Received: 13 June 2026 | Revised: 27 July 2026 | Accepted: 19 August 2026 | Published online: 11 September 2026

RESEARCH ARTICLE

Journal of Data Science and Intelligent Systems
2026, Vol. 00(00) 1–19
DOI: 10.47852/bonviewJDSIS620210864

BON VIEW PUBLISHING

Optimizing Deep Learning for U.S. Technology Stock Price Forecasting Through Data Preprocessing and Hyperparameter Tuning

Fauzan Ghazi[1,*], Syahid Anuar[1], Saharudin Ismail[1] and Mariam Mazlan[2]

[1] *Faculty of Artificial Intelligence, Universiti Teknologi Malaysia, Malaysia*

[2] *Azman Hashim International Business School, Universiti Teknologi Malaysia, Malaysia*

Abstract: This study examines how data preprocessing and hyperparameter tuning affect Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) models for forecasting selected U.S. technology stock prices. Daily Open, High, Low, Close, and Volume (OHLCV) data were collected for seven major technology companies over ticker-specific periods between 2000 and 2025. The models were evaluated using baseline, feature-augmented, Grid Search, and Optuna-tuned configurations. The results show that structured preprocessing and tuning influenced performance, although the effects differed across architectures and stocks. Adding technical indicators did not consistently improve forecasting accuracy and sometimes increased error, particularly in feature-augmented CNN configurations. Optuna achieved lower observed errors than Grid Search in several cases, while Grid Search remained superior for some tickers. LSTM configurations generally achieved stronger aggregate variance explanation, while CNN with raw inputs and Grid Search achieved the lowest aggregate Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). CNN produced higher exploratory Sharpe values in several configurations, but these values were calculated without a complete trading-cost and execution framework. The study contributes an empirical comparison of preprocessing and tuning choices within LSTM and CNN forecasting pipelines. The findings are limited to the selected stocks, fixed validation and test periods, and evaluated neural configurations and do not establish statistical superiority or realizable trading performance.

Keywords: deep learning, stock price forecasting, LSTM, CNN, hyperparameter tuning

1. Introduction

The complexity of financial markets has grown because of innovation cycles and shifting macroeconomic factors. These factors strongly affect the U.S. technology sector. Nonlinear patterns and time-varying volatility arose from rapid price swings, speculative trading, and sensitivity to external shocks.

These conditions can reduce the effectiveness of forecasting frameworks that rely on linearity and stationarity assumptions [1]. Models such as Autoregressive Integrated Moving Average (ARIMA) and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) often struggle during periods characterized by structural breaks or abrupt changes.

Deep learning methods have become increasingly popular because they can learn hidden temporal relationships directly from historical data. Architectures such as Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNN) can capture nonlinear and nonstationary patterns without extensive handcrafted features, providing advantages over traditional linear models [2]. However, their performance in financial forecasting remains inconsistent across volatile assets.

For example, Zhang and Pinsky [3] showed that LSTM models performed differently on S&P 500 and NASDAQ-100 price movement prediction, with model generalization depending strongly on the selected training period and volatility conditions. Inconsistencies stem as much from data preparation and hyperparameters as from architecture. Preprocessing choices, such as scaling, sequence design, and indicator selection, can strongly affect forecasting results. The same applies to hyperparameter tuning, since even small changes in model settings may lead to different levels of accuracy and stability.

Although earlier studies have discussed the importance of preprocessing and tuning, they often treat them as separate steps. This leaves a gap in understanding how both actually work together during model development. This issue is important in the U.S. technology sector because prices can move quickly and small forecasting errors can still matter. A clearer

***Corresponding author:** Fauzan Ghazi, Faculty of Artificial Intelligence, Universiti Teknologi Malaysia, Malaysia. Email: ahmad.fauzan@graduate.utm.my

understanding is therefore needed of how preprocessing and tuning can work together to keep forecasting models stable in volatile markets.

This study looks at how preprocessing and hyperparameter optimization affect LSTM and CNN models when forecasting major U.S. technology stocks. It is based on earlier findings showing that deep learning models can be highly sensitive to how the data are prepared and how the models are tuned [3–5].

It also considers concerns mentioned in the reference [6] about the limited ability of models that ignore structural changes during preparation. Using over two decades of market data, the study gives an overall evaluation of these factors. It offers helpful advice to analysts and researchers seeking reliable, data-driven forecasting tools in unstable financial environments.

The contribution of this study is empirical and procedural rather than architectural. First, it compares minimal and feature-augmented preprocessing within LSTM and CNN forecasting pipelines. Second, it evaluates untuned, Grid Search, and Optuna-based configurations under a common chronological evaluation structure. Third, it documents how preprocessing and tuning effects differ across selected technology stocks with heterogeneous historical coverage and volatility profiles. Finally, it provides a transparent account of the strengths and limitations of the evaluated configurations without claiming a new neural architecture or superiority over external benchmark models.

The framework advances existing LSTM- and CNN-based stock forecasting studies through a controlled within-study comparison of preprocessing depth, tuning strategy, model architecture, and ticker-specific behavior under the same chronological evaluation design. Its contribution lies in evaluating these interacting design choices together rather than proposing a new neural architecture.

The rest of this paper is organized as follows. Section 2 reviews related work on forecasting models, deep learning methods, preprocessing, hyperparameter tuning, and evaluation metrics. Section 3 explains the methodology, including data collection, preprocessing, model development, tuning, and evaluation. Section 4 presents the experimental results and compares model performance. Section 5 discusses the findings, implications, limitations, future research, and conclusion.

2. Literature Review

In the U.S. technology market, the forecasting of stock prices is a difficult task. Stock prices fluctuate a lot; hence, they are considered nonlinear and nonstationary time series. Moreover, technology stocks do not follow the general trend of the overall stock market and are sensitive to the VIX index [1, 7]. Figure 1 summarizes the structure of the literature review and the main categories of stock price forecasting models discussed in this section.

2.1. Traditional statistical models and their limitations

Conventional models such as ARIMA and GARCH remain widely used for their interpretability but often struggle during regime shifts. ARIMA requires stable time series and is sensitive to noise, while GARCH improves volatility modeling but reacts slowly to abrupt changes. Applications to technology stocks and indices show that both methods deteriorate when faced with rapid market movement or structural breaks [8, 9]. These limitations motivate a shift toward methods that can learn nonlinear, nonstationary dynamics.

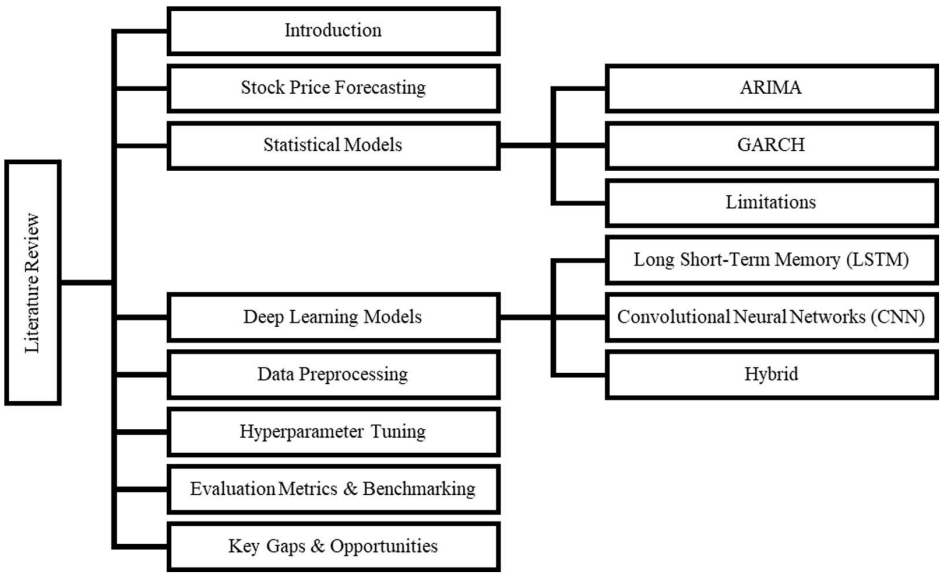
$$y_t = c + \sum_{i=1}^{p} \varphi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t \tag{1}$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1} \tag{2}$$

In the ARIMA equation, (y_t) denotes the observed time-series value at time (t), (c) is a constant, (p) and (q) represent the autoregressive and moving-average orders, respectively, (φ_i) denotes the $(i) - th$ autoregressive coefficient, (θ_j) denotes the $(j) - th$ moving-average coefficient, and (ε_t) is the error term at time (t). The terms (y_{t-i}) and (ε_{t-j}) represent the lagged observations and lagged error terms, respectively.

In the GARCH equation, (σ_t^2) denotes the conditional variance at time (t), (α_0) is the constant variance term, (α_1) measures the effect of the previous squared error, (ε_{t-1}^2), and (β_1) measures the persistence of the previous conditional variance, (σ_{t-1}^2).

Figure 1
Structure of literature review on stock price forecasting models



2.2. Deep learning for financial time series

Deep learning has become a common way to address these limitations. LSTM and CNN models are widely used in stock forecasting because they can learn nonlinear temporal patterns from market data with less manual feature design. For example, Yüksel [10] found that two-layer LSTM models outperformed deeper stacked models in Exchange-Traded Fund (ETF) forecasting, showing that greater model depth does not always lead to better accuracy. Zhang and Pinsky [3] highlighted that tuning training window lengths based on volatility improves generalization for both S&P 500 and NASDAQ-100 data.

Hybrid LSTM models often add techniques that make the model more robust. For example, Olorunnimbe and Viktor [11] discussed models such as ModAugNet, which use data augmentation and regularization layers to reduce overfitting. Similarly, Tuhin et al. [5] proposed an LSTM autoencoder that learns from related stocks to reconstruct the target stock pattern, producing strong results on the NASDAQ stock index.

Bao et al. [6] found that LSTMs generally outperformed baseline methods on U.S. stock indices but were inferior to econometric models in terms of interpretability.

CNNs also appear frequently in this domain. CNNs capture short-term local patterns and are especially effective in hybrid settings. Kumbure et al. [12] reported that CNN layers detect spatial motifs which, when passed to LSTM layers, substantially improve forecasting accuracy compared to standalone models. Olorunnimbe and Viktor [11] documented successful CNN-LSTM hybrids, including DeepLOB and MarketSegNet, where CNNs extract structural representations of limit order book data for LSTM processing. These architectures outperform many single-stage models but remain sensitive to tuning.

2.3. Hybrid deep learning architectures

CNN-LSTM models are the mainstream for modeling both spatial and temporal dependencies. Kumbure et al. [12] pointed out that the CNN-LSTM encoder–decoder architecture increases the model's robustness, particularly in a highly fluctuating environment. Bao et al. [6] also reached the same conclusion in a literature review that CNN-LSTM, BiLSTM-CNN, LSTM autoencoder, and other models consistently perform well on stock indexes. Inception modules in the CNNs, such as DeepLOB, proved the importance of multiscale feature learning [11]. However, it is hard to compare the models, as various datasets and evaluation metrics are used across these studies.

2.4. Data preprocessing practices

Preprocessing is an important step for stabilizing model training. Akşehir and Kılıç [13] found that normalization, denoising, and feature selection are the necessary parts of data preparation in finance. Yilmaz and Yildiztepe [14] employed log returns and z-score normalization on BIST30 stocks, which led to better RMSE results for BiLSTM. Sen and Mehtab [15] adopted a formal framework for missing-value imputation, min-max normalization, and Principal Component Analysis (PCA), leading to above 75% directional accuracy (DA) for both LSTM and BiLSTM. Giantsidi and Tarantola [16] noted that normalization is a widely used technique among the works, but most of them are not transparent about their preprocessing procedures.

The last four studies are related to preprocessing techniques. Korade and Zuber [17] employed PSO and firefly optimization algorithms on a CNN. S et al. [18] applied PCA and min–max normalization to an LSTM model optimized by PSO with an accuracy of over 98. Terven et al. [19] investigated the impact of preprocessing on the loss function of a deep learning model. This paper confirms that preprocessing affects model convergence. Although there is a general consensus that preprocessing is an essential part of an experiment, there is no consistency in how it is reported, making it difficult to reproduce a study.

2.5. Hyperparameter tuning strategies

For hyperparameter tuning, Hoque and Aljamaan [20] adopted a Grid Search strategy to optimize the machine learning models, including Support Vector Regression (SVR), Gaussian Process Regression (GPR), and K-Nearest Neighbors (KNN), and reported that optimizing kernel hyperparameters, learning rate, and regularization strength led to considerable enhancement of the models. Similar improvements were obtained for deep learning models. You [21] used Bayesian optimization using Optuna to optimize the dropout rate and number of units in LSTM and reported an accuracy of 97.5% on the Dow Jones stocks during market turbulence. Peng et al. [22] applied a Grid Search strategy and manual optimization on deep learning models and reported convergence of performance at two accuracy levels due to the interaction of the optimized hyperparameters.

Korade and Zuber [17] compared random search, Particle Swarm Optimization (PSO), and firefly optimization for CNNs, noting lower Mean Squared Error (MSE) with PSO and firefly but also higher computational cost. Zhang and Pinsky [3] validated the benefits of manual tuning for LSTM models by adjusting dropout, batch size, and epoch settings across S&P 500 and NASDAQ-100 thresholds. Reviews by Olorunnimbe and Viktor [11] identify dropout, learning rate, batch size, and depth as the most influential parameters in financial forecasting architectures.

2.6. Evaluation metrics for benchmarking forecasting models

Evaluating forecasting quality requires multiple complementary metrics. The most common measures are Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE), which appear across comparisons of ARIMA, machine learning, and deep learning models [23]. RMSE is useful for high-volatility data because it places greater weight on larger forecast errors. MAE offers a more stable view of the average error, while MAPE makes the results easier to interpret by expressing errors as percentages. For models that focus on predicting market direction rather than exact price values, studies also use DA and classification measures such as precision, recall, and F1 score. Together, these metrics provide a broader view of how well a forecasting model performs.

2.7. Key gaps identified in existing studies

The literature points to several gaps. Conventional models often struggle with market fluctuations, especially in the U.S. technology sector. Deep learning models can perform better, but they remain sensitive to parameter choices and are difficult to interpret. Research on deep learning for technology stocks is also limited, as most studies focus on broader indices or international markets with different volatility patterns.

Most previous studies focus on either preprocessing or hyperparameter tuning, but not both together. This is a gap because both

steps can strongly influence deep learning performance. Treating them as connected parts of model design may help produce more stable forecasting models for fast-moving financial data.

Recent studies on FinTech and artificial intelligence (AI) in finance also stress the importance of reliable data pipelines and adaptive analytics. Bahoo et al. [24] showed that AI is increasingly used in forecasting, risk management, fraud detection, and algorithmic decision-making. Mhlanga [25] highlighted how big data supports more adaptive financial services. Together, these studies support the need for structured preprocessing, systematic tuning, and robust forecasting frameworks in financial AI.

These research gaps motivate the current study, which investigates the mutual impact of data preprocessing techniques and hyperparameter tuning algorithms on the predictive accuracy of LSTM and CNN models in the U.S. technology stock market.

3. Research Methodology

This study proposes a four-stage framework for predicting the U.S. technology stock market through deep learning. The framework includes the following modules: data collection, data preprocessing, model implementation, and result analysis. Given the significant effects of preprocessing and hyperparameter tuning on the model's performance and reliability under changing market conditions, these two modules were prioritized. The framework is an actionable model that aims to strike an optimal balance between theory and practice. Table 1 describes the framework's stages and the goals and tasks associated with each.

3.1. Research design

This study follows a four-phase methodology to develop and evaluate deep learning models for forecasting next-day closing prices of selected U.S. technology stocks. The phases cover data acquisition, preprocessing and feature engineering, model development, and performance evaluation. Together, they form a complete workflow that reflects how modern quantitative investment pipelines are usually built. Raw market data are first collected, cleaned, and converted into structured inputs for learning. These inputs are then processed through advanced models and evaluated using both statistical and financial metrics.

3.2. Phase 1: data acquisition

The first phase focuses on collecting historical daily OHLCV data for the selected technology stocks. Daily OHLCV data were retrieved for major U.S. technology companies, including Apple, Microsoft, Amazon, NVIDIA, Meta, Tesla, and Alphabet. These firms were selected due to their large market capitalizations and high volatility. Data were obtained through Python-based Application Programming Interfaces (APIs). Yahoo Finance was the primary source of adjusted prices.

The overall dataset covers the period from 2000 to 2025. However, the available historical period differs across companies because of their respective initial public offering dates. AAPL, AMZN, MSFT, and NVDA contain observations from 2000, GOOGL from 2004, TSLA from 2010, and META from 2012. No artificial or imputed observations were created for periods before a company became publicly listed. Consequently, the number of training observations differs across tickers.

The pre-2025 data were divided chronologically into a training period covering 2000–2022 and a validation period covering 2023–2024. The January–June 2025 period was retained for final out-of-sample evaluation. This chronological structure exposed the models to several market regimes, including the dot-com aftermath, the 2008 global financial crisis, the COVID-19 market shock, the post-pandemic recovery, and the 2023–2024 AI-driven technology rally.

The long historical period allowed the models to be evaluated after exposure to both stable and volatile market conditions. However, the long horizon does not eliminate survivorship bias. The sample was selected retrospectively from large technology companies that remained prominent during the study period. Delisted, failed, or formerly prominent technology companies were not included. The findings should therefore be interpreted as applying to the selected companies rather than to the complete historical population of U.S. technology firms.

The exact number of usable observations differed across tickers because of listing dates, missing-value handling, indicator lookback periods, and 60-day sequence construction. The retained manuscript records do not include a complete final observation-count table or full descriptive statistics for every engineered feature. The shared repository provides the underlying datasets needed to reconstruct these summaries.

3.3. Phase 2: data preprocessing and feature engineering

The data were cleaned, normalized, and converted into model-ready inputs.

Table 1
Summary of methodology phases, objectives, and key activities

Phase	Objective	Key activities
Phase 1: Data Acquisition	Collect ticker-specific historical stock-price data	Collect ticker-specific daily OHLCV data from Yahoo Finance
Phase 2: Data Preprocessing and Feature Engineering	Prepare and enrich data for modeling	Normalize data, compute log returns and volatility, extract technical indicators, add lag features, manage outliers, and build 60-day input sequences
Phase 3: Model Development	Develop and tune deep learning models	Baseline LSTM and CNN models, feature-augmented configurations, and hyperparameter tuning using Grid Search and Optuna
Phase 4: Model Evaluation	Assess forecasting accuracy and practical performance	Evaluate models using RMSE, MAE, MAPE, R^2, directional accuracy, and Sharpe ratio on the out-of-sample test period

The minimal preprocessing pipeline used OHLCV variables, missing-value handling, normalization, and 60-day sequence framing.

Z-score scaling was used for the LSTM to help stabilize gradients during recurrent learning.

For the CNN models, min–max scaling was applied because it keeps the input values within a fixed range, making them more suitable for convolutional feature extraction.

The forecasting task was formulated as one-step-ahead price regression. Each input contained a rolling sequence of 60 trading days ending on day (t), while the prediction target was the closing price on day ($t + 1$). The models were not trained using a separate binary classification label.

The primary model output was the forecasted next-day closing price. DA was also reported in the original experimental outputs, although the retained documentation does not allow its exact implementation to be independently verified.

Preprocessing was performed separately for each ticker within the chronological training, validation, and testing structure. The 2000–2022 period was used for model training, the 2023–2024 period was used for hyperparameter validation, and the January–June 2025 period was retained for final out-of-sample evaluation.

Because the reported error metrics are expressed on the original price scale, normalized model outputs were converted back to price units before evaluation. The retained report does not document the inverse-transformation implementation in further detail.

The retained project documentation confirms that z-score scaling was used for the LSTM pipeline and min–max scaling for the CNN pipeline. However, complete scaler-fitting logs were not retained. Therefore, the revised study does not claim that the scaler-fitting sequence was independently verified through a formal leakage audit. This is acknowledged as a reproducibility limitation.

All rolling indicators and technical variables were constructed using trailing historical observations. A feature recorded for day (t) was calculated from information available up to day (t) and was used to forecast day ($t + 1$). Centered rolling windows were not used.

The extensive preprocessing pipeline refers to the feature-augmented configuration, where additional technical indicators were added to the raw OHLCV inputs. These indicators included logarithmic returns, 20-day rolling volatility, Relative Strength Index (RSI, Moving Average Convergence Divergence (MACD), Exponential Moving Average (EMA), Bollinger Bands, Average True Range (ATR), On-Balance Volume (OBV), lagged values, and calendar-based features. These variables were intended to capture trend, momentum, volatility, and volume-related signals.

Correlation analysis was used diagnostically to identify potentially redundant indicators and to support the interpretation of feature-augmented model performance. No universal automated correlation threshold was applied for feature elimination.

Stationarity was addressed through log-return transformation rather than through a formally reported stationarity test. Raw stock-price levels are generally nonstationary, while logarithmic returns help reduce trend effects and stabilize variance. The prediction target nevertheless remained the next-day closing-price level. The raw configurations retained normalized OHLCV price information, while the feature-augmented configurations added return-based, momentum, trend, volatility, volume, lagged, and calendar variables.

Because consecutive stock-price levels are highly persistent, price-level R^2 may appear high even when incremental forecasting gains are limited. R^2 was therefore interpreted together with RMSE, MAE, MAPE, DA, and the forecast plots rather than as independent evidence of economic forecasting superiority.

3.4. Phase 3: model development

The third phase involved developing and improving deep learning models to forecast stock prices. The baseline LSTM received an input of 60 trading days by the number of available features. It consisted of one LSTM layer with 64 units and tanh activation, followed by a dropout layer with a rate of 0.20 and a single-unit dense output layer. The model used the Adam optimizer with a learning rate of 0.001 and mean squared error as the loss function.

The baseline CNN received the same 60-day input structure. It consisted of a one-dimensional convolutional layer with 64 filters, a kernel size of 3, and Rectified Linear Unit (ReLU) activation. This was followed by a MaxPooling1D layer with a pool size of 2, a flattening layer, a dense layer with 50 ReLU-activated units, and a single-unit dense output layer. The CNN also used the Adam optimizer with a learning rate of 0.001 and mean squared error as the loss function.

The original project report refers to stacked LSTM and deeper CNN variants as part of model enhancement. However, complete layer-by-layer specifications for those enhanced configurations were not preserved in the retained documentation. The revised manuscript therefore reports the fully documented baseline architectures and the recorded optimized hyperparameters, without assigning a single uniform enhanced architecture to all ticker-specific runs.

First, baseline models were constructed using a single-layer LSTM and a simple one-dimensional CNN to provide a controlled reference point before feature augmentation and hyperparameter tuning. The baseline configuration was not designed as an optimized model from prior literature. Instead, it used standard Keras/TensorFlow implementation settings with only basic architectural choices fixed for comparability. The baseline models used only normalized OHLCV inputs. No technical indicators, feature-augmented variables, Grid Search, or Optuna-based tuning were applied at this stage.

Model improvements were introduced through feature augmentation, hyperparameter tuning, and changes to the model architecture. Grid Search was used for smaller, predefined parameter combinations, while Optuna-based Bayesian Optimization was used for broader, more adaptive searches.

The tuned parameters included LSTM units, CNN filters, kernel size, dropout rate, learning rate, batch size, and number of epochs. The input lookback was fixed at 60 trading days.

Hyperparameter selection was conducted using the 2023–2024 validation period. The final January–June 2025 test period was reserved for out-of-sample evaluation and was not used as the routine optimization objective. The study used one fixed chronological validation window rather than repeated walk-forward or multi-fold time-series cross-validation.

Grid Search evaluated fixed values for the principal hyperparameters. The documented grid included LSTM units of 32, 64, and 128; dropout rates of 0.10, 0.20, and 0.30; learning rates of 0.001, 0.005, and 0.010; batch sizes of 16, 32, and 64; and epoch counts of 10, 20, and 30. The CNN result records also show evaluated filter counts of 32 and 64, kernel sizes of 2 and 3, dropout rates of 0.20 and 0.30, batch sizes of 16 and 32, and 10 training epochs.

Optuna minimized RMSE on the validation period and used 50 trials per ticker. Its documented search ranges included LSTM units from 16 to 256 on a logarithmic scale, dropout rates from

0.10 to 0.50, learning rates from $\left(10^{-5}\right)$ to $\left(10^{-2}\right)$ on a logarithmic scale, batch sizes from 8 to 128, and epoch counts from 10 to 50. The CNN optimization records show selected filter counts between 32 and 128 and kernel sizes between 2 and 5, together with continuous dropout and learning-rate optimization. The hyperparameter search settings used for Grid Search and Optuna optimization are summarized in Table 2.

The retained report does not specify an early-stopping patience value or an Optuna pruning rule. These procedures are therefore not claimed in the revised manuscript. The original experiment also did not retain complete runtime measurements for every trial. Computational cost is consequently discussed qualitatively rather than through a formal timing comparison.

Optuna used 50 trials per ticker and minimized validation RMSE.

In the Optuna search space, the main LSTM parameters were the number of units, dropout rate, learning rate, batch size, and number of epochs. For CNN models, the main parameters were the number of filters, kernel size, dropout rate, learning rate, batch size, and number of epochs.

The recorded selected configurations varied primarily in learning rate, dropout rate, batch size, epoch count, and model capacity. Because the study did not perform a formal hyperparameter sensitivity or ablation analysis, the independent contribution of each parameter cannot be determined from the reported results.

For the feature-augmented setups, the inputs included logarithmic returns, momentum indicators, volatility measures, lagged values, and calendar-based features. These were framed into 60-day trading sequences, with each model trained to forecast the next day's closing price.

The experiments were conducted in Python 3.10 using TensorFlow 2.x in Google Colab Pro+ with a T4 GPU runtime.

Optuna required repeated model training during optimization. Because exact runtime measurements were not retained, no formal claim is made regarding its computational advantage or suitability for real-time deployment.

3.5. Phase 4: model evaluation

Performance was evaluated using statistical forecasting metrics together with an exploratory Sharpe-ratio calculation.

Statistical metrics include RMSE, MAE, MAPE, R^2, and DA. RMSE penalizes large errors and is especially helpful in volatile markets. MAE provides a clear picture of average deviation, while MAPE enables comparison across stocks with different price levels.

R^2 measures the proportion of variation in the observed price level explained by the forecasts. Because daily price levels are highly persistent, R^2 was interpreted together with the error metrics and forecast plots rather than as standalone evidence of incremental forecasting value.

DA values were retained from the original experimental outputs. However, the equation reproduced in the project documentation compares the signs of positive price levels and does not adequately represent next-day price direction. The retained records do not allow the exact implementation used to generate the reported values to be independently verified. Accordingly, DA is treated only as a secondary reported diagnostic, and no conclusions concerning directional trading ability are based on this metric.

$$\mathrm{RMSE} = \sqrt{\left(\frac{1}{n}\right) \times \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{3}$$

$$\mathrm{MAE} = \frac{1}{n} \sum_{t=1}^{n} |y_t - \hat{y}_t| \tag{4}$$

$$\mathrm{MAPE} = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{y_t - \hat{y}_t}{y_t} \right| \tag{5}$$

$$R^2 = 1 - \frac{\sum_{t=1}^{n} (y_t - \hat{y}_t)^2}{\sum_{t=1}^{n} (y_t - \bar{y})^2} \tag{6}$$

Because the exact DA implementation used in the original experiment could not be verified, the following equation is presented only as the recommended definition for future implementation:

$$\mathrm{DA} = \frac{1}{n} \sum_{t=1}^{n} I\left[\mathrm{sign}\,(y_{t+1} - y_t) = \mathrm{sign}\left(y_{t+1} - y_t\right)\right] \tag{7}$$

The Sharpe ratio was included as a simplified exploratory financial metric. The retained experimental documentation provides the Sharpe formula but does not define a complete forecast-to-position rule, execution schedule, position-sizing framework,

Table 2
Hyperparameter search settings

Model	Parameter	Grid Search values	Optuna search
LSTM	Units	32, 64, 128	16–256, logarithmic integer sampling
LSTM	Dropout	0.10, 0.20, 0.30	0.10–0.50
LSTM	Learning rate	0.001, 0.005, 0.010	10^{-5}–10^{-2}, logarithmic
LSTM	Batch size	16, 32, 64	8–128
LSTM	Epochs	10, 20, 30	10–50
CNN	Filters	Recorded selected values: 32, 64	Recorded selected values: 32–128
CNN	Kernel size	Recorded selected values: 2, 3	Recorded selected values: 2–5
CNN	Dropout	Recorded selected values: 0.20, 0.30	Recorded selected values: 0.10–0.35
CNN	Learning rate	Not fully documented	Recorded selected values: 0.0003–0.0043
CNN	Batch size	Recorded selected values: 16, 32	Recorded selected values: 16–64
CNN	Epochs	Recorded selected value: 10	Recorded selected values: 10–30

or portfolio-construction method. The calculation also did not document transaction costs, bid-ask spreads, slippage, market impact, borrowing costs, leverage, or volatility targeting.

The reported Sharpe values should therefore be interpreted only as relative diagnostics under the original simplified assumptions. They should not be interpreted as evidence of realizable trading profitability, implementable portfolio performance, or reduced drawdown after costs.

$$\text{Sharpe Ratio} = \frac{\overline{R_p} - R_f}{\sigma_p} \tag{8}$$

The study used a fixed chronological training, validation, and testing structure. The 2000–2022 period was used for model training, the 2023–2024 period was used for hyperparameter validation, and January–June 2025 was used for final out-of-sample evaluation. The temporal order of the observations was preserved.

The study did not use repeated walk-forward validation, rolling-window validation, or nested time-series cross-validation. The models were not repeatedly retrained across multiple test regimes. The evaluation was therefore designed to compare preprocessing and tuning strategies under one consistent final test period rather than to simulate continuous model deployment.

The results were tabulated and visualized to compare performance across stocks with different volatility profiles.

4. Results

4.1. Dataset and exploratory patterns

The dataset contains daily OHLCV records for seven major U.S. technology stocks from 2000 to 2024, with 2025 used for out-of-sample testing. The enhanced datasets included RSI, MACD, EMA, and Bollinger Bands. The exploratory analysis showed clear differences across the stocks. Apple, Microsoft, and Alphabet were more stable, while Tesla and NVIDIA showed stronger fluctuations and momentum cycles.

Pearson correlation analysis was used diagnostically to identify overlapping technical indicators. The strongest observed relationship was between the Bollinger Band midline and the fast EMA, with coefficients of approximately $r = 0.99$ across several tickers. MACD and RSI also showed notable relationships for META $r = 0.75$, AMZN $r = 0.72$, and NVDA $r = 0.49$. No single automated cutoff was documented as a universal elimination rule. The correlation results were therefore used to interpret potential redundancy and multicollinearity rather than to claim a uniform threshold-based feature-removal procedure.

Since formal Augmented Dickey-Fuller (ADF) statistics were not reported this study does not claim ADF-based stationarity confirmation. Instead, log-return transformation was used to reduce trend effects and stabilize variance before training. These findings guided the later preprocessing and tuning choices.

4.2. Baseline deep learning models results

As defined in Section 3.4, the baseline models were untuned LSTM and CNN configurations trained only on normalized OHLCV sequences. These results establish the reference performance levels for evaluating the effects of feature augmentation and hyperparameter optimization. The baseline RMSE, MAE, and R^2 results for both architectures across all seven stocks are summarized in Table 3.

LSTM reported higher R^2 values for most stocks. This pattern is consistent with the ability of recurrent architectures to model longer temporal dependencies, although the present experiment did not isolate architecture as the sole causal factor.

Both models showed mixed performance across stocks, providing a reference point for evaluating the observed effects of feature augmentation and hyperparameter tuning.

LSTM reported higher R^2 values for most stocks, particularly MSFT, NVDA, and TSLA. The two architectures produced equal R^2 values for AAPL and GOOGL, while CNN achieved lower baseline errors and a higher R^2 for META. These differences show that neither architecture dominated every ticker and that performance varied across company-specific price patterns.

4.3. Tuned deep learning models results

Two tuning strategies were used on both architectures. Grid Search evaluated fixed combinations of selected parameters, while Optuna-based Bayesian optimization adaptively searched a wider parameter space. For LSTM models, Optuna tuned the number of units, dropout rate, learning rate, batch size, and the number of epochs.

For CNN models, Optuna tuned the number of filters, kernel size, dropout rate, learning rate, batch size, and number of epochs.

These parameters were included because they control model capacity and training behavior. However, their independent effects were not evaluated through a formal sensitivity or ablation analysis.

Across the reported configurations, Optuna produced lower errors than Grid Search for several stocks, although Grid Search remained superior in some cases.

4.3.1. LSTM optimization

Grid Search improved LSTM performance across most stocks, especially Amazon, NVIDIA, and Microsoft. Feature-based setups helped in some cases but introduced noise for highly volatile stocks such as Meta.

Table 3
Baseline performance of LSTM and CNN models

Stock	LSTM RMSE	LSTM MAE	LSTM R^2	CNN RMSE	CNN MAE	CNN R^2
AAPL	9.25	6.64	0.74	9.36	6.72	0.74
AMZN	6.7	4.75	0.86	7.53	5.76	0.82
MSFT	14.14	12.13	0.92	21.79	18.62	0.81
NVDA	7.64	6.57	0.93	8.46	7.38	0.91
META	32.65	25.71	0.84	28.35	22	0.88
TSLA	18.35	14.13	0.9	21.09	16.45	0.82
GOOGL	7.43	5.93	0.94	7.57	5.79	0.94

Table 4
LSTM performance after Grid Search and Optuna tuning

Ticker	Grid Search RMSE	Grid Search R^2	Optuna RMSE	Optuna R^2
AAPL	9.91	0.7	6.75	0.86
AMZN	5.54	0.9	5.8	0.89
MSFT	15.82	0.9	18.7	0.86
NVDA	4.92	0.97	6.28	0.95
META	25.36	0.9	30.89	0.86
TSLA	14.7	0.91	12.68	0.94
GOOGL	5.67	0.97	4.94	0.97

Optuna produced strong gains for several LSTM configurations, particularly AAPL, TSLA, and GOOGL. However, Grid Search achieved lower RMSE for AMZN, MSFT, NVDA, and META, showing that no tuning method dominated every ticker.

As shown in Table 4, Optuna reduced RMSE for several tickers and produced R^2 values above 0.90 for NVIDIA, Tesla, and Alphabet.

These results show that Optuna generated competitive LSTM configurations for several tickers within the fixed validation and test structure.

The Optuna forecast plots appeared smoother for several tickers. This observation is descriptive and should be interpreted together with the numerical error metrics rather than as evidence of improved market responsiveness.

Optuna achieved lower RMSE than Grid Search for AAPL, TSLA, and GOOGL, while Grid Search performed better for AMZN, MSFT, NVDA, and META.

4.3.2. CNN optimization

For CNN models, Grid Search produced competitive results for several raw OHLCV configurations, while feature-augmented CNN configurations often reported higher errors and lower R^2, particularly for AAPL and META. Optuna produced lower RMSE than Grid Search for several tickers, although the benefits were not uniform.

The selected CNN configurations differed in filter count, kernel size, dropout rate, learning rate, batch size, and epoch count. Because no formal sensitivity analysis was conducted, the independent effect of each hyperparameter cannot be determined.

As shown in Table 5, Optuna reduced RMSE and improved R^2 for Microsoft, NVIDIA, Meta, and Amazon.

These results show that Optuna produced lower errors for several CNN configurations within the fixed validation and test structure.

Optuna achieved lower RMSE than Grid Search for AAPL, AMZN, MSFT, NVDA, and META. Grid Search remained better for TSLA, while the two methods produced nearly identical RMSE for GOOGL.

4.4. Combined model performance comparison

The aggregated metrics across all configurations clearly compare architectures and tuning strategies.

Table 5
CNN performance after Grid Search and Optuna tuning

Ticker	Grid Search RMSE	Grid Search R^2	Optuna RMSE	Optuna R^2
AAPL	10.65	0.66	9.28	0.74
AMZN	10.1	0.68	7.34	0.83
MSFT	24.8	0.75	13.82	0.92
NVDA	8.75	0.9	5.98	0.95
META	42.25	0.73	28.16	0.88
TSLA	19.96	0.84	26.05	0.73
GOOGL	16.58	0.71	16.59	0.71

Table 6
Averaged error-based metrics across models

Model/setup	LSTM/raw	LSTM/features	CNN/raw	CNN/features
RMSE_Default	17.5	15.3	14.85	28.61
RMSE_Grid	15.95	13.25	10.92	24.81
RMSE_Optuna	15.4	13.07	15.36	27.22
MAE_Default	12.03	10.71	11.87	26.88
MAE_Grid	10.91	10.23	9.97	19.71
MAE_Optuna	13.36	11.93	12.36	17.36
MAPE_Default	3.49	4.77	3.41	7.82
MAPE_Grid	3.71	3.84	3.83	5.98
MAPE_Optuna	4.03	3.28	4.39	5.06

4.4.1. Error and variance metrics

Tuning improved several configurations, although the benefits were not uniform across architectures, input sets, and tuning methods. LSTM generally retained stronger aggregate R^2 values, while CNN with raw inputs and Grid Search achieved competitive aggregate error values. Feature-augmented CNN configurations produced the weakest aggregate results. The averaged RMSE, MAE, and MAPE results across the model configurations and tuning methods are summarized in Table 6.

The weaker performance of feature-engineered configurations suggests that the added indicators introduced redundant information rather than independent predictive signals. Several indicators measured overlapping market behavior, including trend, momentum, and volatility.

For example, EMA and Bollinger Bands both reflect trend-related movement, while RSI and MACD both capture momentum conditions. This overlap increased multicollinearity and raised the risk of overfitting, especially for CNN models that are sensitive to local input patterns.

As a result, feature augmentation sometimes weakened generalization instead of improving forecast accuracy. Table 7 presents the averaged R^2 values across the baseline, Grid Search, and Optuna configurations.

Table 7
Variance explanation (R^2)

Model/setup	R^2 (Default)	R^2 (Grid Search)	R^2 (Optuna)
LSTM/raw	0.87	0.886	0.887
LSTM/features	0.781	0.871	0.881
CNN/raw	0.845	0.794	0.824
CNN/features	0.402	0.551	0.53

LSTM had the highest variance explained. CNN performed well only with raw inputs; engineered features reduced stability.

4.4.2. Directional accuracy and Sharpe ratio

DA values were reported near 0.50 across most configurations. However, because the retained documentation does not allow the exact DA implementation to be independently verified, these values are treated only as secondary reported diagnostics. They are not used to support claims of directional forecasting ability.

CNN configurations produced higher reported Sharpe values than LSTM configurations in several cases. Nevertheless, the Sharpe calculation was not based on a fully documented trading strategy and did not include transaction costs, slippage, position sizing, leverage, volatility targeting, or execution timing. These values therefore provide only a simplified relative comparison within the original experiment.

The reported Sharpe values do not establish that CNN generated more profitable, more stable, or more implementable trading signals. They indicate only that CNN received higher values under the original undocumented financial calculation. The reported DA and Sharpe ratio values across all model configurations are summarized in Table 8.

4.4.3. Summary of insights

No single configuration dominated every metric. CNN with raw inputs and Grid Search achieved the lowest aggregate RMSE and MAE. In contrast, feature-augmented LSTM with Optuna achieved the lowest aggregate MAPE, and raw LSTM with Optuna achieved the highest aggregate R^2.

4.5. Integrated cross-model visualization

The cross-model visualizations summarize the results for error, R^2, directional accuracy, and Sharpe ratio. Optuna improved several LSTM and CNN configurations, although its benefits varied by ticker and architecture. LSTM remained stronger in R^2, while CNN produced higher exploratory Sharpe values in several configurations.

The reported DA values changed only slightly, but no substantive conclusion is drawn because the exact DA implementation could not be independently verified.

4.6. Forecast visualization and temporal alignment

Forecast plots showed differences in smoothness and alignment across architectures and tuning configurations. Figure 2 shows the LSTM forecasts across all tickers, while Figure 3 presents the corresponding CNN forecasts. Figure 4 shows the LSTM forecasts with feature augmentation, followed by the feature-augmented CNN forecasts in Figure 5. The tuned LSTM forecasts obtained using Grid Search and Optuna are presented in Figure 6, while the corresponding tuned CNN forecasts are shown in Figure 7.

LSTM forecasts generally appeared smoother, while CNN forecasts displayed greater short-term variation. Feature augmentation improved visual alignment for some stocks but weakened it for others. These visual patterns are descriptive and should be interpreted together with the reported error metrics rather than as evidence of trading responsiveness or regime adaptation.

5. Findings, Discussion, and Conclusion

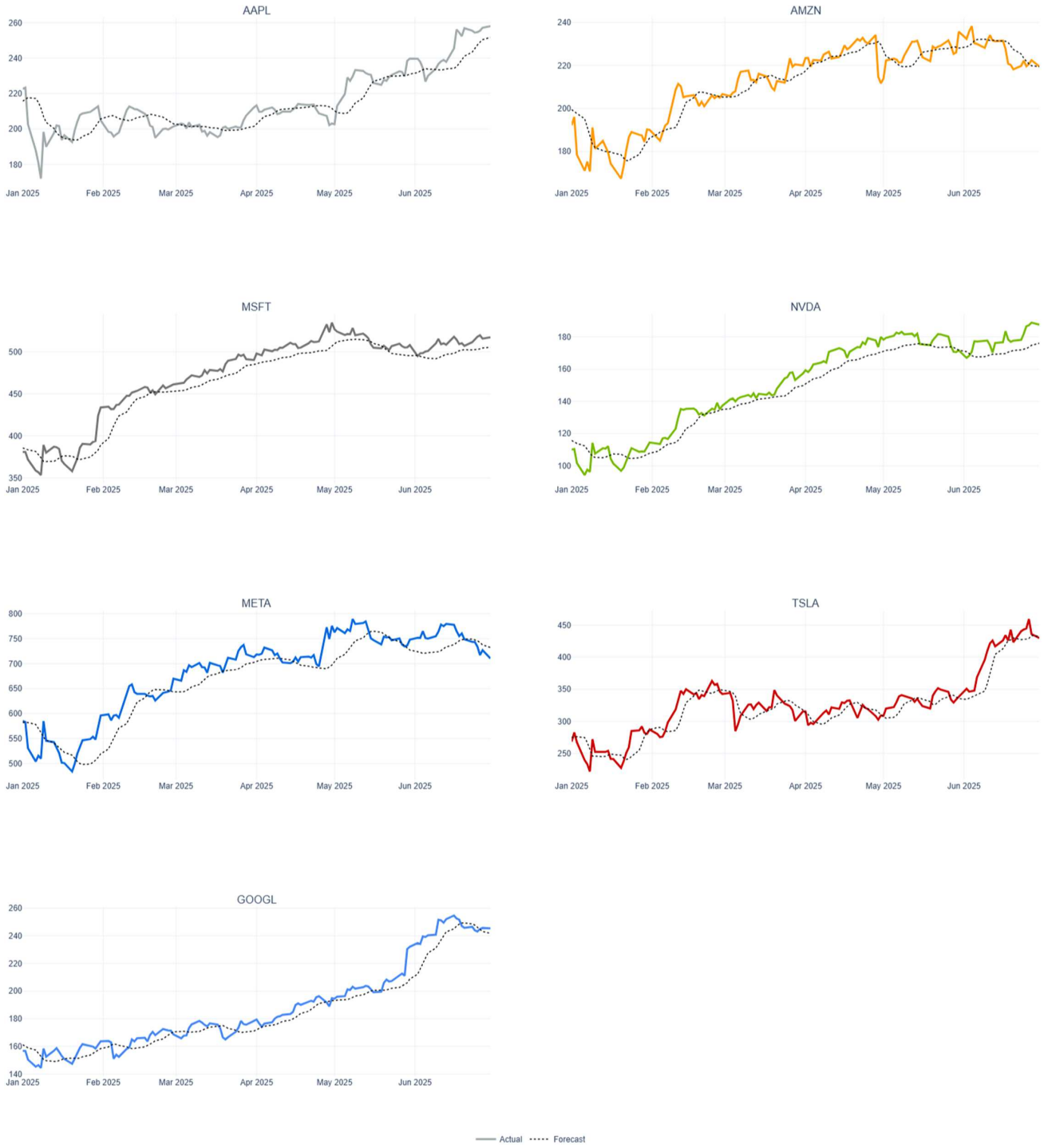
The results highlight how preprocessing and hyperparameter tuning influenced the reported performance of LSTM and CNN models on volatile U.S. technology stocks.

The discussion links these findings to prior research, notes limitations, and suggests future directions. Overall, the evidence shows that preprocessing and tuning are key parts of model design.

Table 8
Directional accuracy (DA) and Sharpe ratio (SR)

Model/setup	DA (Default)	DA (Grid Search)	DA (Optuna)	SR (Default)	SR (Grid Search)	SR (Optuna)
LSTM/raw	0.512	0.49	0.507	0.276	0.245	0.224
LSTM/features	0.506	0.52	0.511	0.256	0.24	0.222
CNN/raw	0.51	0.52	0.508	0.368	0.326	0.336
CNN/features	0.45	0.509	0.489	0.227	0.265	0.307

Figure 2
LSTM: actual and forecast values (all tickers)



The mixed results across Grid Search and Optuna highlight the importance of systematic hyperparameter evaluation rather than reliance on a single tuning method.

5.1. Summary of key findings

The experiments showed how preprocessing, tuning, and model architecture shaped forecasting performance.

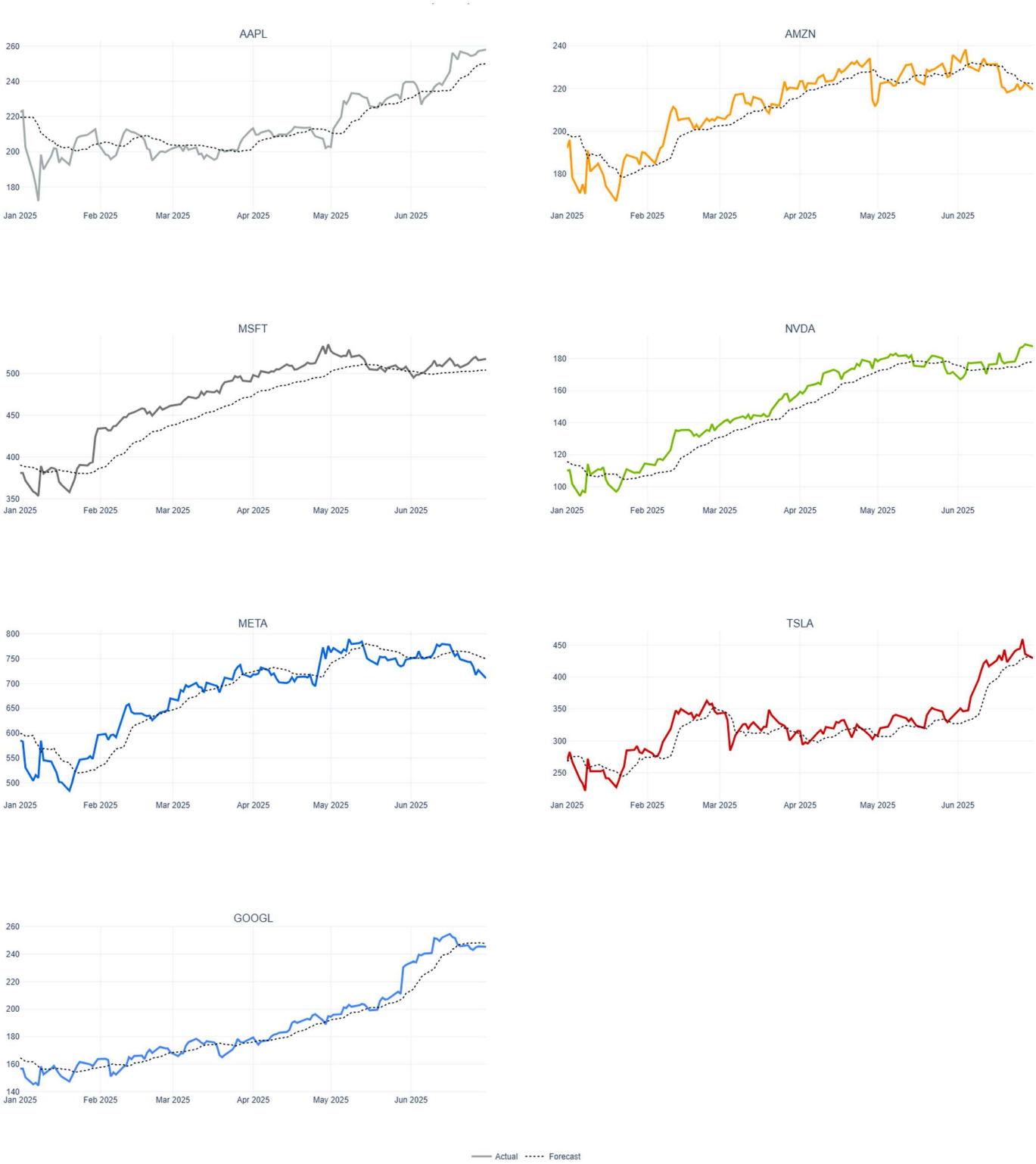
5.1.1. Impact of data preprocessing

Preprocessing had a clear effect on model performance.

The minimal pipeline used normalized OHLCV inputs and 60-day sequence framing, while logarithmic returns and technical indicators were included in the feature-augmented configurations.

However, adding engineered features did not always improve accuracy. Although indicators such as RSI, MACD, EMA, Bollinger Bands, rolling volatility, ATR, and OBV were added to capture momentum, trend, volatility, and volume effects, some of

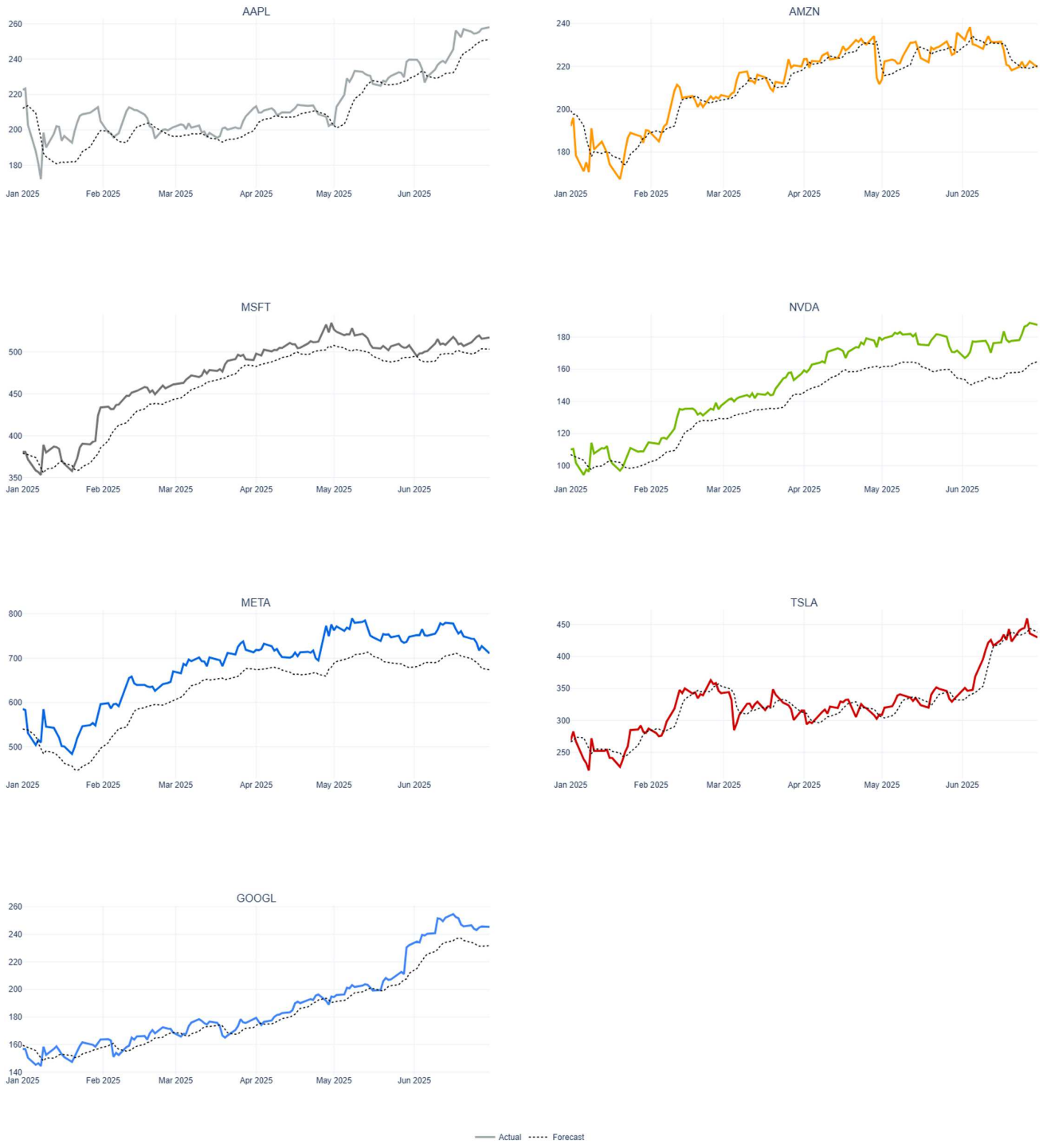
Figure 3
CNN: actual and forecast values (all tickers)



them carried overlapping information. The correlation analysis also showed strong links between several trend and momentum indicators, suggesting possible multicollinearity.

This helps explain why feature-heavy inputs sometimes weakened generalization, especially for volatile stocks such as TSLA and META. Instead of adding useful signals, overlapping indicators may have increased short-term noise and overfitting risk. CNN models were more sensitive to this problem because their filters respond strongly to local patterns in the input sequence. The feature-augmented LSTM configurations were less adversely affected than the corresponding CNN configurations in several aggregate comparisons. Overall, the results suggest that clean and well-structured inputs matter more than simply adding more features.

Figure 4
LSTM (with features): actual and forecast values (all tickers)



5.1.2. Effect of hyperparameter tuning

Hyperparameter tuning materially affected model performance and improved several configurations.

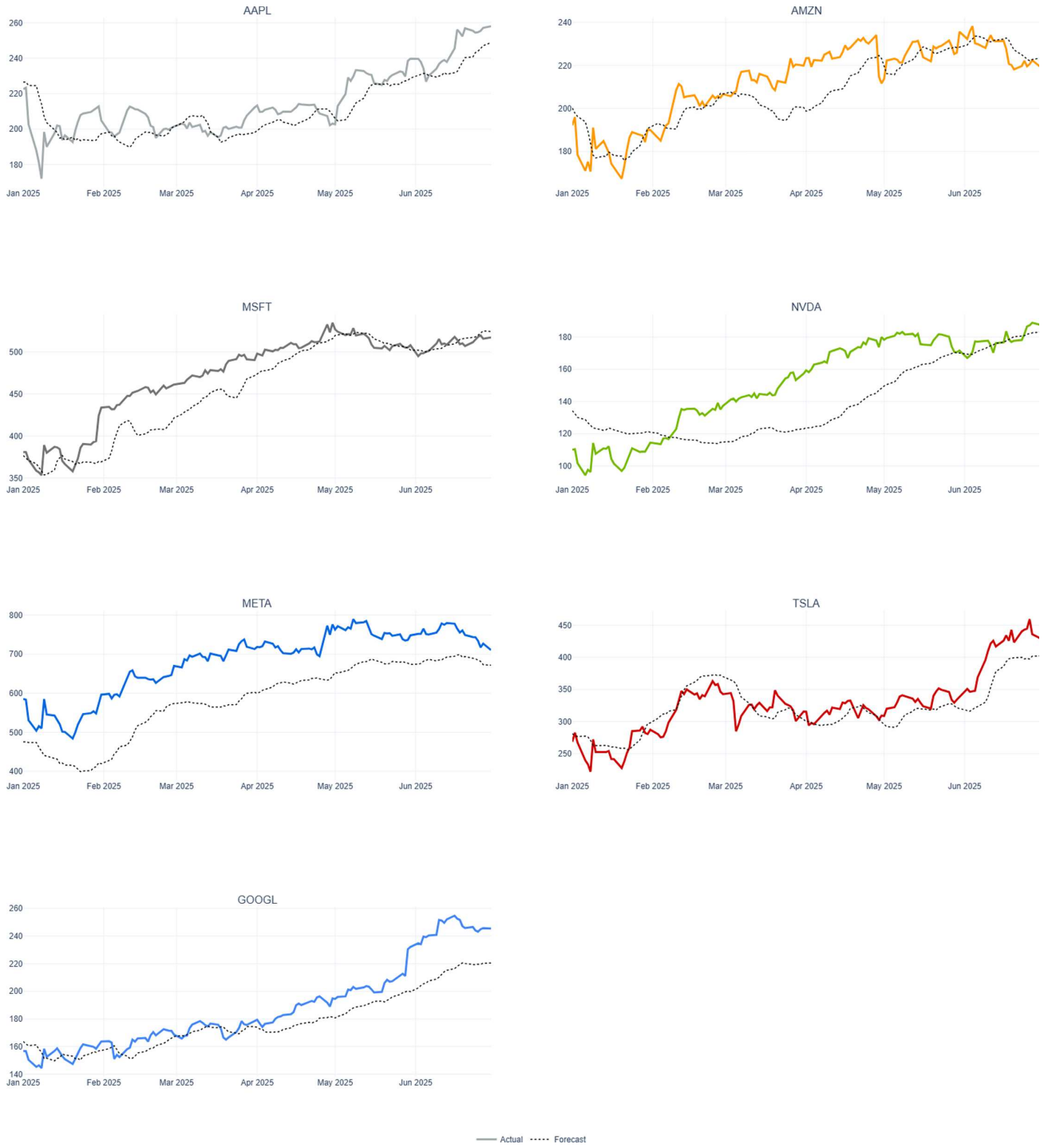
Bayesian optimization provided an adaptive alternative to Grid Search and achieved lower observed errors in several configurations, although Grid Search remained superior for some tickers.

Optuna reduced RMSE and improved R^2 in several evaluated configurations by tuning key parameters such as learning rate, dropout rate, model capacity, batch size, and epochs.

Its forecast plots appeared smoother in several configurations, although smoothness was not evaluated using a separate quantitative measure.

The Optuna-tuned LSTM models achieved an R^2 of 0.95 for NVDA and 0.94 for TSLA. CNN models also received higher reported exploratory Sharpe values for some tickers, including MSFT and GOOGL.

Figure 5
CNN (with features): actual and forecast values (all tickers)



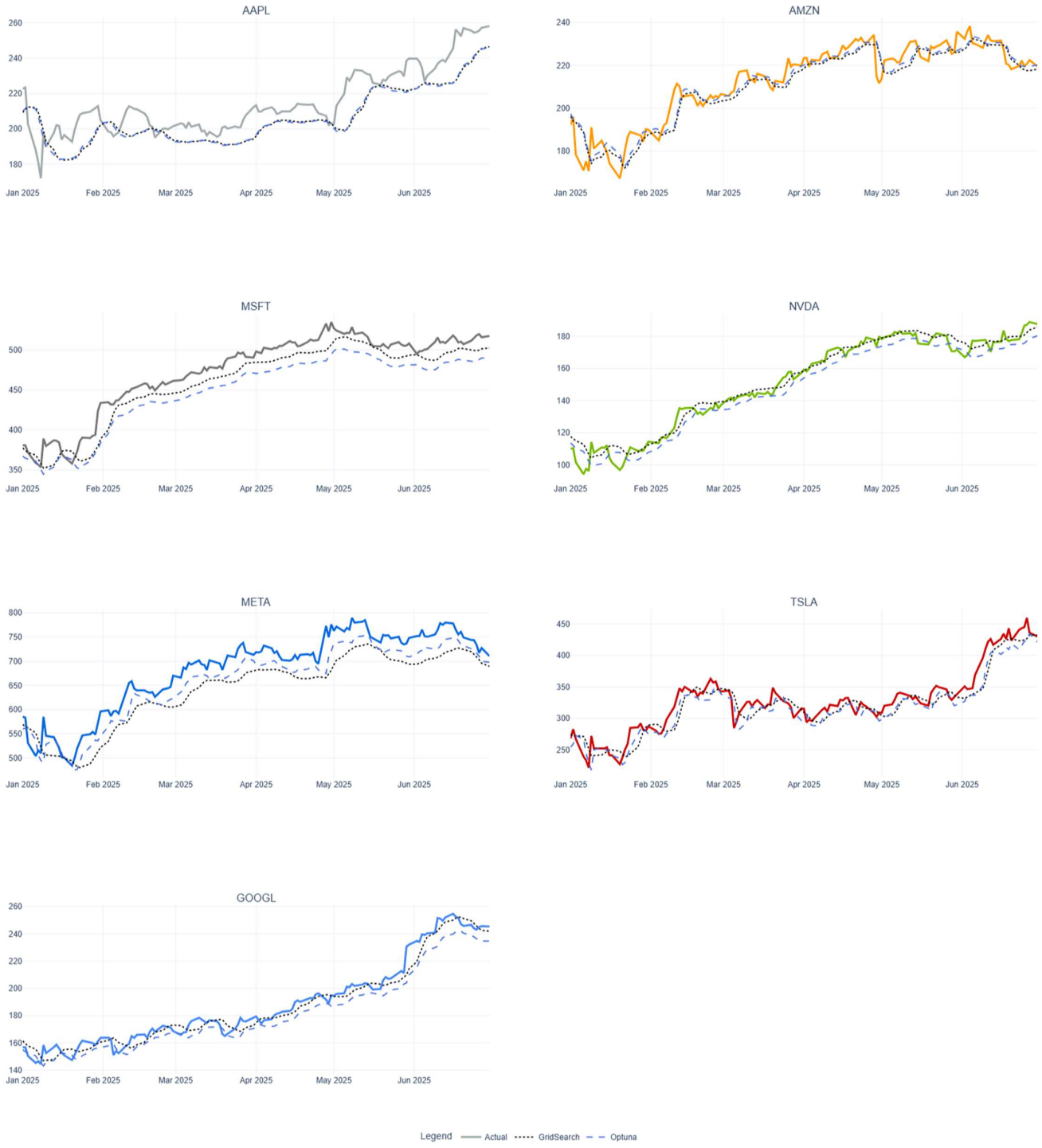
The observed results indicate that both tuning methods generated competitive configurations, but their relative performance varied by ticker and was not confirmed through repeated runs or statistical testing.

5.1.3. Comparative model performance

CNN models produced higher exploratory Sharpe values in several configurations. However, these values were calculated without a complete trading execution and cost framework. They should therefore be interpreted only as relative results within the original simplified calculation.

The Sharpe results do not establish that CNN-based strategies would deliver smoother profits, reduced drawdowns, or superior portfolio performance after implementation costs. A complete financial evaluation would require an explicit position rule, execution timing, transaction costs, slippage, leverage assumptions, drawdown analysis, and other risk measures such as the Sortino ratio.

Figure 6
LSTM (Grid Search and Optuna): actual and forecast values (all tickers)



5.2. Discussion of findings

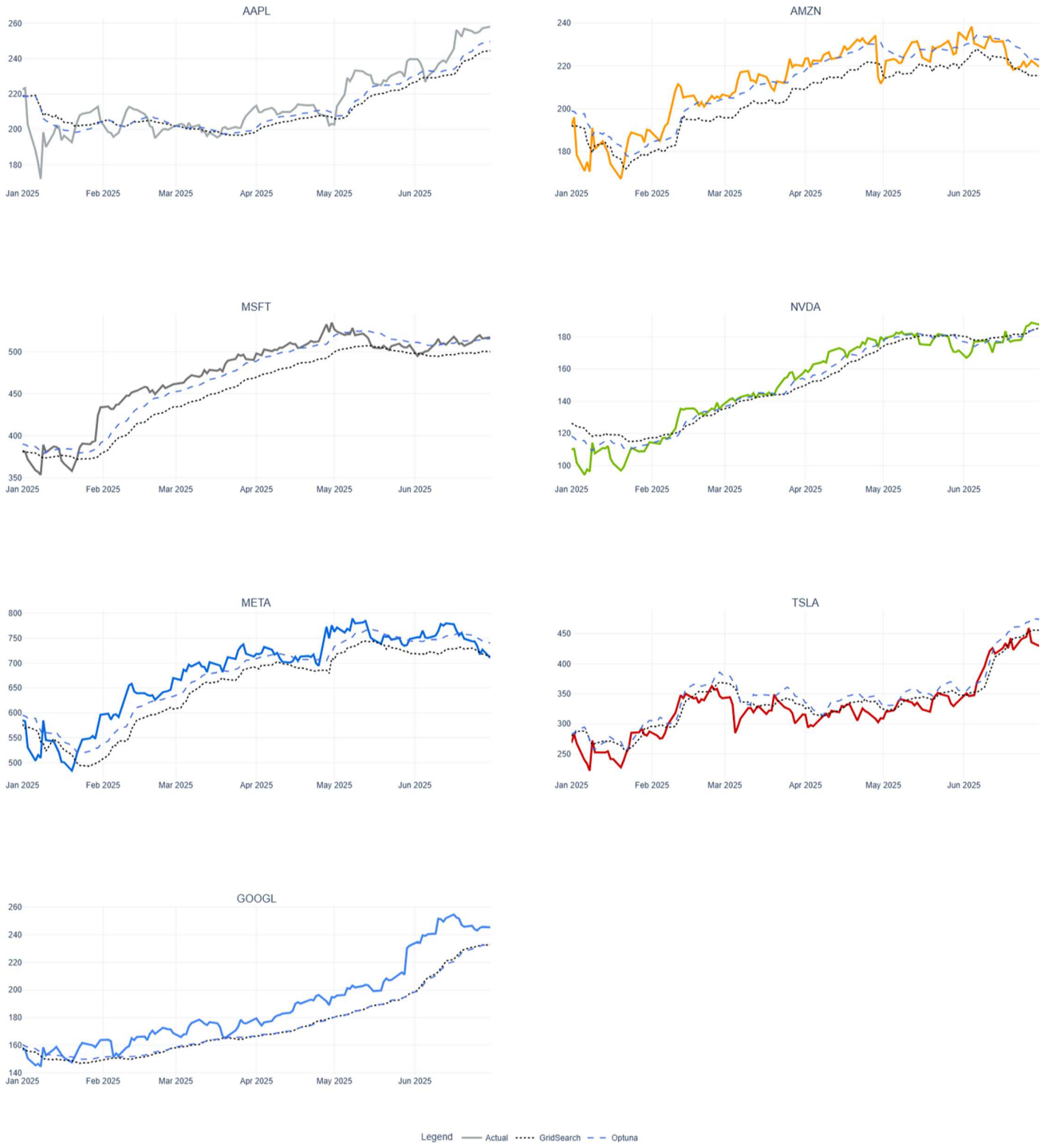
5.2.1. Preprocessing as a determinant of model stability

The results show that preprocessing choices were associated with different forecasting outcomes. Normalization and sequence framing provided a consistent input structure across the evaluated models. This is consistent with earlier studies that identify normalization, noise reduction, and feature selection as important components of deep learning-based stock forecasting [13, 14].

However, feature-heavy datasets did not consistently improve forecasting accuracy. This challenges the idea that adding more technical indicators automatically improves model performance. The weaker performance of several feature-augmented setups may be associated with redundancy and overfitting risk.

Several indicators carried overlapping information. For example, EMA and Bollinger Bands both reflected trend-related movements, while RSI and MACD both captured momentum conditions. When these indicators were used together, they

Figure 7
CNN (Grid Search and Optuna): actual and forecast values (all tickers)



increased redundancy in the input space and made the temporal signal less clear. This issue was more visible in CNN models because convolutional filters are sensitive to local input patterns.

Redundant indicators may increase short-term input variation and make it more difficult for the model to distinguish independent predictive information. This supports [22], which showed that technical indicators and feature selection can strongly affect deep neural network performance in stock direction forecasting.

In the aggregate comparisons, feature-augmented LSTM configurations were less adversely affected than feature-augmented CNN configurations. Overall, these findings support a simpler approach to feature design. For volatile financial data, clean and well-structured inputs may generalize better than large feature sets.

The interaction between feature redundancy, model capacity, and finite-sample learning may explain the poorer performance of several feature-augmented configurations. When highly correlated indicators represent similar trend or momentum behavior, they increase the dimensionality of the input without contributing proportionate independent information.

This can make model training less stable and encourage the learning of relationships that are specific to the training period.

The effect may be more pronounced in CNN models because convolutional filters learn local combinations across the input channels; redundant indicators can therefore produce repeated patterns that appear useful during training but generalize poorly.

This interpretation is consistent with Dehghani and Larijani [26], who combined the Mutual Information Difference (MID) variable-selection algorithm with a neural network for stock-index prediction. Their approach supports the principle that structured selection and integration of financial and economic variables may be more effective than unrestricted feature inclusion. Similarly, Pazouki et al. [27] emphasized that the value of big data in FinTech depends on data quality, suitable processing, infrastructure, and analytical design rather than data volume alone.

5.2.2. The efficacy of Bayesian optimization

Bayesian optimization provided an adaptive approach to hyperparameter selection by using previous trial outcomes to guide later evaluations. Within the fixed validation and test structure, Optuna produced lower observed errors in several LSTM and CNN configurations. However, Grid Search remained superior for some tickers, and the differences were not evaluated through repeated seeds, confidence intervals, or formal statistical tests. The results therefore support Optuna as a useful tuning method within this experiment, but they do not establish universal superiority over Grid Search.

The structure of the optimization problem can also explain the mixed performance of Grid Search and Optuna. Grid Search evaluates a fixed set of predefined combinations and may perform well when suitable values are already represented in the selected grid. Optuna explores a broader and partly continuous parameter space by using the results of earlier trials to guide later evaluations.

This can identify competitive configurations more efficiently, but it may also favor configurations that are particularly adapted to the single validation period. The relative performance of the two methods therefore depends on the search-space boundaries, interactions among hyperparameters, stochastic training behavior, and the market regime represented by the validation data.

A related principle appears in the work of Salimipour and Ebadi [28], who integrated machine learning techniques with multilevel Monte Carlo simulation for European option-price prediction. Although option pricing is different from stock price forecasting, their study demonstrates that financial prediction performance is influenced by the interaction between the learning model and the wider computational and numerical framework rather than by model architecture alone.

5.2.3. Architectural implications

LSTM and CNN models performed differently across market conditions. LSTM produced stronger aggregate (R^2) values, a pattern consistent with the temporal-memory structure of recurrent models, although architecture was not isolated as the sole causal factor.

LSTM and CNN showed different error patterns across the selected stocks. LSTM generally achieved stronger aggregate R^2 values, while CNN with raw inputs and Grid Search achieved the lowest aggregate RMSE and MAE. CNN also produced higher reported exploratory Sharpe values in several configurations. Because the Sharpe calculation was not supported by a complete documented trading framework, no causal architectural explanation or claim of realizable risk-adjusted superiority is made.

These differences support the future evaluation of hybrid models that combine LSTM's long-range memory with CNN's short-term pattern detection. Such models could be evaluated to determine whether combining local and long-term temporal learning improves forecasting accuracy under a more rigorous validation framework.

5.2.4. Sectoral and market insights

The selected technology stocks behaved differently. AAPL, MSFT, and GOOGL were comparatively more stable, whereas TSLA, NVDA, and META displayed stronger fluctuations. These differences indicate that tuning outcomes were influenced by ticker-specific volatility, price persistence, momentum structure, and sensitivity to changing market conditions.

The observed differences cannot be attributed solely to model architecture. Highly volatile stocks may contain stronger short-term patterns that CNN filters can capture, but these patterns may also change rapidly and weaken out-of-sample generalization. LSTM models may produce smoother forecasts because their recurrent memory combines information across the 60-day sequence. However, this temporal aggregation may also reduce responsiveness to abrupt price changes. The relative performance of the two architectures therefore reflects an interaction among model structure, input representation, ticker characteristics, and the market regime represented by the validation and test periods.

Behavioral conditions may also explain part of the stock-specific variation that cannot be represented by OHLCV data and technical indicators alone.

Mojtahedi et al. [29] examined the relationship between the MAX effect and investor sentiment in the Swedish stock market. Their findings support the broader view that investor preferences for extreme-return stocks and the pricing of such stocks may vary across sentiment conditions. This is relevant to the present results because highly volatile technology stocks may be more sensitive to speculative behavior and changing investor sentiment. However, sentiment variables were not included as direct model inputs.

More broadly, Pazouki et al. [30] reviewed the transformative role of AI and digital technologies in FinTech, including their use in predictive analytics, risk assessment, automation, and financial decision-making.

Pazouki et al. [27] also highlighted the role of big-data analytics in real-time decision-making and financial-service innovation. Together, these studies indicate that forecasting performance depends not only on the predictive algorithm but also on data quality, infrastructure, governance, variable integration, and responsible implementation. The current findings therefore support carefully controlled preprocessing and tuning while indicating that future models should evaluate sentiment, macroeconomic, and broader market variables alongside price-based inputs.

5.3. Practical implications

The findings are relevant to the design and evaluation of financial forecasting models.

They show that deep learning models can support volatile-market forecasting when the data are prepared well and the models are tuned carefully.

The reported workflow provides a documented basis for future replication, although complete reproduction would require additional implementation records, including scaler fitting, random seeds, and full enhanced architecture specifications.

Adaptive tuning produced lower observed errors in several configurations, although stability and overfitting reductions were not formally assessed through repeated runs.

Because runtime was not formally recorded, the computational suitability of Grid Search and Optuna for offline, periodic, or real-time deployment could not be established.

The study improves transparency through clear reporting of the preprocessing, tuning, and evaluation steps. It does not directly explain individual LSTM or CNN predictions.

5.4. Limitations of the study

This study has several limitations. First, the analysis focused on selected large U.S. technology stocks. The findings may not generalize to other industries, smaller firms, delisted companies, or international markets. The available historical period also differed across stocks because GOOGL, TSLA, and META entered the sample only after their respective public listings. No pre-IPO observations were created. Nevertheless, the retrospective selection of prominent surviving companies introduces potential survivorship and ex-post selection bias.

Second, the models forecasted next-day closing-price levels rather than next-day returns. Price levels are highly persistent, and this may produce comparatively high R^2 values even when incremental forecasting gains are modest. R^2 should therefore be interpreted together with RMSE, MAE, MAPE, and the forecast plots. The study did not calculate persistence-relative measures such as Theil's U or Mean Absolute Scaled Error (MASE).

Third, the study used a fixed chronological structure consisting of training from 2000–2022, validation from 2023–2024, and final evaluation during January–June 2025. Repeated walk-forward validation, rolling-window validation, and nested time-series cross-validation were not applied. The findings may therefore be influenced by the specific market regime represented by the 2025 test period. Future work should use expanding or sliding training windows, refit preprocessing operations within each window, tune models on a later validation segment, and evaluate forecasts on the next non-overlapping period.

Using a single fixed validation and test structure also creates a risk of regime-specific overfitting. Hyperparameters selected using the 2023–2024 validation period may favor patterns associated with that market environment, while performance during January–June 2025 may not represent other volatility, interest-rate, or market-regime conditions. Although the final test period was not used as the routine optimization objective, the absence of repeated evaluation windows limits evidence of temporal robustness.

A future walk-forward design should divide the data into multiple sequential training, validation, and test blocks. For each iteration, the training window should either expand or slide forward; all preprocessing operations should be refitted using only the current training block; hyperparameters should be selected on the following validation block; and performance should be measured on the next non-overlapping test block. Results should then be aggregated across all test windows to evaluate performance under different market regimes.

Fourth, complete scaler-fitting logs were not retained in the original experimental records. Although the design used chronological training, validation, and testing periods, the study does not claim verification through an independent leakage audit. Future implementations should explicitly record the fitting of every scaler, winsorization boundary, feature-selection rule, and transformation using training data only before applying the fitted operations to validation and test observations.

Fifth, each configuration represents the recorded outcome from the original experiment. The models were not repeatedly trained across multiple random seeds, and standard deviations, confidence intervals, or bootstrap intervals were not calculated. Formal forecast-comparison tests such as the Diebold–Mariano test, paired *t*-test, Wilcoxon signed-rank test, and Friedman test were not conducted. The reported differences should therefore be interpreted as observed performance differences rather than statistically confirmed superiority.

Sixth, the study's baseline models were untuned LSTM and CNN configurations. Naïve last-close, linear autoregressive, ARIMA, XGBoost, Transformer, and other external benchmarks were not included in the experiment. The results consequently demonstrate comparative differences among the evaluated LSTM and CNN configurations but do not establish superiority over persistence, econometric, machine learning, or recent deep learning alternatives.

Seventh, the reported Sharpe ratio was based on a simplified calculation rather than a complete trading-system backtest. The retained documentation does not define a complete signal-to-position rule, execution timing, leverage level, volatility target, or dynamic position-sizing method. Transaction costs, slippage, bid-ask spreads, market impact, borrowing costs, and short-selling constraints were not modeled. The Sharpe values should therefore be treated as exploratory relative diagnostics rather than realizable investment returns.

Eighth, the DA measure retained from the original experiment was not documented as a return-based classification measure. Its financial interpretation is therefore limited, and future studies should calculate direction from next-day price changes or returns using a clearly defined decision threshold.

Ninth, exact runtime measurements were not retained for every Grid Search and Optuna trial. Although Optuna used adaptive sampling and 50 trials per ticker, its computational efficiency relative to Grid Search was not formally benchmarked. Claims concerning exact percentage reductions in runtime or trial count are therefore avoided.

The retained documentation also does not include a complete source-code package, exact random-seed records, a final observation-count table, or full descriptive statistics for every engineered feature. Although the shared repository contains the main datasets and chronological splits, complete computational reproduction would require these additional records.

Finally, the study used daily data and did not directly integrate macroeconomic and sentiment variables into the final model inputs. LSTM and CNN models also remain difficult to interpret. Explainability tools such as SHapley Additive exPlanations (SHAP), Local Interpretable Model-Agnostic Explanations (LIME), and Integrated Gradients were not applied.

5.5. Recommendations for future research

Future research should explore Optuna-optimized CNN-LSTM models to combine short-term pattern detection with long-term temporal learning. Future studies should also apply explainable AI methods, such as SHAP, LIME, or Integrated Gradients, to interpret feature contributions and model behavior.

5.6. Conclusion

This study examined how preprocessing and hyperparameter tuning affect deep learning forecasts of volatile U.S. technology stocks.

The results show that preprocessing and tuning choices materially affected the reported forecasting metrics, although

their benefits varied across stocks, architectures, and evaluation measures.

Minimal preprocessing generally produced more consistent results than extensive indicator engineering, particularly for CNN configurations.

Within the evaluated configurations and fixed test period, Bayesian optimization produced lower observed errors than Grid Search in several cases.

Within the fixed experimental setting, LSTM configurations generally achieved stronger aggregate R^2 values, while CNN with raw inputs and Grid Search achieved the lowest aggregate RMSE and MAE. CNN configurations also received higher exploratory Sharpe values in several cases under the simplified financial calculation.

The findings are also consistent with recent financial-AI research showing that forecasting performance depends on interactions among variable selection, model architecture, optimization strategy, data quality, investor behavior, and changing market conditions.

The main contribution of this study is empirical and procedural rather than architectural. It provides a controlled comparison of minimal and feature-augmented preprocessing, untuned and optimized configurations, and LSTM and CNN models across selected U.S. technology stocks.

The findings should be interpreted within the boundaries of the original fixed-period experiment and not as evidence of statistical superiority, live-trading profitability, or dominance over external forecasting benchmarks.

Acknowledgments

The authors would like to thank the Faculty of Artificial Intelligence, Universiti Teknologi Malaysia (FAI, UTM), for the support and facilities provided throughout this study. The academic environment and technical resources at FAI, UTM helped support the development and completion of this work. This manuscript is partly based on the corresponding author's Master Project Report, titled "Optimizing Deep Learning for U.S. Technology Stock Price Forecasting through Data Preprocessing and Hyperparameter Tuning," submitted to Universiti Teknologi Malaysia [31].

Funding Support

This work was supported/funded by the Universiti Teknologi Malaysia under UTM NEXUS Translasi (PY/2025/00988).

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available on Google Drive at https://drive.google.com/drive/folders/1F71VKe468V0iVB5cvm5kt5sS25DRKoXk?usp=sharing. The original stock-price data were obtained from the Yahoo Finance historical-data pages for Apple (AAPL), https://finance.yahoo.com/quote/AAPL/history/; Amazon (AMZN), https://finance.yahoo.com/quote/AMZN/history/; Microsoft (MSFT), https://finance.yahoo.com/quote/MSFT/history/; NVIDIA (NVDA), https://finance.yahoo.com/quote/NVDA/history/; Meta Platforms (META), https://finance.yahoo.com/quote/META/history/; Tesla (TSLA), https://finance.yahoo.com/quote/TSLA/history/; and Alphabet (GOOGL), https://finance.yahoo.com/quote/GOOGL/history/. The contextual macroeconomic series were obtained from Federal Reserve Economic Data (FRED), including the CBOE Volatility Index (VIXCLS), https://fred.stlouisfed.org/series/VIXCLS; Consumer Price Index for All Urban Consumers (CPIAUCSL), https://fred.stlouisfed.org/series/CPIAUCSL; Federal Funds Effective Rate (FEDFUNDS), https://fred.stlouisfed.org/series/FEDFUNDS; Real Gross Domestic Product (GDPC1), https://fred.stlouisfed.org/series/GDPC1; and Unemployment Rate (UNRATE), https://fred.stlouisfed.org/series/UNRATE. The repository contains the raw OHLCV data, feature-engineered datasets, chronological data splits, and collected macroeconomic series. The macroeconomic series were retained as contextual data but were not included as direct inputs in the final LSTM and CNN experiments.

Author Contribution Statement

Fauzan Ghazi: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Project administration. **Syahid Anuar:** Methodology, Validation, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Saharudin Ismail:** Validation, Writing – review & editing, Supervision, Funding acquisition. **Mariam Mazlan:** Writing – review & editing.

References

[1] Rašiová, B., & Árendáš, P. (2023). Copula approach to market volatility and technology stocks dependence. *Finance Research Letters*, *52*, 103553. https://doi.org/10.1016/j.frl.2022.103553

[2] Ji, X., Wang, J., & Yan, Z. (2021). A stock price prediction method based on deep learning technology. *International Journal of Crowd Science*, *5*(1), 55–72. https://doi.org/10.1108/IJCS-05-2020-0012

[3] Zhang, X., & Pinsky, E. (2025). S&P-500 vs. Nasdaq-100 price movement prediction with LSTM for different daily periods. *Machine Learning with Applications*, *19*, 100617. https://doi.org/10.1016/j.mlwa.2024.100617

[4] Kanwal, A., Lau, M. F., Ng, S. P. H., Sim, K. Y., & Chandrasekaran, S. (2022). BiCuDNNLSTM-1dCNN—A hybrid deep learning-based predictive model for stock price prediction. *Expert Systems with Applications*, *202*, 117123. https://doi.org/10.1016/j.eswa.2022.117123

[5] Tuhin, K. H., Nobi, A., Rakib, M. H., & Lee, J. W. (2025). Long short-term memory autoencoder based network of financial indices. *Humanities and Social Sciences Communications*, *12*(1), 100. https://doi.org/10.1057/s41599-025-04412-y

[6] Bao, W., Cao, Y., Yang, Y., Che, H., Huang, J., & Wen, S. (2025). Data-driven stock forecasting models based on neural networks: A review. *Information Fusion*, *113*, 102616. https://doi.org/10.1016/j.inffus.2024.102616

[7] López, R., Sevillano, M. C., & Jareño, F. (2023). Uncertainty and US stock market dynamics. *Global Finance Journal*, *56*, 100779. https://doi.org/10.1016/j.gfj.2022.100779

[8] Toma, L. R. (2023). Exploring the effectiveness of ARIMA and GARCH models in stock price forecasting: An application in the IT industry. *Informatica Economica*, *27*(3/2023), 61–72. https://doi.org/10.24818/issn14531305/27.3.2023.05

[9] Arashi, M., & Rounaghi, M. M. (2022). Analysis of market efficiency and fractal feature of NASDAQ stock exchange: Time series modeling and forecasting of stock index using ARMA-GARCH model. *Future Business Journal*, *8*(1), 14. https://doi.org/10.1186/s43093-022-00125-9

[10] Yüksel, B. (2023). *Comparative analysis of LSTM model in predicting ETF stock prices for different sectors* [Master's thesis, Marmara University]. ProQuest Dissertations & Theses Global.

[11] Olorunnimbe, K., & Viktor, H. (2023). Deep learning in the stock market—A systematic survey of practice, backtesting, and applications. *Artificial Intelligence Review*, *56*(3), 2057–2109. https://doi.org/10.1007/s10462-022-10226-0

[12] Kumbure, M. M., Lohrmann, C., Luukka, P., & Porras, J. (2022). Machine learning techniques and data for stock market forecasting: A literature review. *Expert Systems with Applications*, *197*, 116659. https://doi.org/10.1016/j.eswa.2022.116659

[13] Akşehir, Z. D., & Kılıç, E. (2024). Analyzing the critical steps in deep learning-based stock forecasting: A literature review. *PeerJ Computer Science*, *10*, e2312. https://doi.org/10.7717/peerj-cs.2312

[14] Yilmaz, F. M., & Yildiztepe, E. (2024). Statistical evaluation of deep learning models for stock return forecasting. *Computational Economics*, *63*(1), 221–244. https://doi.org/10.1007/s10614-022-10338-3

[15] Sen, J., & Mehtab, S. (2021). Accurate stock price forecasting using robust and optimized deep learning models. In *2021 International Conference on Intelligent Technologies*, 1–9. https://doi.org/10.1109/CONIT51480.2021.9498565

[16] Giantsidi, S., & Tarantola, C. (2025). Deep learning for financial forecasting: A review of recent trends. *International Review of Economics & Finance*, *104*, 104719. https://doi.org/10.1016/j.iref.2025.104719

[17] Korade, N. B., & Zuber, M. (2022). Stock price forecasting using convolutional neural networks and optimization techniques. *International Journal of Advanced Computer Science and Applications*, *13*(11), 42. https://doi.org/10.14569/IJACSA.2022.0131142

[18] S, B., Rao, Y. S., Reddy, K. R., Ramesh, J. V. N., Muniyandy, E., Rao, M. V. A. L. N., ..., & Bala, B. K. (2025). Improving financial forecasting accuracy through swarm optimization-enhanced deep learning models. *International Journal of Advanced Computer Science and Applications*, *16*(3), 86. https://doi.org/10.14569/IJACSA.2025.0160386

[19] Terven, J., Cordova-Esparza, D.-M., Romero-González, J.-A., Ramírez-Pedraza, A., & Chávez-Urbiola, E. A. (2025). A comprehensive survey of loss functions and metrics in deep learning. *Artificial Intelligence Review*, *58*(7), 195. https://doi.org/10.1007/s10462-025-11198-7

[20] Hoque, K. E., & Aljamaan, H. (2021). Impact of hyperparameter tuning on machine learning models in stock price forecasting. *IEEE Access*, *9*, 163815–163830. https://doi.org/10.1109/ACCESS.2021.3134138

[21] You, Y. (2024). Forecasting stock price: A deep learning approach with LSTM and hyperparameter optimization. *Highlights in Science, Engineering and Technology*, *85*, 328–338. https://doi.org/10.54097/vfa8fe80

[22] Peng, Y., Albuquerque, P. H. M., Kimura, H., & Saavedra, C. A. P. B. (2021). Feature selection and deep neural networks for stock price direction forecasting using technical analysis indicators. *Machine Learning with Applications*, *5*, 100060. https://doi.org/10.1016/j.mlwa.2021.100060

[23] Vuong, P. H., Phu, L. H., van Nguyen, T. H., Duy, L. N., Bao, P. T., & Trinh, T. D. (2024). A bibliometric literature review of stock price forecasting: From statistical model to deep learning approach. *Science Progress*, *107*(1), 00368504241236557. https://doi.org/10.1177/00368504241236557

[24] Bahoo, S., Cucculelli, M., Goga, X., & Mondolo, J. (2024). Artificial intelligence in finance: A comprehensive review through bibliometric and content analysis. *SN Business & Economics*, *4*(2), 23. https://doi.org/10.1007/s43546-023-00618-x

[25] Mhlanga, D. (2024). The role of big data in financial technology toward financial inclusion. *Frontiers in Big Data*, *7*, 1184444. https://doi.org/10.3389/fdata.2024.1184444

[26] Dehghani, F., & Larijani, A. (2023). *An algorithm for predicting stock market's index based on MID algorithm and neural network*. SSRN. https://doi.org/10.2139/ssrn.4448033

[27] Pazouki, S., Jamshidi, M. B., Jalali, M., & Tafreshi, A. (2025). The integration of big data in FinTech: Review of enhancing financial services through advanced technologies. *World Journal of Advanced Research and Reviews*, *25*(1), 546–556. https://doi.org/10.30574/wjarr.2025.25.1.0060

[28] Salimipour, A., & Ebadi, M. J. (2025). Prediction European option prices using machine learning techniques integrated with multilevel Monte Carlo simulation. *Computational Algorithms and Numerical Dimensions*, *4*(4), 385–398. https://doi.org/10.22105/cand.2026.542696.1207

[29] Mojtahedi, S., Mashhadi, S., & Savin, A. (2025). MAX effect and investor sentiment: Evidence from the Swedish stock market. *Journal of Business and Management Studies*, *7*(2), 184–206. https://doi.org/10.32996/jbms.2025.7.2.13

[30] Pazouki, S., Jamshidi, M., Jalali, M., & Tafreshi, A. (2025). Transformative impact of AI and digital technologies on the FinTech industry: A comprehensive review. *International Journal of Advanced Research in Humanities and Law*, *2*(2), 1–27. https://doi.org/10.63053/ijrel.37

[31] Ghazi, F. (2025). *Optimizing deep learning for U.S. technology stock price forecasting through data preprocessing and hyperparameter tuning* [Unpublished master's thesis]. Universiti Teknologi Malaysia.

How to Cite: Ghazi, F., Anuar, S., Ismail, S., & Mazlan, M. (2026). Optimizing Deep Learning for U.S. Technology Stock Price Forecasting Through Data Preprocessing and Hyperparameter Tuning. *Journal of Data Science and Intelligent Systems*. https://doi.org/10.47852/bonviewJDSIS620210864