CAUCHY – Jurnal Matematika Murni dan Aplikasi
Volume 10 (2) (2025), Pages 1401-1416
p-ISSN: 2086-0382; e-ISSN: 2477-3344

On the Approximation Capabilities of Deep Neural Networks for Multivariate Time Series: A Case Study on AAPL Stock Prices

Mohammad Jamhuri[1], Mohammad Isa Irawan[2*], Ari Kusumastuti[1], Kartick Chandra Mondal[3], and Juhari[1]

[1]*Department of Mathematics, Faculty of Science and Technology, UIN Maulana Malik Ibrahim, Malang, Indonesia*
[2]*Department of Mathematics, Faculty of Science and Data Analytics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia*
[3]*Jadavpur University, Salt Lake Campus, India*

Abstract

This work studies how well different deep learning architectures approximate the mapping from past multivariate observations to next-day stock prices. We compare a Multilayer Perceptron (MLP), a one-dimensional Convolutional Neural Network (1D-CNN), and a Long Short-Term Memory (LSTM) network against a Vector Autoregression (VAR) baseline using daily Apple Inc. stock prices (low, high, and close). The forecasting task is framed as learning from sliding windows of past prices with lengths of 5, 10, 30, 180, and 360 days, representing short, medium, and long horizons. All models are trained on the same normalized data and evaluated using multivariate mean squared error, root mean squared error, mean absolute error, and mean absolute percentage error, complemented by residual and box-plot analyses. The results show that MLP provides the most accurate and stable approximation for short and medium windows, while LSTM clearly dominates for long windows. CNN occupies an intermediate position, and VAR consistently underperforms. These findings highlight that approximation capability is strongly horizon-dependent and offer practical guidance for model selection in multivariate financial time series.

Keywords: Deep neural networks; Approximation capability; Multivariate time series; Stock price forecasting; AAPL stock prices; Vector autoregression

1 Introduction

Time series forecasting plays a central role in finance, energy, healthcare, and meteorology, where accurate predictions support decision-making, risk management, and planning [1], [2], [3]. In financial markets, the ability to forecast stock prices based on historical data is particularly important for portfolio allocation and trading strategies [4], [5]. In this paper, we consider the daily stock prices of Apple Inc. (AAPL) as a multivariate time series, focusing on the joint forecasting of three variables: the lowest price (low), the highest price (high), and the closing price (close). The data are obtained from Yahoo Finance and cover the period from January 2020 to November 2024.

*Corresponding author. E-mail: mii@its.ac.id

Submitted: April 28, 2025 Reviewed: June 21, 2025 Accepted: November 30, 2025
DOI: https://doi.org/10.18860/cauchy.v10i2.32760

Classical time series models such as Autoregressive Integrated Moving Average (ARIMA), Exponential Smoothing (ETS), and Vector Autoregression (VAR) have long been used for forecasting [6]. However, these models are fundamentally linear and can struggle with nonlinear dynamics, complex cross-variable interactions, and long-term temporal dependencies [7]. This is particularly relevant for multivariate financial series, where nonlinear effects and interactions between variables are common. In such cases, models with richer approximation capabilities are needed to capture the underlying dynamics more accurately.

Deep learning models, including Multilayer Perceptrons (MLP), Convolutional Neural Networks (CNN), and Recurrent Neural Networks such as Long Short-Term Memory (LSTM), provide flexible nonlinear function approximators [8], [9], [10]. In the time series setting, these architectures can be viewed as learning an unknown mapping from a window of past multivariate observations to the next multivariate value. MLPs approximate this mapping through fully connected layers, CNNs capture local patterns via convolutional filters [11], and LSTMs introduce memory cells designed to capture long-term temporal dependencies [12], [13]. From an approximation-theoretic perspective, such architectures can be related to universal approximation results for neural networks [14], [15].

Formally, forecasting a multivariate time series can be formulated as learning a nonlinear operator

$$f^* : \mathbb{R}^{p \times d} \to \mathbb{R}^d,$$

where d is the number of variables (here $d = 3$ for low, high, and close prices) and p is the window length. Deep neural networks provide parametric families f_θ that attempt to approximate f^* on the data domain. The *approximation capability* of a given architecture is then reflected in the quality of this approximation under constraints on depth, width, and available data. In this sense, time series forecasting performance—measured by MSE, RMSE, MAE, and MAPE on a held-out test set—serves as an empirical proxy for approximation capability.

A large body of work has applied deep learning to financial and stock-price forecasting, typically emphasizing predictive accuracy for a particular architecture or dataset [5], [13], [16], [17]. Comprehensive surveys on deep learning for time series and forecasting further document the rapid growth of this literature [9], [10]. However, many studies treat the models as black boxes and do not explicitly frame the problem in terms of approximation capabilities across architectures and horizons. Moreover, most works focus on short or fixed window lengths, providing limited insight into how well different architectures scale from short to long multivariate histories under comparable capacity constraints.

In this paper, we adopt an explicit approximation viewpoint and address the following questions: (i) how do MLP, 1D-CNN, and LSTM differ in their ability to approximate the multivariate mapping from past AAPL prices (low, high, close) to the next-day prices; (ii) how does this comparison depend on the input window length, ranging from short (5 days) and medium (10, 30 days) to long (180, 360 days); and (iii) how do these deep models compare with a conventional VAR baseline that is restricted to linear dependencies. We combine quantitative metrics with residual and error-distribution analyses to illuminate not only which model performs best, but also how approximation errors are structured.

The main contributions of this work are as follows:

- We formulate multivariate stock-price forecasting explicitly as a nonlinear function approximation problem and interpret deep neural networks and VAR as competing function approximators with different capacities.
- We perform a systematic empirical study comparing MLP, 1D-CNN, LSTM, and VAR on a multivariate AAPL stock dataset, across window lengths $p \in \{5, 10, 30, 180, 360\}$, thereby probing short-, medium-, and long-horizon approximation regimes.
- We provide detailed methodological specifications for all architectures (including layer configurations, hyperparameters, and VAR lag order) to ensure reproducibility, and we clarify how multivariate evaluation metrics are aggregated across variables.

- We analyze prediction errors using residual plots and box plots, highlighting how approximation behaviour differs across architectures and horizons and offering practical guidelines on when to prefer each model.

The remainder of the paper is organized as follows. Section 2 presents the data, problem formulation, preprocessing, model architectures, and training and evaluation procedures. Section 3 reports the experimental results for different window lengths and discusses the approximation behaviour of each model, including residual analysis and error distributions. Section 4 concludes with a summary of findings and directions for future work.

2 Methods

This section describes the dataset, the formal problem formulation as multivariate function approximation, the preprocessing steps, the deep learning and VAR models, and the training and evaluation procedures.

2.1 Dataset and Problem Formulation

We consider a multivariate time series constructed from daily AAPL stock prices. For each trading day $t = 1, 2, \ldots, T$, we observe three variables:

$$\mathbf{X}(t) = [X_1(t), X_2(t), X_3(t)],$$

where $X_1(t)$ is the lowest price (low), $X_2(t)$ is the highest price (high), and $X_3(t)$ is the closing price (close). The data span from January 2020 to November 2024 and are obtained from Yahoo Finance via the `yfinance` API.

Let $d = 3$ denote the number of variables. For a given window length p, we define the input at time t as the stacked window

$$\mathbf{X}_{t-p+1:t} = [\mathbf{X}(t-p+1),\, \mathbf{X}(t-p+2),\, \ldots,\, \mathbf{X}(t)] \in \mathbb{R}^{p\times d},$$

and the corresponding target as the next-day vector

$$\mathbf{Y}_t = \mathbf{X}(t+1) \in \mathbb{R}^d.$$

The forecasting problem is thus to learn a parametric function $f_\theta : \mathbb{R}^{p\times d} \to \mathbb{R}^d$ that approximates the unknown mapping f^* defined by the data-generating process, such that

$$\hat{\mathbf{Y}}_t = f_\theta(\mathbf{X}_{t-p+1:t})$$

is close to $\mathbf{Y}_t$ on a held-out test set. The approximation capability of a model family is reflected in the generalization error $\ell(\hat{\mathbf{Y}}_t, \mathbf{Y}_t)$ achieved on this test set.

In our experiments, we investigate five window lengths:

$$p \in \{5, 10, 30, 180, 360\},$$

corresponding to short (5 days), short-to-medium (10, 30 days), and long (180, 360 days) horizons.

2.2 Data Partitioning and Preprocessing

This subsection describes how the raw AAPL price series is transformed into supervised learning data suitable for training the deep models and the VAR baseline.

2.2.1 Train–validation–test split

To ensure a fair and reproducible evaluation, we partition the time series chronologically into three disjoint subsets:

- 70% of the data for training,
- 15% for validation (hyperparameter tuning and early stopping),
- 15% for testing (final evaluation).

The test set is never used during model selection or training. The same chronological split is applied consistently to all models, including the VAR baseline.

2.2.2 Handling missing values

Before scaling and windowing, occasional missing values in the raw series are handled by linear interpolation to preserve temporal continuity. For a missing value $X_i(t)$ between two observed values $X_i(t-1)$ and $X_i(t+1)$, we use the midpoint

$$X_i(t) = \frac{X_i(t-1) + X_i(t+1)}{2},$$

which is the standard linear interpolation formula. This operation is applied independently to each variable X_i.

2.2.3 Normalization

Deep neural networks are sensitive to feature scaling. Each variable is therefore normalized to the range $[0, 1]$ using min–max scaling computed on the *training* set:

$$X_i'(t) = \frac{X_i(t) - \min(X_i)}{\max(X_i) - \min(X_i)},$$

where $\min(X_i)$ and $\max(X_i)$ denote the minimum and maximum values of X_i over the training period. The same scaling parameters are applied to the validation and test sets. All models, including VAR, are trained and evaluated on the normalized data; for interpretability, plots can be presented either in normalized units or after inverse transformation to the original scale.

2.2.4 Sliding-window construction

After normalization, we transform the series into supervised learning examples using a sliding-window operator. For a given window length p, each input–target pair is constructed as

$$\mathbf{X}_{t-p+1:t} = \begin{bmatrix} \mathbf{X}(t-p+1) \\ \mathbf{X}(t-p+2) \\ \vdots \\ \mathbf{X}(t) \end{bmatrix} \in \mathbb{R}^{p \times d}, \qquad \mathbf{Y}_t = \mathbf{X}(t+1) \in \mathbb{R}^d,$$

for $t = p, p+1, \ldots, T-1$. The sliding-window construction is applied separately on the training, validation, and test segments to avoid information leakage from future observations.

The resulting supervised dataset can be viewed as a matrix of inputs $\mathbf{X}$ and a matrix of targets $\mathbf{Y}$. For illustration, when $p = 2$ the input and target matrices can be written as

$$\mathbf{X} = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{21} & y_{22} & y_{23} \\ y_{21} & y_{22} & y_{23} & y_{31} & y_{32} & y_{33} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{(m-2)1} & y_{(m-2)2} & y_{(m-2)3} & y_{(m-1)1} & y_{(m-1)2} & y_{(m-1)3} \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} y_{31} & y_{32} & y_{33} \\ y_{41} & y_{42} & y_{43} \\ \vdots & \vdots & \vdots \\ y_{m1} & y_{m2} & y_{m3} \end{bmatrix},$$

where each row corresponds to an input window (two consecutive days of low, high, close) and its next-day target. A similar representation holds for $p = 3$. In the experiments, however, we use the window lengths $p \in \{5, 10, 30, 180, 360\}$; the generalisation from these illustrative cases is straightforward.

2.3 Deep Learning Architectures

We implement three deep learning architectures to approximate the mapping from past AAPL prices (low, high, close) to the next-day prices: a Multilayer Perceptron (MLP), a one-dimensional Convolutional Neural Network (1D-CNN), and a Long Short-Term Memory (LSTM) network. All models are implemented using the Keras API of TensorFlow and operate on the same windowed inputs.

2.3.1 Multilayer Perceptron (MLP)

For a window length p and $d = 3$ variables, the input $\mathbf{X}_{t-p+1:t} \in \mathbb{R}^{p\times d}$ is first flattened into a vector $\mathbf{z}_t \in \mathbb{R}^{pd}$. The overall architecture of the Multilayer Perceptron used in this study is summarized in Figure 1: the flattened input $\mathbf{z}_t$ is propagated through three fully connected hidden layers with decreasing widths (64, 32, and 16 units), followed by a 3-dimensional linear output layer producing the forecast $\hat{\mathbf{Y}}_t$.

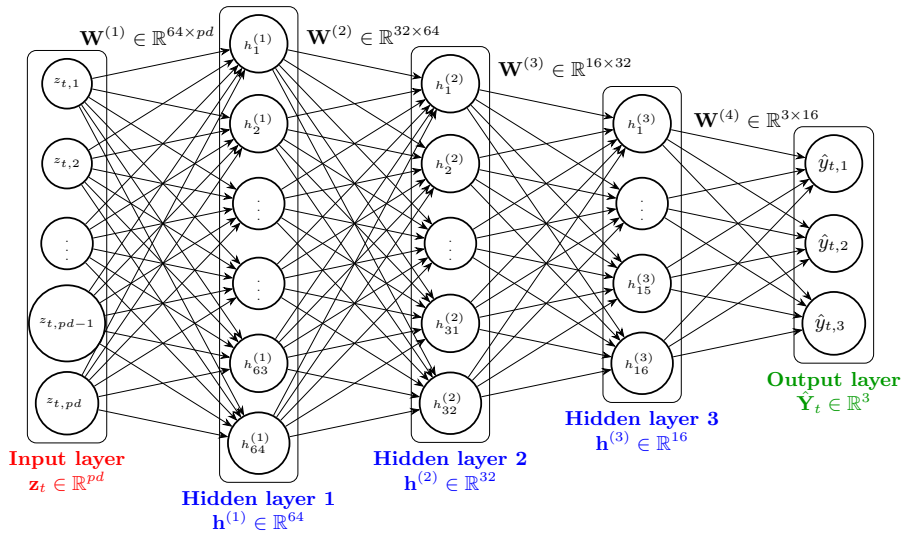

Figure 1: Schematic illustration of the MLP architecture used in this study. The input vector $\mathbf{z}_t \in \mathbb{R}^{pd}$ (obtained by flattening the $p \times 3$ window of past prices) is mapped through three hidden layers with 64, 32, and 16 ReLU units, respectively, to a three-dimensional output $\hat{\mathbf{Y}}_t \in \mathbb{R}^3$ containing the predicted low, high, and close prices.

Formally, the transformations implemented by the MLP can be written as

$$\begin{aligned}
\mathbf{h}^{(1)} &= \sigma\left(\mathbf{W}^{(1)}\mathbf{z}_t + \mathbf{b}^{(1)}\right), \quad \mathbf{W}^{(1)} \in \mathbb{R}^{64\times pd},\\
\mathbf{h}^{(2)} &= \sigma\left(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)}\right), \quad \mathbf{W}^{(2)} \in \mathbb{R}^{32\times 64},\\
\mathbf{h}^{(3)} &= \sigma\left(\mathbf{W}^{(3)}\mathbf{h}^{(2)} + \mathbf{b}^{(3)}\right), \quad \mathbf{W}^{(3)} \in \mathbb{R}^{16\times 32},\\
\hat{\mathbf{Y}}_t &= \mathbf{W}^{(4)}\mathbf{h}^{(3)} + \mathbf{b}^{(4)}, \quad \mathbf{W}^{(4)} \in \mathbb{R}^{3\times 16},
\end{aligned}$$

where σ denotes the Rectified Linear Unit (ReLU) activation and $\hat{\mathbf{Y}}_t = (\hat{y}_{t,1}, \hat{y}_{t,2}, \hat{y}_{t,3})$ collects the predicted low, high, and close prices for day $t + 1$. To reduce overfitting, we apply a dropout

rate of **0.2** after the first and second hidden layers (i.e., between $\mathbf{h}^{(1)}$ and $\mathbf{h}^{(2)}$, and between $\mathbf{h}^{(2)}$ and $\mathbf{h}^{(3)}$).[1]

All weights and biases are learned via backpropagation using the Adam optimizer.

2.3.2 One-Dimensional Convolutional Neural Network (1D-CNN)

The 1D-CNN treats each input window as a sequence of length p with $d = 3$ channels (low, high, close). Thus, the input tensor has shape $(p, 3)$.

We use the following architecture:

- a first convolutional layer with **64** filters, kernel size **3**, stride 1, "same" padding, and ReLU activation;
- a max-pooling layer with pool size **2**;
- a second convolutional layer with **32** filters, kernel size **3**, stride 1, "same" padding, and ReLU activation;
- a global average pooling layer over the time dimension;
- a fully connected layer with **32** units and ReLU activation;
- a linear output layer with 3 units.

Let $\mathbf{X}_{t-p+1:t} \in \mathbb{R}^{p\times 3}$ denote the input tensor. The output of the first convolutional layer for filter k is

$$\mathbf{h}_k^{(1)} = \sigma(\mathbf{X}_{t-p+1:t} * \mathbf{W}^{(1,k)} + b_k^{(1)}),$$

where $*$ denotes convolution along the time axis, $\mathbf{W}^{(1,k)}$ is a kernel of size 3×3, and $b_k^{(1)}$ is a scalar bias. After the second convolutional layer and pooling operations, the resulting feature maps are aggregated by global average pooling and fed into the fully connected layers to produce $\hat{\mathbf{Y}}_t$.

2.3.3 Long Short-Term Memory (LSTM)

The LSTM model processes each window as a sequence of p time steps, where the input at time τ is $\mathbf{X}(\tau) \in \mathbb{R}^3$. In our implementation, we use two stacked LSTM layers:

- a first LSTM layer with hidden size **64** and `return_sequences = True`;
- a second LSTM layer with hidden size **32** and `return_sequences = False`;
- a final linear dense layer with 3 units.

Let $\mathbf{h}_\tau^{(1)}$ and $\mathbf{c}_\tau^{(1)}$ denote the hidden and cell states of the first LSTM layer at time τ, and $\mathbf{h}_\tau^{(2)}$, $\mathbf{c}_\tau^{(2)}$ those of the second layer. The update equations within each layer follow the standard LSTM formulation with input, forget, and output gates:

$$\begin{aligned}
i_\tau &= \sigma(\mathbf{W}_i[\mathbf{h}_{\tau-1}; \mathbf{X}(\tau)] + \mathbf{b}_i),\\
f_\tau &= \sigma(\mathbf{W}_f[\mathbf{h}_{\tau-1}; \mathbf{X}(\tau)] + \mathbf{b}_f),\\
o_\tau &= \sigma(\mathbf{W}_o[\mathbf{h}_{\tau-1}; \mathbf{X}(\tau)] + \mathbf{b}_o),\\
\tilde{\mathbf{c}}_\tau &= \tanh(\mathbf{W}_c[\mathbf{h}_{\tau-1}; \mathbf{X}(\tau)] + \mathbf{b}_c),\\
\mathbf{c}_\tau &= f_\tau \odot \mathbf{c}_{\tau-1} + i_\tau \odot \tilde{\mathbf{c}}_\tau,\\
\mathbf{h}_\tau &= o_\tau \odot \tanh(\mathbf{c}_\tau),
\end{aligned}$$

where σ denotes the sigmoid activation, $\odot$ denotes element-wise multiplication, and $\mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_o, \mathbf{W}_c$ and $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o, \mathbf{b}_c$ are the trainable parameters of the LSTM layer.

[1] In Keras notation: `Dense(64, relu)` – `Dense(32, relu)` – `Dense(16, relu)` – `Dense(3, linear)`, with dropout layers of rate 0.2 inserted after the first and second hidden layers.

The second LSTM layer takes the full sequence $\{\mathbf{h}_\tau^{(1)}\}_{\tau=t-p+1}^{t}$ as input, and its last hidden state $\mathbf{h}_t^{(2)}$ is mapped to the output via

$$\hat{\mathbf{Y}}_t = \mathbf{W}_{\text{out}}\mathbf{h}_t^{(2)} + \mathbf{b}_{\text{out}},$$

with $\mathbf{W}_{\text{out}} \in \mathbb{R}^{3\times 32}$ and $\mathbf{b}_{\text{out}} \in \mathbb{R}^3$. A dropout rate of **0.2** is applied to the recurrent outputs between the two LSTM layers.

All LSTM parameters are learned via backpropagation through time using the Adam optimizer.

2.4 VAR Baseline

As a conventional linear baseline, we fit a Vector Autoregression (VAR) model to the same normalized multivariate series $\mathbf{X}(t) = [X_1(t), X_2(t), X_3(t)]$. The VAR($p_{\text{VAR}}$) model is specified as

$$\mathbf{X}(t) = \mathbf{c} + \sum_{k=1}^{p_{\text{VAR}}} \mathbf{A}_k \mathbf{X}(t-k) + \boldsymbol{\varepsilon}(t),$$

where $\mathbf{c} \in \mathbb{R}^3$ is a constant vector, $\mathbf{A}_k \in \mathbb{R}^{3\times 3}$ are coefficient matrices, and $\boldsymbol{\varepsilon}(t)$ is a zero-mean white noise process with covariance matrix $\boldsymbol{\Sigma}$.

We estimate the VAR model using ordinary least squares (OLS) as implemented in the `statsmodels VAR` class. The lag order p_{VAR} is selected on the training set by minimizing the Akaike Information Criterion (AIC) over candidate orders $p_{\text{VAR}} \in \{1, \ldots, 10\}$. In our experiments, this procedure selects $p_{\text{VAR}} = \mathbf{2}$, and we report results for this lag order throughout the paper.

Once the VAR(2) model is fitted, we generate one-step-ahead forecasts on the validation and test sets by rolling the model forward through time. Because VAR operates on its own autoregressive lag structure rather than on externally specified window lengths p, its performance is independent of the deep-model window choices. For convenience of comparison, we therefore report the same VAR metrics in the tables for all window settings.

2.5 Training Setup and Evaluation Metrics

All deep models are trained using the Adam optimizer with learning rate $\alpha = 0.001$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. We use a batch size of 32 and a maximum of 200 epochs. Early stopping based on validation MAPE with a patience of 20 epochs is employed to prevent overfitting. For each architecture and window length, we train the model once with a fixed random seed; extensions with multiple runs and confidence intervals are left for future work.

Let $\hat{\mathbf{Y}}_i = (\hat{y}_{i1}, \hat{y}_{i2}, \hat{y}_{i3})$ and $\mathbf{Y}_i = (y_{i1}, y_{i2}, y_{i3})$ denote the predicted and true vectors for the i-th test example, respectively, with $i = 1, \ldots, N$. To quantify approximation error in the multivariate setting, we treat each component (i, j), $j = 1, 2, 3$, as a single observation and aggregate errors across both time and variables.

The Mean Squared Error (MSE) is defined as

$$\text{MSE} = \frac{1}{3N}\sum_{i=1}^{N}\sum_{j=1}^{3}\big(\hat{y}_{ij} - y_{ij}\big)^2,$$

and the Mean Absolute Error (MAE) as

$$\text{MAE} = \frac{1}{3N}\sum_{i=1}^{N}\sum_{j=1}^{3}\big|\hat{y}_{ij} - y_{ij}\big|.$$

The Root Mean Squared Error (RMSE) is

$$\text{RMSE} = \sqrt{\text{MSE}}.$$

The Mean Absolute Percentage Error (MAPE), expressed as a percentage, is computed as

$$\text{MAPE} = \frac{100\%}{|\mathcal{I}|} \sum_{(i,j)\in\mathcal{I}} \left| \frac{\hat{y}_{ij} - y_{ij}}{y_{ij}} \right|,$$

where $\mathcal{I}$ is the set of index pairs (i, j) such that $y_{ij} \neq 0$ to avoid division by zero. All metrics reported in Tables 1 and 2 are computed according to these multivariate definitions.

These four metrics together provide a comprehensive view of approximation capability: MSE and RMSE emphasize large errors, MAE provides a robust measure in the original scale, and MAPE expresses the average relative error in percentage terms.

3 Results and Discussion

This section reports the empirical approximation performance of the deep models (MLP, 1D-CNN, LSTM) and the VAR baseline on the multivariate AAPL stock data. Recall that, for each window length p, the models approximate the mapping $f^* : \mathbb{R}^{p\times 3} \to \mathbb{R}^3$ from the past p days of (low, high, close) to the next-day vector. The test-set errors (MSE, MAE, RMSE, MAPE) therefore provide an empirical measure of how well each architecture approximates f^* under the given horizon and model capacity.

We first examine short and medium windows ($p = 5, 10, 30$), then long windows ($p = 180, 360$), and finally analyse residuals and error distributions for the representative case $p = 10$.

3.1 Short and Medium Windows ($p = 5, 10, 30$)

Table 1 reports the multivariate MSE, MAE, RMSE, and MAPE for all models at window lengths $p \in \{5, 10, 30\}$. Each metric aggregates errors across the three variables (low, high, close) and all test examples, as defined in Section 2.6.

Table 1: MSE, MAE, RMSE, and MAPE for each model by time window length (short and medium horizons). Metrics are aggregated across low, high, and close prices.

Window p	Model	MSE	MAE	RMSE	MAPE (%)
5	MLP	**20.28**	**3.66**	**4.50**	**1.76**
	CNN	34.87	4.69	5.90	2.22
	LSTM	41.48	4.84	6.44	2.25
	VAR	507.76	20.01	22.53	9.47
10	MLP	**0.80**	**0.67**	**0.90**	**1.22**
	CNN	2.07	1.17	1.44	2.05
	LSTM	25.71	3.77	5.07	6.27
	VAR	72.71	6.14	8.53	10.03
30	MLP	**1.90**	**1.19**	**1.38**	**2.10**
	CNN	5.11	1.81	2.26	3.02
	LSTM	3.54	1.34	1.88	2.28
	VAR	84.73	6.77	9.20	10.94

For all three short/medium windows, the MLP attains the lowest empirical approximation error on every metric. In particular, the 10-day MLP configuration yields an MSE of 0.80, MAE of 0.67, RMSE of 0.90, and MAPE of 1.22%, indicating that the learned mapping f_θ^{MLP} provides a very accurate approximation of f^* when restricted to the recent 10-day history.

The 1D-CNN consistently appears as the second-best deep model in this regime. Its convolutional filters capture local temporal patterns and reduce the approximation error compared to the linear VAR baseline, but the fully connected MLP achieves lower test error, suggesting that for windows up to 30 days the complexity of f^* can already be captured by a relatively shallow feedforward network acting on the flattened input.

LSTM exhibits higher errors than MLP and CNN at these window lengths. Although its recurrent structure is, in principle, capable of approximating complex sequence-to-sequence mappings, the empirical results indicate that this additional machinery does not translate into superior approximation quality when only 5–30 past days are available. In this short-to-medium regime, the mapping from input window to next-day prices seems to be sufficiently low-dimensional and local that simpler architectures approximate it more effectively.

Across all $p \in \{5, 10, 30\}$, the VAR baseline exhibits the largest errors by a wide margin. This behaviour is consistent with its restricted approximation capacity: as a linear model, VAR cannot represent the nonlinear interactions and regime changes present in the multivariate AAPL series, leading to large residuals even when only short windows are considered. In approximation terms, VAR provides only a coarse linear surrogate of f^*.

3.2 Long Windows ($p = 180, 360$)

We now move to long windows, where the input dimensionality and potential temporal dependencies are substantially larger. Table 2 reports the multivariate errors for $p = 180$ and $p = 360$.

Table 2: Comparison of MSE, MAE, RMSE, and MAPE across models for long windows ($p = 180$ and $p = 360$). VAR results are independent of p and therefore identical across rows.

Window p	Model	MSE	MAE	RMSE	MAPE (%)
180	MLP	24.21	4.88	4.92	6.85
	CNN	78.87	8.86	8.88	12.44
	LSTM	**3.76**	**1.69**	**1.93**	**2.38**
	VAR	529.44	22.98	23.00	32.26
360	MLP	19.28	4.34	4.39	6.11
	CNN	35.72	5.93	5.97	8.33
	LSTM	**1.07**	**0.87**	**1.03**	**1.22**
	VAR	529.44	22.98	23.00	32.26

In contrast to the short-window regime, LSTM now clearly provides the best empirical approximation of f^*. For $p = 180$, the LSTM achieves an MSE of 3.76, MAE of 1.69, RMSE of 1.93, and MAPE of 2.38%, substantially improving on the feedforward MLP and 1D-CNN. At $p = 360$, the advantage becomes even more pronounced: the LSTM reaches an MSE of 1.07, MAE of 0.87, RMSE of 1.03, and MAPE of 1.22%, while the alternative deep models incur significantly larger errors.

This behaviour is consistent with the approximation-theoretic perspective. When p is large, the effective domain $\mathbb{R}^{p\times 3}$ is high-dimensional, and the mapping f^* may depend on long-range temporal patterns that smaller windows cannot express. The LSTM's recurrent state and gating mechanisms allow it to compress long histories into an internal representation that better matches the complexity of f^*, thereby reducing the empirical approximation error compared to static architectures that process all p inputs in a single feedforward pass.

The MLP remains the second-best model in the long-window regime but with noticeably higher errors than LSTM. Its fixed-size hidden layers must map a very high-dimensional input vector to the output, making it harder to approximate f^* without either increasing capacity or overfitting. The 1D-CNN performs worst among the deep models for $p = 180$, and although its errors decrease at $p = 360$, it still lags behind both LSTM and MLP. This suggests that local convolutional filters alone are insufficient to capture the long-range dependencies that become important when using year-long windows.

The VAR baseline again exhibits the largest errors by a considerable margin, with MAPE above 30% for both long-window settings. Since a single VAR model with lag order selected by AIC on the training set is used, its empirical approximation performance is invariant across the

p values considered here. In other words, VAR approximates f^* via a fixed-order linear recursion that cannot be improved simply by feeding longer external windows.

3.3 Approximation Error Structure for a 10-Day Window

To further characterise approximation behaviour in the short-to-medium regime, we analyse the 10-day window in more detail. In this setting, the input $\mathbf{X}_{t-9:t}$ contains the most recent 10 days of low, high, and close prices, and the models attempt to approximate the mapping to the next-day prices $\mathbf{X}(t+1)$.

We first inspect how well each model reproduces the observed price trajectories on the test set. Figure 2 plots the actual and predicted *low* prices for MLP, 1D-CNN, LSTM, and VAR over time, allowing a visual comparison of their one-step-ahead forecasts.

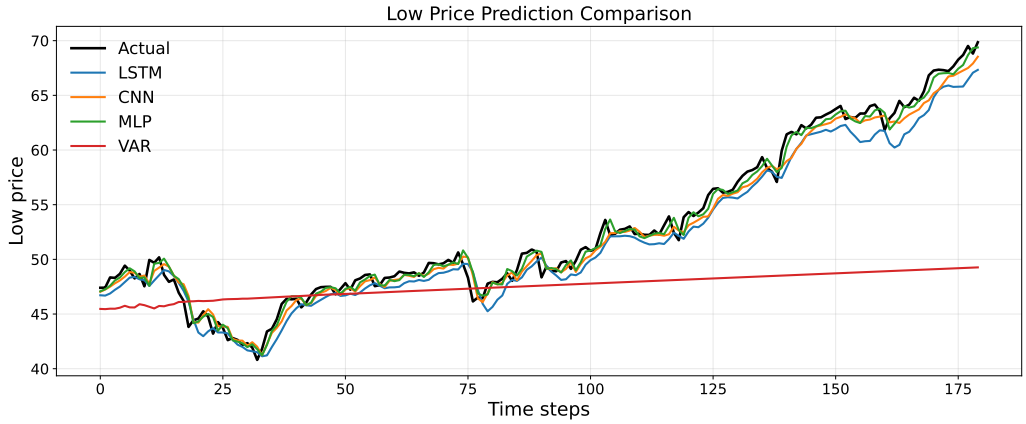

Figure 2: Forecast comparison for low prices on the test set with a 10-day input window.

Figure 3 presents the analogous comparison for the *high* prices, again overlaying the four models' predictions on the realised series.

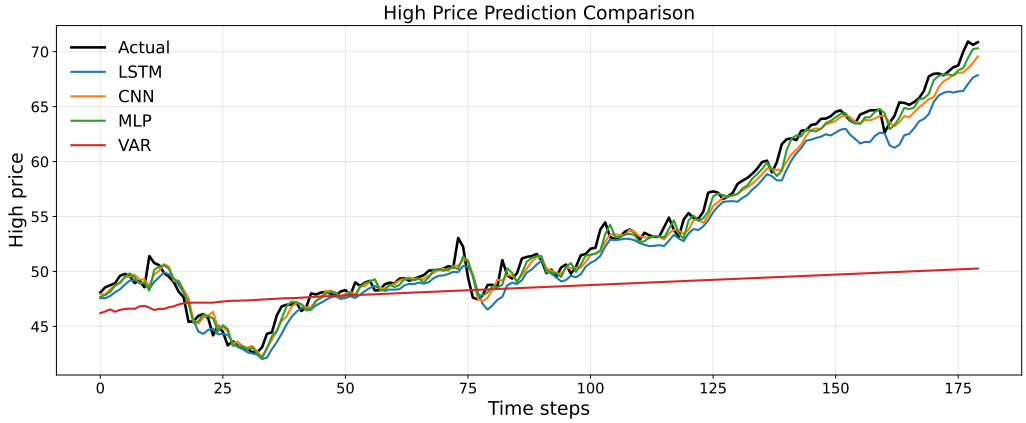

Figure 3: Forecast comparison for high prices on the test set with a 10-day input window.

Finally, Figure 4 shows the approximation behaviour for the *close* prices, completing the picture across the three components of the multivariate series.

Across all three variables, the MLP predictions closely follow the actual series, with only small deviations, in line with its low multivariate errors in Table 1. This visual agreement indicates that f_θ^{MLP} provides an accurate approximation to f^* when restricted to inputs of length 10. The 1D-CNN also tracks the true series reasonably well, but its prediction curves deviate more visibly from the observed prices, confirming its slightly higher empirical approximation error in this regime. LSTM and VAR exhibit larger and more frequent discrepancies, particularly around

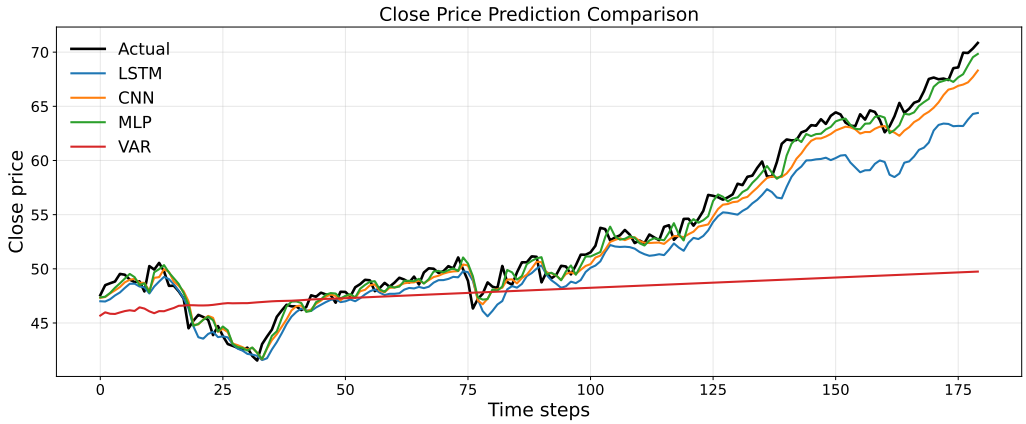

Figure 4: Forecast comparison for close prices on the test set with a 10-day input window.

periods of sharper price movements, highlighting their weaker approximation performance at $p = 10$.

To make the approximation errors more explicit, we next examine the residual series, defined as (actual − predicted) prices. Figure 5 shows the residuals for the low prices for each model, revealing how the errors evolve over the test period.

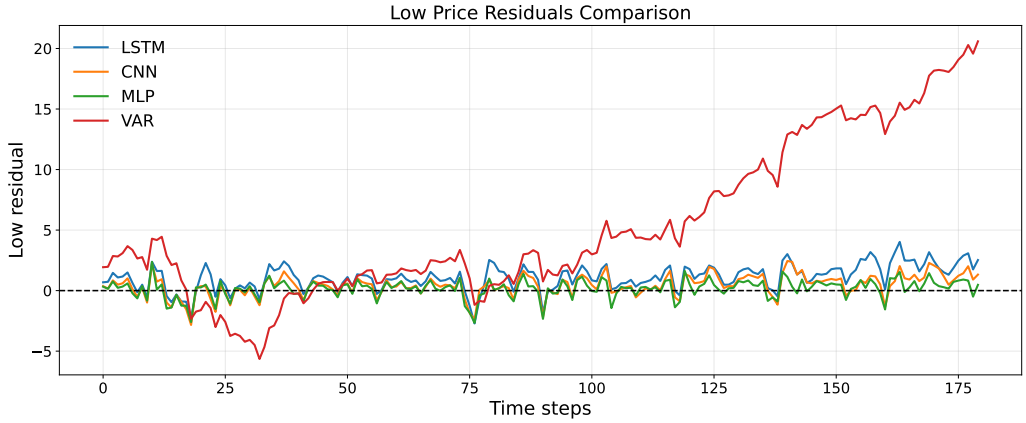

Figure 5: Residuals for low prices on the test set with a 10-day input window.

Figure 6 presents the residuals for the high prices, and Figure 7 provides the corresponding residual plots for the close prices.

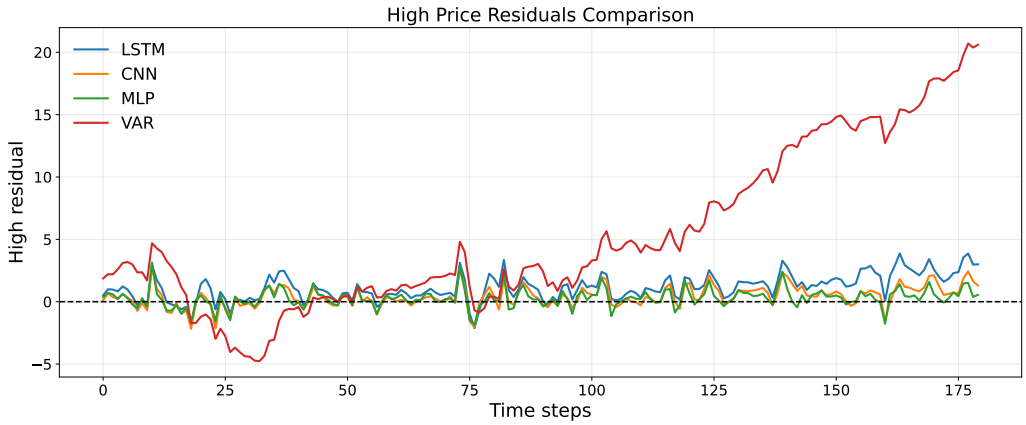

Figure 6: Residuals for high prices on the test set with a 10-day input window.

For all three variables, the MLP residuals are relatively small in magnitude and oscillate

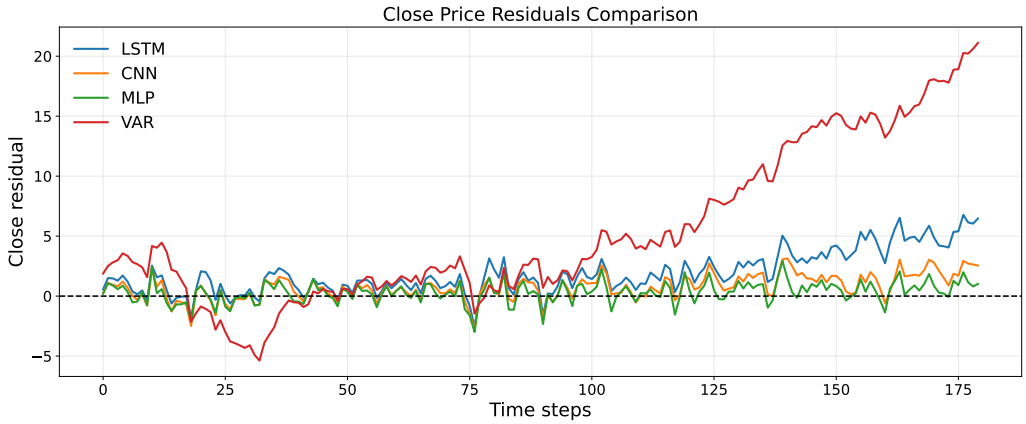

Figure 7: Residuals for close prices on the test set with a 10-day input window.

around zero without pronounced bursts, consistent with an accurate and stable approximation of f^*. CNN residuals are more variable but remain centred near zero. In contrast, LSTM residuals display larger swings, and VAR residuals are both large and volatile, indicating a substantial mismatch between the linear approximation and the true nonlinear dynamics that govern the multivariate AAPL series.

While the residual time series highlight temporal structure, a complementary view is provided by summarising their distributions. Figures 8, 9, and 10 display box plots of the residuals for low, high, and close prices, respectively, for all four models.

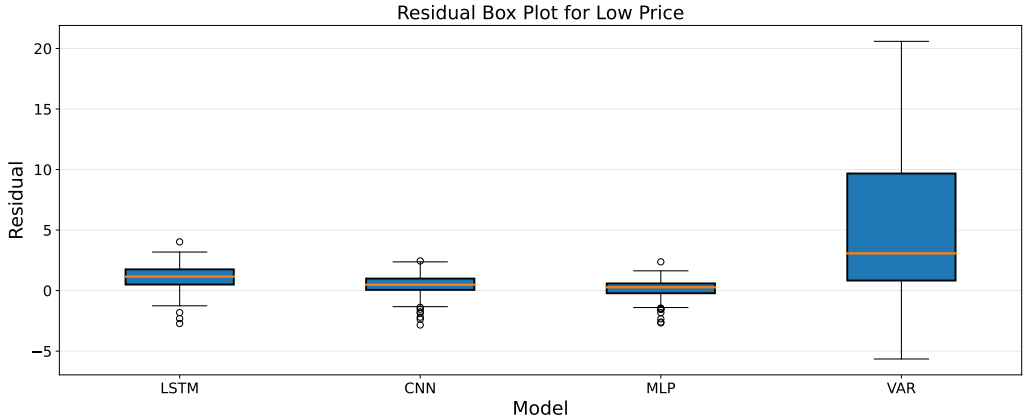

Figure 8: Residual box plots for low prices on the test set with a 10-day input window.

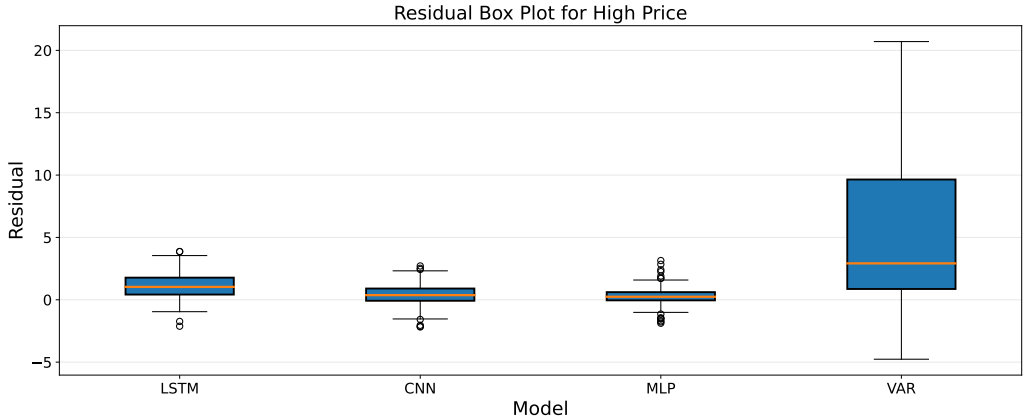

Figure 9: Residual box plots for high prices on the test set with a 10-day input window.

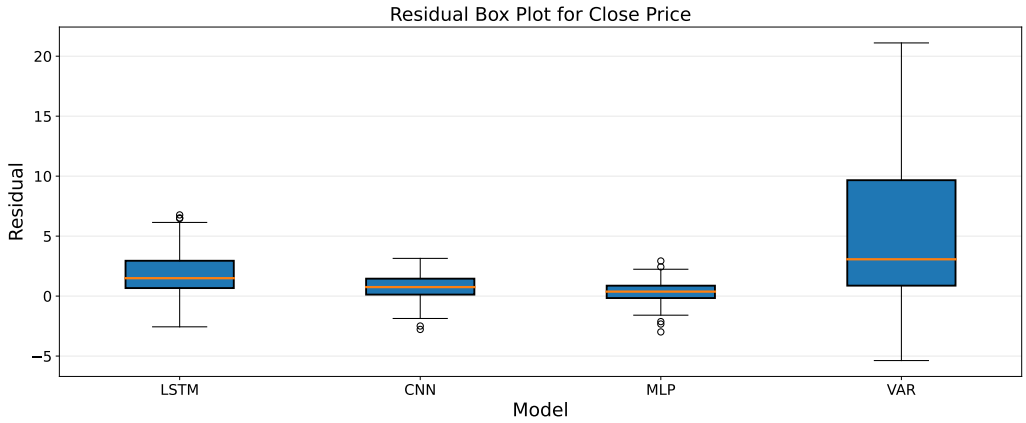

Figure 10: Residual box plots for close prices on the test set with a 10-day input window.

In all three box-plot panels, the MLP residuals exhibit narrow interquartile ranges and medians close to zero, indicating concentrated and approximately unbiased approximation errors. CNN residuals are somewhat more dispersed but still markedly tighter than those of LSTM and VAR. LSTM residuals show wider boxes and more extreme values, while VAR residuals display the broadest distributions and many outliers. This ranking of error spreads (MLP $<$ CNN $<$ LSTM $<$ VAR) mirrors the multivariate metrics in Table 1 and reinforces the conclusion that, at $p = 10$, the MLP architecture has the strongest empirical approximation capability among the models considered. The qualitative pattern is consistent for low, high, and close prices, suggesting that the relative approximation strengths and weaknesses of the architectures are shared across all three components of the multivariate series rather than being driven by a single variable.

3.4 Summary of Approximation Capabilities Across Horizons

The numerical results and residual analyses together reveal a coherent picture of how approximation capabilities depend on both architecture and window length.

For short and medium windows ($p = 5, 10, 30$), the mapping f^* from recent multivariate prices to the next-day vector can be approximated very accurately by a moderately sized MLP. In this regime, the fully connected model attains the lowest empirical approximation error on all multivariate metrics and produces residuals that are small, concentrated, and centred around zero. The forecast and residual plots for the 10-day window (Section 3) visually confirm that f_θ^{MLP} tracks the true series closely for low, high, and close prices, whereas CNN and LSTM provide weaker approximations and VAR performs poorly due to its linearity.

For long windows ($p = 180, 360$), the approximation problem becomes more demanding: the input is high-dimensional and long-range temporal dependencies are likely to matter. In this setting, the LSTM architecture exhibits a clear advantage, achieving substantially lower multivariate errors than MLP, CNN, and VAR. Its recurrent state provides the capacity to represent complex, long-horizon mappings that static architectures struggle to approximate with the same number of parameters. MLP remains a reasonable second-best model in this regime, while CNN and especially VAR show large residual spreads and higher errors.

Overall, the experiments show that approximation capability is not a static property of a model family, but depends critically on the effective horizon and problem structure. On the AAPL case study considered here, MLP is the most efficient approximator for short-to-medium windows, LSTM is the most capable for long windows, CNN occupies an intermediate position, and VAR consistently underperforms because its linear structure cannot capture the nonlinear dynamics of the multivariate series.

4 Conclusion

In this paper, we examined the approximation capabilities of three deep neural network architectures—MLP, 1D-CNN, and LSTM—relative to a conventional VAR model for multivariate time series forecasting of AAPL stock prices (low, high, and close). By framing forecasting as a nonlinear function approximation problem and systematically varying the input window length, we characterised how each architecture behaves across short, medium, and long horizons.

For short and medium windows ($p = 5, 10, 30$), MLP consistently achieved the lowest MSE, RMSE, MAE, and MAPE, with the 10-day MLP providing the best overall approximation. In this regime, the fully connected structure of MLP appears sufficient to capture the relevant nonlinear relationships in the recent multivariate history. CNN performed as a strong, though slightly weaker, competitor, while LSTM did not yet realise its full potential, and VAR performed worst across all metrics. The forecast, residual, and box-plot analyses for the 10-day window further show that MLP yields the tightest and most nearly unbiased error distributions across all three price variables.

For long windows ($p = 180, 360$), the picture changes. LSTM becomes the best-performing model by a substantial margin, achieving the lowest errors on all metrics. This confirms that LSTM's recurrent structure and gating mechanisms provide superior approximation capabilities when the mapping involves long-term temporal dependencies and high-dimensional input histories. MLP remains a reasonable second-best model, while CNN and especially VAR struggle in this long-horizon setting, with broader residual distributions and larger systematic errors.

Taken together, our results provide a nuanced view of approximation capabilities for multivariate financial time series: MLP is an effective and relatively simple choice for short-to-medium windows; LSTM is the preferred architecture for long windows where long-range dependencies are prominent; CNN can be useful when local temporal patterns dominate; and linear models such as VAR are inadequate for capturing the nonlinear dynamics in AAPL prices across all horizons considered.

Future work may extend this analysis in several directions: (i) exploring a wider range of assets and multivariate configurations (e.g., including volume or technical indicators) to assess robustness across datasets; (ii) systematically varying model capacity (depth, width) to study approximation–generalisation trade-offs; (iii) incorporating regularisation techniques and ensembles; and (iv) complementing the empirical study with more formal approximation-theoretic results or synthetic benchmarks where the true mapping is known.

CRediT Authorship Contribution Statement

Mohammad Jamhuri: Conceptualization, Methodology, Software, Data Curation, Formal Analysis, Visualization, Writing–Original Draft Preparation, Writing–Review & Editing, Project Administration. **Mohammad Isa Irawan:** Conceptualization, Methodology, Validation, Formal Analysis, Supervision, Writing–Review & Editing. **Ari Kusumastuti:** Resources, Supervision, Writing–Review & Editing. **Kartick Chandra Mondal:** Methodology, Formal Analysis, Validation, Writing–Review & Editing. **Juhari:** Resources, Supervision, Writing–Review & Editing. All authors have read and approved the final version of the manuscript.

Declaration of Generative AI and AI-assisted Technologies

The authors used generative AI tools to assist in refining the English language, improving the clarity of explanations, and polishing the LaTeX structure of the manuscript. The scientific content, research design, data analysis, model implementation, and interpretation of the results were conceived, developed, and verified by the authors. All outputs generated by AI tools

were critically reviewed and, where necessary, edited by the authors to ensure accuracy and appropriateness.

Declaration of Competing Interest

The authors declare no competing interests.

Funding and Acknowledgments

This research received no external funding.

The authors gratefully acknowledge the support and research environment provided by the Department of Mathematics, Faculty of Science and Technology, UIN Maulana Malik Ibrahim Malang, and the Department of Mathematics, Faculty of Science and Data Analytics, Institut Teknologi Sepuluh Nopember Surabaya. The authors also thank the maintainers of the `yfinance` Python package and Yahoo Finance for providing accessible historical stock price data used in this study.

Data and Code Availability

The multivariate AAPL stock-price data (low, high, close) analysed in this study are publicly available from Yahoo Finance[2] and can be accessed programmatically via the `yfinance` Python package.

The preprocessed datasets and source code used to implement the MLP, 1D-CNN, LSTM, and VAR models, as well as the scripts for generating the figures and tables, are available from the corresponding author upon reasonable request.

References

[1] M. A. Villegas and D. J. Pedregal, "Automatic selection of unobserved components models for supply chain forecasting," *International Journal of Forecasting*, vol. 35, no. 1, pp. 157–169, 2019. DOI: 10.1016/j.ijforecast.2017.11.001.

[2] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 competition: Results, findings, conclusion and way forward," *International Journal of Forecasting*, vol. 34, no. 4, pp. 802–808, 2018. DOI: 10.1016/j.ijforecast.2018.06.001.

[3] G. Bontempi, S. Ben Taieb, and Y.-A. Le Borgne, "Machine learning strategies for time series forecasting," in *European Business Intelligence Summer School*, Springer, 2013, pp. 62–77. DOI: 10.1007/978-3-642-36318-4_3.

[4] E. F. Fama and K. R. French, "Multifactor explanations of asset pricing anomalies," *The journal of finance*, vol. 51, no. 1, pp. 55–84, 1996. DOI: 10.1111/j.1540-6261.1996.tb05202.x.

[5] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019," *Applied Soft Computing*, vol. 90, p. 106 181, 2020. DOI: 10.1016/j.asoc.2020.106181.

[6] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[7] G. P. Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 2003. DOI: 10.1016/S0925-2312(01)00702-0.

[2] https://finance.yahoo.com

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[9] B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2194, p. 20 200 209, 2021. DOI: 10.1098/rsta.2020.0209.

[10] J. F. Torres, R. Hadjout, A. Sebaa, F. Martínez-Álvarez, and A. Troncoso, "Deep learning for time series forecasting: A survey," *Big Data*, vol. 9, no. 1, pp. 3–21, 2021. DOI: 10.1089/big.2020.0159.

[11] A. Borovykh, S. Bohte, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," *arXiv preprint arXiv:1703.04691*, 2017. DOI: 10.48550/arXiv.1703.04691.

[12] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994. DOI: 10.1109/72.279181.

[13] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018. DOI: 10.1016/j.ejor.2017.11.054.

[14] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989. DOI: 10.1007/BF02551274.

[15] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991. DOI: 10.1016/0893-6080(91)90009-T.

[16] J. Sirignano and R. Cont, "Universal features of price formation in financial markets: Perspectives from deep learning," *Quantitative Finance*, vol. 19, no. 9, pp. 1449–1459, 2019. DOI: 10.1080/14697688.2019.1622295.

[17] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and LSTM," *PLOS ONE*, vol. 12, no. 7, e0180944, 2017. DOI: 10.1371/journal.pone.0180944.