level constraints. Third, although chronological train–validation–test splits were used to prevent look-ahead bias, rolling or walk-forward validation strategies were not explored, which may better capture regime shifts in financial markets.

7. Conclusions and Future Work

Accurate stock price forecasting remains challenging due to the non-linear, noisy, and non-stationary nature of financial time series.

In this study, we systematically evaluated Mamba-augmented Transformer architectures, introduced *CrossMamba* for efficient cross-sequence modeling, and benchmarked these approaches against state-of-the-art forecasting methods. Our findings show that CrossMamba achieves superior performance for short-term forecasting (5-day windows with R^2 up to 0.963), while Hybrid Bi-Mamba provides the most stable and accurate results for longer horizons (10-day windows with MAE as low as \$0.67). We further observe that series decomposition improves performance for relatively stable stocks, but can degrade accuracy on highly volatile equities.

These results suggest that Mamba-based architectures provide a favorable balance between predictive accuracy and computational efficiency, making them promising candidates for large-scale financial forecasting tasks. Their linear-time complexity enables more efficient processing of longer historical sequences and multi-asset datasets compared to attention-based models, while maintaining competitive accuracy across different forecasting horizons.

These limitations motivate several avenues for future work. Incorporating alternative data sources such as news sentiment, macroeconomic indicators, or cross-asset signals may improve robustness, particularly during periods of heightened volatility. Evaluating Mamba-based models within trading or portfolio optimization frameworks would allow assessment of their economic significance beyond predictive accuracy. Future studies should also investigate rolling validation schemes and adaptive architectures that dynamically adjust to changing market regimes, further enhancing generalization and practical applicability. Future research should also explore: (1) adaptive decomposition techniques that dynamically adjust to market volatility, (2) hybrid architectures that automatically switch between Mamba and attention components based on sequence characteristics, and (3) applications to different asset classes (cryptocurrencies, commodities) to further validate generalizability. While SHAP-based interpretability offers valuable insights into feature contributions, applying SHAP to deep state-space sequence models such as Mamba remains computationally challenging and is left as an important direction for future work.

Author Contributions: Conceptualization, M.S.S.S., A.T.A. and R.M.R.; methodology, M.S.S.S.; software, M.S.S.S.; validation, M.S.S.S., A.T.A., M.E.A.E. and A.R.; formal analysis, M.S.S.S.; investigation, M.S.S.S.; resources, R.M.R.; data curation, M.S.S.S.; writing—original draft preparation, M.S.S.S.; writing—review and editing, A.T.A., A.R. and R.M.R.; visualization, M.S.S.S.; supervision, R.M.R.; project administration, R.M.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are publicly available at Kaggle. The S&P 500 dataset can be accessed at https://www.kaggle.com/datasets/camnugent/sandp500 (accessed on 15 January 2026), and additional stock data were obtained from Yahoo Finance

https://www.kaggle.com/datasets/iveeaten3223times/massive-yahoo-finance-dataset (accessed on 15 January 2026).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Fattah, J.; Ezzine, L.; Aman, Z.; El Moussami, H.; Lachhab, A. Forecasting of demand using ARIMA model. *Int. J. Eng. Bus. Manag.* **2018**, *10*, 1–9. [CrossRef]
2. Alfaro, R.; Inzunza, A. Modeling S&P500 returns with GARCH models. *Latin Am. J. Cent. Bank.* **2023**, *4*, 100096.
3. Wang, M. Advanced stock market forecasting: A comparative analysis of ARIMA-GARCH, LSTM, and integrated wavelet-LSTM models. *SHS Web Conf.* **2024**, *196*, 02008. [CrossRef]
4. Tu, T. Bridging short- and long-term dependencies: A CNN-transformer hybrid for financial time series forecasting. *arXiv* **2025**, arXiv:2504.19309.
5. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
6. Xiao, R.; Feng, Y.; Yan, L.; Ma, Y. Predict stock prices with ARIMA and LSTM. *arXiv* **2022**, arXiv:2209.02407. [CrossRef]
7. Pascanu, R.; Mikolov, T.; Bengio, Y. On the difficulty of training recurrent neural networks. *ICML* **2013**, *28*, 1310–1318.
8. Le, P.; Zuidema, W. Quantifying the vanishing gradient and long-distance dependency problem in recursive neural networks and recursive LSTMs. *arXiv* **2016**, arXiv:1603.00423.
9. Philps, D.; d'Avila Garcez, A.; Weyde, T. Making good on LSTMs' unfulfilled promise. *arXiv* **2019**, arXiv:1911.04489. [CrossRef]
10. Wen, Q.; Zhou, T.; Zhang, C.; Chen, W.; Ma, Z.; Yan, J.; Sun, L. Transformers in time series: A survey. *arXiv* **2022**, arXiv:2202.07125.
11. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
12. Zaheer, M.; Guruganesh, G.; Dubey, A.; Ainslie, J.; Alberti, C.; Ontañón, S.; Pham, P.; Ravula, A.; Wang, Q.; Yang, L.; et al. Big Bird: Transformers for longer sequences. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 17283–17297.
13. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. *AAAI* **2021**, *35*, 11106–11115. [CrossRef]
14. Wu, H.; Xu, J.; Wang, J.; Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *NeurIPS* **2021**, *34*, 22419–22430.
15. Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; Long, M. iTransformer: Inverted transformers are effective for time series forecasting. *arXiv* **2023**, arXiv:2310.06625.
16. Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; Jin, R. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. *ICML* **2022**, *162*, 27268–27286.
17. Gu, A.; Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. In Proceedings of the First Conference on Language Modeling (COLM 2023), Princeton, NJ, USA, 24–27 October 2023.
18. Wang, Z.; Kong, F.; Feng, S.; Wang, M.; Yang, X.; Zhao, H.; Wang, D.; Zhang, Y. Is Mamba effective for time series forecasting? *Neurocomputing* **2025**, *619*, 129178. [CrossRef]
19. Zhu, L.; Liao, B.; Zhang, Q.; Wang, X.; Liu, W.; Wang, X. Vision Mamba: Efficient visual representation learning with bidirectional state space model. *arXiv* **2024**, arXiv:2401.09417. [CrossRef]
20. Ruan, J.; Li, J.; Xiang, S. VM-UNet: Vision Mamba UNet for Medical Image Segmentation. *arXiv* **2024**, arXiv:2402.02491. Available online: https://arxiv.org/abs/2402.02491 (accessed on 15 January 2026).
21. Ma, H.; Chen, Y.; Zhao, W.; Yang, J.; Ji, Y.; Xu, X.; Liu, X.; Jing, H.; Liu, S.; Yang, G. A Mamba foundation model for time series forecasting. *arXiv* **2024**, arXiv:2411.02941. [CrossRef]
22. Liang, A.; Jiang, X.; Sun, Y.; Lu, C. Bi-Mamba+: Bidirectional Mamba for time series forecasting. *arXiv* **2024**, arXiv:2404.15772.
23. Zeng, C.; Liu, Z.; Zheng, G.; Kong, L. C-Mamba: Channel correlation enhanced state space models for multivariate time series forecasting. *arXiv* **2024**, arXiv:2406.05316. [CrossRef]
24. Shi, Z. MambaStock: Selective state space model for stock prediction. *arXiv* **2024**, arXiv:2402.18959. [CrossRef]
25. Mehrabian, A.; Hoseinzade, E.; Mazloum, M.; Chen, X. Mamba meets financial markets: A graph-Mamba approach for stock price prediction. In Proceedings of the ICASSP 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Hyderabad, India, 6–11 April 2025.
26. Yan, W.; Wang, S.; Tan, Y. Hierarchical information-guided spatio-temporal Mamba for stock time series forecasting. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*; Springer: Berlin, Germany, 2025.
27. Zhang, W.; Huang, J.; Wang, R.; Wei, C.; Huang, W.; Qiao, Y. Integration of Mamba and Transformer—MAT for long-short range time series forecasting with application to weather dynamics. In Proceedings of the 2024 International Conference on Electrical, Communication and Computer Engineering (ICECCE), Kuala Lumpur, Malaysia, 30–31 October 2024.

28. Keles, F.D.; Wijewardena, P.M.; Hegde, C. On the computational complexity of self-attention. In Proceedings of the International Conference on Algorithmic Learning Theory, Singapore, 20–23 February 2023; Volume 201, pp. 597–619.
29. Fournier, Q.; Marceau Caron, G.; Aloise, D. A practical survey on faster and lighter transformers. *ACM Comput. Surv.* **2023**, *55*, 1–40. [CrossRef]
30. Gu, A.; Goel, K.; Ré, C. Efficiently modeling long sequences with structured state spaces. *arXiv* **2021**, arXiv:2111.00396.
31. Xu, X.; Chen, C.; Liang, Y.; Huang, B.; Bai, G.; Zhao, L.; Shu, K. SST: Multi-scale hybrid Mamba-transformer experts for long-short range time series forecasting. *arXiv* **2024**, arXiv:2404.14757.
32. Ye, H.; Duan, G.; Zeng, H.; Zhu, Y.; Meng, L.; Zheng, X.; Zhu, Y. KARMA: A multilevel decomposition hybrid Mamba framework for multivariate long-term time series forecasting. In *Proceedings of the International Conference on Wireless Artificial Intelligent Computing Systems and Applications*; Springer: Singapore, 2025.
33. Yang, Y.; Huang, S.; Lu, X.; Jiao, L.; Liu, X.; Chen, P. PDMamba: Hybrid Mamba Attention Network for Stock Data Prediction. 2025. Available online: https://ssrn.com/abstract=5244598 (accessed on 15 January 2026).
34. Yang, Y.; Huang, S.; Lu, X.; Jiao, L.; Liu, X.; Chen, P. T-Mamba: A hybrid Mamba-transformer framework for stock price prediction. In Proceedings of the 2025 6th International Conference on Computer Information and Big Data Applications, Wuhan, China, 14–16 March 2025.
35. Gu, S.; Kelly, B.; Xiu, D. Empirical Asset Pricing via Machine Learning. *Rev. Financ. Stud.* **2020**, *33*, 2223–2273. [CrossRef]
36. Kelly, B.T.; Xiu, D. Financial Machine Learning. *NBER Work. Paper* **2023**, *13*, 205–363. [CrossRef]
37. Raza, H.; Akhtar, Z. Predicting stock prices in the Pakistan market using machine learning and technical indicators. *Mod. Financ.* **2024**, *2*, 46–63. [CrossRef]
38. Wu, H.; Wu, J.; Xu, J.; Wang, J.; Long, M. Flowformer: Linearizing transformers with conservation flows. *ICML* **2022**, *162*, 24226–24242.
39. Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; Wang, J.; Long, M. TimesNet: Temporal 2D-variation modeling for general time series analysis. *arXiv* **2022**, arXiv:2210.02186.
40. Nie, Y.; Nguyen, N.H.; Sinthong, P.; Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv* **2022**, arXiv:2211.14730.
41. Nrusimha, A.; Brandon, W.; Mishra, M.; Shen, Y.; Panda, R.; Ragan-Kelley, J.; Kim, Y. FlashFormer: Whole-model kernels for efficient low-batch inference. *arXiv* **2025**, arXiv:2505.22758.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.