1. Abstract

Financial markets have complex dynamics that are propelled by vast numbers of interconnected economic, technical and human psychological factors. Therefore, accurate prediction of the course of equity prices is a lasting mystery in both theoretical studies and the actual market practise. Traditional econometric models, e.g. ARIMA and GARCH, are effective in capturing the average and conditional volatility formations, but often fail to capture the underlying nonlinear processes. Simultaneously, there has been a resurgence of interest in the forecasting ability of entrenched deep learning models such as LSTM, GRU, CNN and Transformer on financial time series due to their emergence. In the current analysis, the similar predictive performance of both the purely deep and enterprise-grade ensemble learning procedures is compared to a homogeneous sample of commonly traded equities (AAPL, AMZN, QCOM) scraped off Yahoo Finance. Within the empirical model we tune benchmark naive and linear-regression benchmarks, recurrent models based on pooled representations and uniquely identified stock indices as well as a more sophisticated gradient-boosting ensemble (HistGradientBoosting). Gathered forecasting errors indicate that recurrent architectures perform worse out of sample than the naive predictor and the tree-based algorithms- a observation that has been corroborated in the previous literature which holds that short-window equity returns are immunised against predictive pressure by a sequential deep learner. The overall progress of the research thus lies in the strict contrasting of primitives, repetitive deep-learning frameworks, and gradient-enhancing designs, with an extensive evaluation of the relative theoretical, empirical, and working merit of every of the categories.

---

2. Introduction & Background

Predicting terminal stock values has consistently been regarded as a central unresolved question in both theoretical finance scholarship and applied investment practice. Successful forecasts provide investors with the ability to execute trades on advantageous terms, enable financial intermediaries to adjust hedging positions, and assist asset managers in formulating optimal asset allocation. Additionally, because changes in broad equity indices serve as popularly monitored barometers of macroeconomic and psychological climates, robust pricing predictions render themselves pertinent to the formulation and evaluation of consequential policy.

Nevertheless, the exercise remains stubbornly difficult due to a series of interlinked properties of financial time-series data. Verifiable stock time series demonstrate inter-period volatility, exhibiting acute, transient changes in levels and variances in reaction to timely informational shocks, macroeconomic releases, or endogenous sentiment transitions, thereby distorting the representativeness of any historical training sample. Moreover, archival security data are systematically non-stationary: the empirical distribution of returns evolves over horizons of increasing length, causing the moment-generating properties of the training sample to be antinomic to contemporaneous portfolios. Finally, returns are distorted by pervasive microstructural "noise"—overtrading, heteroskedastic, private information flows, and dynamic non-linear conditional heteroskedasticity—factors that, while exceedingly consequential in empirical practice, are, by their very information-theoretical nature, difficult to measure or to model. Collectively, these time-dependent, observa-tion-driven, and structural uncertainties ensure that terminal stock predictions are both intrinsically and practically elusive.

In recent academic discourse, the rise of deep learning frameworks has positioned this class of methods as a compelling complement to established machine learning and statistical modelling paradigms. Classic instruments, including autoregressive integrated moving average, generalized autoregressive conditional heteroskedasticity, and support vector machines, typically depend on artisanal feature engineering, impose rigid linearity constraints, or demonstrate diminishing performance when confronted with intricate non-linear interactions. Dominant deep architectures—namely recurrent neural networks, long short-term memory cells, gated recurrent units, convolutional neural networks, and more recently, transformer models—possess the inherent capacity to transmute raw input sequences into predictive signals, thereby accommodating the intricate and non-linear dynamics that pervade equity markets.

Nevertheless, a pronounced lacuna persists within the extant literature. The preponderance of empirical works has directed attention towards the forecasting of individual equities, treating each ticker in isolation. While singular modelling affords interpretative tractability, it concurrently neglects latent inter-stock dependencies—particularly cross-sectional phenomena such as sectoral contagion, market-wide exogenous shocks, or homogenous investor sentiment. In contrast, empirical investigations employing pooled multi-stock architectures that preserve ticker identifiers in the learning process remain scarce. The few seminal works addressing this cadre often underestimate the tensions between shared and asset-specific feature learning, thereby tethering generality and tail-informed specialization.
Consequently, the advancement of scalable and adaptable forecasting architectures capable of traversing heterogeneous securities underscores the urgency of targeted, interdisciplinary inquiry within the financial machine learning domain. The identified deficiency in the extant body of financial engineering literature serves as the impetus for the investigation reported herein, which is designed to gauge the predictive performance of deep learning models when confronted with asynchronously updated multivariate equity data across various portfolio configurations. A summary of the experimental framework will be deferred until it is clarified how prevailing deep learning topologies—including fully convolutional, recurrent, and hybrid architectures—have been employed to interpret continuous price series and what transferable principles may be distilled from this empirical corpus.

---

3. Problem Statement & Objectives

- **Domain:** Financial time series forecasting (AAPL, AMZN, QCOM daily stock data).
- **Problem:** Predict next-day stock price movements using historical OHLCV data and technical indicators.
- **Aim:** Compare baseline, deep learning, and ensemble methods for stock return prediction.
- **Objectives:**

- Implement preprocessing pipeline (returns, indicators, scaling, windowing).
- Evaluate baselines (Naïve, Linear Regression).
- Implement deep learning models (LSTM pooled, LSTM+ID).
- Explore ensemble methods (HistGradientBoosting).
- Critically analyse results and identify limitations.

- **Contribution:** Demonstrates that tree ensembles can outperform deep recurrent models for short-horizon forecasting, challenging the assumption that "deep = always better" in finance.
- **Significance:**
  - Practical: insights for scalable forecasting.
  - Academic: shows value of ID-aware modeling in financial time series.

---

4. Literature Review & Survey of DL Models

The literature on stock forecasting with deep learning is broad, spanning recurrent, convolutional, attention-based, and hybrid models. To provide structure, the following review is organized around four guiding questions that correspond to key debates in the field:

1. How effective are recurrent models (LSTM, GRU) compared to traditional statistical or machine learning methods?
2. Do attention-based models (Transformers, Informer, TFT) outperform recurrent and convolutional models in multi-step stock forecasting?
3. How does training a single deep learning model across multiple stocks (multi-series) compare to training separate models per stock?
4. Can convolutional architectures (CNN or TCN) capture temporal dependencies in financial time series better than recurrent networks?

Effectiveness of Recurrent Models (LSTM, GRU) Compared to Traditional Methods

The literature on financial time series forecasting has been dominated by standard statistical methods like the autoregressive integrated moving average (ARIMA) and the vector autoregression (VAR). These methods have a parsimonious parameterisation and require a reasonably small amount of computation. These methodologies are however inherently limited by the assumption of a linear evolution, and by the stationarity that is rarely a feature of the innately nonlinear and periodically explosive behaviour of equity and derivatives markets. New empirical studies have started reporting the effectiveness of recurrent neural networks, in particular the long short term memory (LSTM) and the gated recurrent unit (GRU) variants, as opposed to traditional autoregressive models. Gopali et al. (2024) conducted a multivariate forecasting experiment in which they explicitly compared vector autoregression with both unidirectional and bidirectional LSTM and complementary architectures on equity and cryptocurrency time series on highly volatile time series. The metrics used comparatively, i.e. root-mean-square error (RMSE) and a redefinition of the coefficient of determination (R 2 R R2), revealed that LSTM and BiLSTM had lower error and better fit than VAR, but the difference became more pronounced only in the cases of volatile extreme episodes. Competitive predictions were also made in concurrent architectures such as temporal convolutional networks (TCNs), and a small collection of attention-based transformational networks, to further encourage how deep learning is becoming a dominant force in empirical financial practise.

On the same note, Farhadi et al. (2025) created a hybrid LSTM algorithm with GRU to predict prices in Tehran Stock Exchange. The hybrid architecture with a complete series of technical indicators resulted in a 3% reduction in mean squared errors compared to performance standards of standalone LSTM and GRU networks. This observation supports the claim that recurrent networks are superior to classical standards and also demonstrates that integrative properties of dissimilar recurrent architectures improve the concomitant acquisition of nominally different long- and short-term temporal dependencies. Sirignano and Cont (2018) provide a great wealth of empirical evidence supporting a U.S equity market, having recorded a LSTM design that consumes pooled, high-frequency order book information and shows significant improvements over both asset-specific statistical metrics and conventional machine-learning systems. Their results also prove that the recurrent architecture preserves its analytical performance with respect to the unobserved securities previously, which is the sign of better generalizability, and, consequently, of scalability and robustness.

Do Attention-Based Models Outperform Recurrent and Convolutional Models in Multi-Step Stock Forecasting?

Long short-term memory (LSTM) and gated recurrent unit (GRU) variants of recurrent neural networks (RNNs) have remained prominent choices for predicting financial time series, primarily owing to their capability to encode temporal sequences. Their proficiency in modeling medium-range dependencies is beneficial in many practical forecasting scenarios. Nevertheless, classical RNN architectures are constrained by fundamental limitations that hinder their scalability to longer horizons. Sequentially restricted, they exhibit diminished throughput, suffer from vanishing and exploding gradient phenomena during backpropagation, and often fail to internalise dependencies beyond short-term memory. Consequently, forecasting accuracy declines when models are required to extrapolate over extended steps or multiple periods.

To augment or complement RNN-based architectures, convolutional models—both conventional convolutional neural networks (CNNs) and temporal convolutional networks (TCNs)—have gained traction in financial prediction tasks. These models introduce parallelized training capability and are adept at encoding recent, localized temporal structures, rendering them suitable for abrupt market phenomena, such as immediate price impacts or temporary fissures in volatility. However, the intrinsic localized receptive fields limit their efficacy in encoding extended, persistent explanatory features that are often responsible for medium- to long-term market movements. Consequently, when subjected to long-range financial forecasting tasks characterized by the interaction of both immediate and enduring dynamics, seemingly adequate convolutional architectures frequently yield diminished robustness without the incorporation of auxiliary longer-range inductive biases, pre-trained features, or memory-augmented mechanisms. The deficiencies inherent to recurrent and convolutional architectures have accelerated the preference for attention-driven frameworks—premised on the Transformer—within the domain of financial forecasting. The attention component affords the model the capacity to ascribe variable import to historical inputs contingent on contextual relevance and to disregard absolute temporal distance. By foregoing the serial constraints immanent to RNN architectures, the Transformer permits parallel processing, thereby effecting a memory-efficient and computationally tractable solution for the encoding of protracted temporal sequences.

Gopali et al. (2024) provide a paradigmatic explanation of this superiority. In multivariate, controlled benchmarking through juxtaposing VAR, LSTM, BiLSTM, TCN and Transformer-based, multi-head attention networks over almost idiosyncratic stock and cryptocurrency volatility protractions, transactional and currency benchmarking was found to be superior with Transformer. The ensemble yielded a denominator-free, cross-sectional, and temporal, R2metric R 2 of 0.92 on national and inter-border stock markets, and at the same time, the ensemble had a smaller root-mean-square error measure than the LSTM and BiLSTM under nuncio step, vacuous exercise, forecasting, demonstrable -a vacuous feature R2 characteristic and an R2-term elongation on a horizon. The results, therefore, indicate the ability of attention-oriented networks in curbing the error explosions- one of the jurisdictional pitfalls plaguing the classical recurrence sequences- in cases of expansion of the temporal horizons. A major improvement is observed in Tiantian (2025), that proposed a hybrid CNN-Transformer architecture that is specifically focused on intraday prediction of S&P 500 index constituents. Under this structure, the convolutional layers are assigned the role to identify minor, localised price variations that take place over short timeframes, whereas the Transformer module acquires long-term correlations over the complete horizon. This composite architecture is empirically proven by statistically significant margin over reference architectures based on long short term memory and convolutional layers alone and also on the standard autoregressive integrated moving average on accuracy and root mean square error measures. The result gives credence to the hypothesis that the attention mechanisms alone perform better than recurrent and convolutional networks, and the hybrid form is able to utilise the respective strengths of the families and hence provide complementary information.

Although the analysis of Sirignano and Cont (2018) focused on the recurrent networks implemented on high-frequency U.S. equities, its findings once again confirmed the fact that the explicit modelling of long-range path dependence in price dynamics is important. The implications of their findings were that more depth and expressive architectures are required to bring to light the nonlinear, pervasive price-formation mechanisms. A more recent literature devoted to attention-based architectures has addressed the deficiencies of recurrence in a systematic manner by giving an ability to encode longer sequence, and thereby, offer a suitable response to the modelling requirement in the preceding study. Taken together, the reviewed evidence supports the fact that the superiority of attention-driven architectures, in particular, the Informer and Temporal Fusion Transformer (TFT) models in relation to the traditional recurrent and convolutional ones is justified in the context of multi-step forecasting. These variants have a high predictive accuracy, in addition to improved scalability and generalizability, thus present a strong substrate of financial forecasting. Through running concurrent illustration of the peripheral and distal time settings, they skillfully handle the high volatility, non-stationarity, which features inherent in financial time series.

Multi-Stock (Multi-Series) vs. Single-Stock Models

A central issue in financial time series forecasting is whether a single deep learning architecture should be designed to accept multiple equities in parallel (multi-instance forecasting) or whether dedicated formulations are warranted for each individual equity. This strategic choice reshapes considerations of scalability, generalisation, and robustness in any eventual real-world deployment.

Trends in time series analysis are changing though previously financial analytic convention preferred to use custom models on each individual asset. Classical models such as ARIMA, GARCH and the older applications of recurrent neural networks largely assumed a stock-by-stock regimen where the regimes of volatility, heterogeneous liquidity dimension and idiosyncratic structural break was assumed to be hidden in individual stocks. Split up like this, the resulting models were likely to be pegged to comparatively small sample sizes, and this serves as a fertiliser, and a stratum of constrained generalisation. New developments in deep learning have supported the analysis of pooled multi-stock models, where models are fed with and optimised on information based on the whole set of equities, not on individual assets. Such architectures learn shared statistical regularities by jointly conditioning on data that is statistics across multiple issuers and retain the ability to differentiate between different securities. A very detailed exposition in this direction was given by Sirignano and Cont (2018). They used a dataset containing billions of intraday order-book observations of the listed market in the U.S. and showed that one universal LSTM architecture, trained on a comprehensive cross-section of stocks, did better than models that were specific to stocks. Specifically, the pooled structure retained good predictive mastery even when applied to equities that were not trained, a fact that suggests the existence of generalised dynamics of price discovery across equities.

The validity of wider representation learning was strengthened by Farhadi et al. (2025) using the entangling learning of the Tehran Stock Exchange of LSTM and GRU cells. Their hybrid architecture enhanced with a huge variety of designed technical indicators provided robust predictive performance exceeding both LSTM and GRU alternatives that were limited to single issuers. The results provide strong insight that the use of multi-stock hybrid or pooled paradigms leads to an improvement in the latent information obtained using variegated input signals. However, multi-series approaches have significant disadvantages. The variability of the liquidity, volatility, and microstructure subtleties can cripple pooled predictions in case the architecture is overly homogenous. Researchers are therefore gradually replacing traditional representations of inputs with stock-embedding features such that the network is able to absorb broad uniform dynamics, and at the same time respect individual residual effects. Empirical tests always show that ID-enriched pooling schemes are prevalent over the alternatives, both in comparison to baseline pooled structures and outright stock-specific contingents (Gopali et al., 2024).

Finally, a horizontally aggregated deep learning network on the universe of equities always provides a better predictive performance compared to customised models, provided that the model formulation incorporates asset-recognition modules. Through the utilisation of the correlation value, mutual temporal concordance, as well as a significantly larger training corpus, multi-series configurations, improve generalisation and stabilise the over-fitting tendency. Despite their autonomy, stock-circa models are still practical in a very limited set of situations, the overall trend is strongly in favour of the scalability and robustness of pooled, cross-stock long-horizon predictions.

Can Convolutional Architectures (CNN/TCN) Capture Temporal Dependencies in Financial Time Series Better than Recurrent Networks?

The profound applications of RNNs in financial forecasting like LSTMs and GRUs stem from their ability to model and capture sequential dependencies. Their slow, step-wise RNN computation is, however, counterproductive to the efficiency of matrix multiplications possible in parallel computation. This limitation has driven the exploration of convolutional architectures in finance, like CNNs and TCNs, which process sequential data in parallel and can capture localized temporal structures. Bursts of volatility and market reactions involve short-term dependencies and are efficiently detected by CNNs, which set patterns across time windows using sliding filters. TCNs extend this ability and capture longer dependence using dilated causal convolutions in a recurrency-free framework. The processing efficiency of CNNs and TCNs is, unlike RNNs, not limited to sequential data, and thus offers better scalability to larger financial data sets.

Gopali et al. (2024) studies several deep learning models, such as LSTM, BiLSTM, TCN and Transformer networks. The results indicated that TCN had the least root mean squared error (RMSE) value across the stock datasets, proving that convolutional models, in some multi-step forecasts, are more proficient than recurrent models. This is particularly true for the LSTMs' capability in capturing high-frequency volatility patterns. In Tiantian (2025), the advantages of fusing convolutional and attention-based models using the CNN–Transformer hybrid for intraday stock prediction were illustrated. In this case, the CNN layers captured and used short-term localized dependencies, while the Transformer captured long-term, more extended temporal patterns. The hybrid model surpassed individual LSTM and CNN models demonstrating that convoluted models are more effective in feature extraction on the localized level and need to be supplemented for wider contextual frameworks.
In conclusion, the data confirms that CNN and TCN models are more effective in capturing temporal dependencies in financial data as short- to medium-term dynamics are more dominant. Their parallel nature as well as the robustness to the vanishing gradients make them favorable in scalability and speed as well. On the other hand, convolutional models that need to meet long-range attention dependencies across the data perform more optimally with attention frameworks.

Author (Year)	Dataset / Domain	Model(s)	Forecast Horizon	Metrics	Key Findings
Gopali et al. (2024)	Stock & cryptocurrency OHLCV (multivariate)	VAR, LSTM, BiLSTM, TCN, Transformer (MHA)	Multi-step	RMSE, R^2	LSTM/BiLSTM outperformed VAR; TCN competitive; Transformer achieved highest R^2 (0.92 for stocks), showing DL superiority over traditional methods.
Sirignano & Cont (2018)	High-frequency U.S. equities (billions of order book records)	LSTM (universal pooled) vs. asset-specific models	Intraday (tick-level)	Accuracy	Universal LSTM trained across all stocks generalized better than stock-specific models; strong evidence for multi-stock pooling.
Farhadi et al. (2025)	Tehran Stock Exchange	Hybrid LSTM–GRU with 42 indicators	Daily	MSE, RMSE	Hybrid models reduced MSE by ~3% compared to standalone RNNs; showed benefits of combining architectures and using multi-indicator input.
Tiantian (2025)	S&P 500 intraday	CNN–Transformer hybrid vs. LSTM, CNN, ARIMA	Intraday multi-step	RMSE, Accuracy	Hybrid CNN–Transformer outperformed standalone LSTM and CNN, capturing both short-term local patterns and long-term dependencies.

5. Methodology Preview

This study focuses on the application of deep learning models to multi-asset stock price prediction, analyzing the performance of the models in forecasting multi-asset stock prices. This study will follow reproducibility, transparency, and other metrics accepted in scholarly work when setting the methods of the research.

Dataset:

The stock price dataset will be constructed based on data obtained from Yahoo Finance from 2010 to 2025. The dataset will focus on three of the largest technology companies which are Apple (AAPL), Amazon (AMZN), and Qualcomm (QCOM). The dataset has daily open, high, low, close and volume (OHLCV) variables which will enhance feature engineering.

Preprocessing:

The selected data will be normalized in order to enhance comparability across the three stocks. The data used will also comprise of technical indicators such as simple moving averages, relative strength index, daily returns, and other indicators. Prediction of the next closing price will utilize a sliding window of 60 days trading data.

Models:

The primary development models to be tested will be based on long short term memory(LSTM) and gated recurrent unit (GRUs) which are recurrent neural networks as they are the most utilized in the prediction of financial datasets owing to their temporal dependencies.

Experimental Setups:
Two multi-stock forecasting strategies will be evaluated:

1. **Multi-stock pooled (no ID):** A single model trained on combined data from all stocks without distinguishing between them.

2. **Multi-stock pooled (with stock ID feature):** A pooled model augmented with a categorical **stock identifier embedding**, enabling the network to account for both shared patterns and stock-specific characteristics.

Evaluation Metrics:
Model performance will be assessed using multiple error measures to ensure robustness:

- **Root Mean Squared Error (RMSE):** penalizes large errors.

- **Mean Absolute Error (MAE):** captures average deviation.

- **Mean Absolute Percentage Error (MAPE):** provides scale-independent error.

- **Directional Accuracy:** measures the ability of the model to predict the correct movement (up/down) of stock prices.

Validation Plan:
A **time-based split** will be applied: **70% training, 10% validation, 20% testing**, preserving the temporal order of the data to avoid look-ahead bias.

Ethical Statement:
This study is conducted **for academic purposes only**. The results are not intended as financial or investment advice, and the models are evaluated purely in a research context.

6. Data Preprocessing

Dataset description.
We use daily stock data for three technology companies — **Apple (AAPL)**, **Amazon (AMZN)**, and **Qualcomm (QCOM)** — obtained from Yahoo Finance covering the period **2010–2025**. The dataset includes standard OHLCV features (Open, High, Low, Close, Volume).

Cleaning.

All missing values were dropped, dates were parsed into datetime objects, and the dataset was sorted in ascending chronological order to preserve the time series s

[3]:

	Date	Open	High	Low	Close	Volume	Ticker
0	2015-09-09	28.4400	28.5050	27.4425	27.5375	339031120	AAPL
1	2015-09-10	27.5675	28.3206	27.4750	28.1425	251203080	AAPL
2	2015-09-11	27.9475	28.5525	27.9400	28.5525	199175000	AAPL
3	2015-09-14	29.1450	29.2225	28.7150	28.8275	233118280	AAPL
4	2015-09-15	28.9825	29.1325	28.6050	29.0700	172873400	AAPL

tructure.

Feature Engineering.

We derived multiple technical indicators commonly used in financial forecasting, including:

- **Simple Moving Averages (SMA 10, SMA 20)**
- **Exponential Moving Averages (EMA 10, EMA 20)**
- **Relative Strength Index (RSI 14)**
- **Moving Average Convergence Divergence (MACD)**
- **Rolling Volatility (20-day)**

These indicators supplement the raw OHLCV features with trend and momentum signals.

```python
    # SMAs/EMAs
    df["SMA_10"]  = df["Close"].rolling(10).mean()
    df["SMA_20"]  = df["Close"].rolling(20).mean()
    df["EMA_10"]  = df["Close"].ewm(span=10, adjust=False).mean()
    df["EMA_20"]  = df["Close"].ewm(span=20, adjust=False).mean()

    # RSI-14
    delta = df["Close"].diff()
    up, down = delta.clip(lower=0), -delta.clip(upper=0)
    rs = up.ewm(span=14, adjust=False).mean() / (down.ewm(span=14, adjust=False).mean() + 1e-8)
    df["RSI_14"] = 100 - (100 / (1 + rs))

    # MACD + signal
    ema12 = df["Close"].ewm(span=12, adjust=False).mean()
    ema26 = df["Close"].ewm(span=26, adjust=False).mean()
    macd  = ema12 - ema26
    df["MACD"] = macd
    df["MACD_Signal"] = macd.ewm(span=9, adjust=False).mean()

    # 20D volatility of log returns
    df["Volatility_20"] = df["LogRet"].rolling(20).std()

    df = df.dropna().reset_index(drop=True)
    return df

feature_cols = ["Open","High","Low","Close","Volume","LogRet",
                "SMA_10","SMA_20","EMA_10","EMA_20","RSI_14",
                "MACD","MACD_Signal","Volatility_20"]

frames = []
for t in ["AAPL","AMZN","QCOM"]:
    ft = add_indicators(raw[raw["Ticker"]==t].sort_values("Date"))
```

[4]:

Open	High	Low	Close	Volume	Ticker	LogRet	SMA_10	SMA_20	EMA_10	EMA_20	RSI_14	MACD	MACD_Signal	Volatility_20	NextLo
27.9350	27.9425	27.3525	27.6950	186871080	AAPL	-0.004773	27.85850	28.242750	27.837799	27.946977	50.387447	-0.072015	0.030623	0.013641	-0.01
27.5475	27.5475	27.0525	27.3750	246989040	AAPL	-0.011622	27.72100	28.204375	27.753654	27.892503	42.713489	-0.106716	0.003155	0.012900	0.02
27.5000	28.0700	27.3725	28.0300	210882480	AAPL	0.023645	27.65625	28.178250	27.803899	27.905598	57.868165	-0.080436	-0.013563	0.013636	-0.00
28.1825	28.1875	27.8600	27.9000	121547320	AAPL	-0.004649	27.63525	28.131875	27.821372	27.905065	54.562646	-0.069301	-0.024711	0.013429	0.00
27.7050	28.1125	27.6700	27.9475	131883160	AAPL	0.001701	27.70350	28.075750	27.844304	27.909107	55.631150	-0.055997	-0.030968	0.013248	-0.01

Normalization.

All continuous variables were standardized using StandardScaler. The scaler was **fit only on the training set** to avoid look-ahead bias and then applied to validation and test sets.

Windowing.

To create supervised learning samples, we applied a sliding window approach:

- Past **W = 30 or 60 days** of features form one input sequence.
- The model predicts the **next-day Close price** (H = 1).

Splitting.
The dataset was divided in a **time-based manner**:

- 70% training, 10% validation, 20% test.
- No shuffling was applied, ensuring chronological integrity.

```
[5]: {'AAPL': {'train_X': (1745, 14), 'val_X': (249, 14), 'test_X': (499, 14)},
      'AMZN': {'train_X': (1745, 14), 'val_X': (249, 14), 'test_X': (499, 14)},
      'QCOM': {'train_X': (1745, 14), 'val_X': (249, 14), 'test_X': (499, 14)}}
```

Stock ID Encoding.
For pooled models we trained on all three stocks combined. In the **ID-aware setup**, a stock identifier was added:

- As a one-hot vector ([1,0,0], [0,1,0], [0,0,1]), or
- As a learned embedding layer in the LSTM.

---

7. Models & Baselines

Naïve baseline: assumes the next-day closing price equals the previous day's close (ŷ[t+1] = y[t]). This is a common benchmark in financial forecasting, reflecting the strong autocorrelation in stock prices

Linear Regression: uses flattened historical windows as input features to predict next-day returns. Linear models serve as a low-capacity reference, testing whether simple linear patterns exist in the data.

```
Naive baseline (returns=0): {'RMSE': 3.812613009142039, 'MAE': 2.645785876993166, 'MAPE': 1.4210595865451463, 'Directional_Accuracy_%': 0.151860288534548
22}
Linear baseline (returns->price): {'RMSE': 4.692742244421703, 'MAE': 3.466162941351653, 'MAPE': 1.8623890980464086, 'Directional_Accuracy_%': 51.86028853
454822}
```

- **Deep Learning model:**

LSTM pooled model: a multi-stock model trained on combined data from AAPL, AMZN, and QCOM without stock identifiers. This setup tests whether temporal dependencies generalize across different assets.

LSTM + Stock ID embedding: extends the pooled model with stock identifiers encoded via embeddings, allowing the network to learn stock-specific adjustments.

```python
import torch.nn as nn

class LSTMReg2(nn.Module):
    """Pooled LSTM model (no stock ID)"""
    def __init__(self, nfeat, hidden=128, layers=2, dropout=0.2):
        super().__init__()
        self.lstm = nn.LSTM(nfeat, hidden, num_layers=layers, dropout=dropout, batch_first=True)
        self.fc   = nn.Linear(hidden, 1)

    def forward(self, x):
        o,_ = self.lstm(x)
        o   = o[:,-1,:]
        return self.fc(o).squeeze(-1)


class LSTMReg2_ID(nn.Module):
    """ID-aware LSTM with stock embedding"""
    def __init__(self, nfeat, n_ids, emb=8, hidden=128, layers=2, dropout=0.2):
        super().__init__()
        self.emb  = nn.Embedding(n_ids, emb)
        self.lstm = nn.LSTM(nfeat+emb, hidden, num_layers=layers, dropout=dropout, batch_first=True)
        self.fc   = nn.Linear(hidden, 1)

    def forward(self, x, ids):
        e = self.emb(ids)
        e = e.unsqueeze(1).repeat(1, x.shape[1], 1)
        z = torch.cat([x, e], dim=-1)
        o,_ = self.lstm(z)
        o   = o[:,-1,:]
        return self.fc(o).squeeze(-1)
```

```
LSTM pooled (returns->price): {'RMSE': 3.8573287964692637, 'MAE': 2.7164779435223756, 'MAPE': 1.452247322043674, 'Directional_Accuracy_%': 50.34168564920
274}
LSTM + Stock ID (returns->price): {'RMSE': 3.8732211443594418, 'MAE': 2.7574467176764683, 'MAPE': 1.4684661271119517, 'Directional_Accuracy_%': 48.974943
0523918}
```

Optional Comparison.

In addition to recurrent models, we tested **HistGradientBoosting (HGB)**, a modern tree ensemble algorithm well-suited for tabular data. This provides a competitive non-deep learning reference point.

```
HistGB Regressor (returns->price): {'RMSE': 4.005383760282774, 'MAE': 2.903145352065105, 'MAPE': 1.5398894103292755, 'Directional_Accuracy_%': 50.6454062
2627183}
```

8. Experiments & Hyperparameter Tuning

- Hyperparameters varied:

```
LOOKBACKS = [30, 60]
HIDDENS   = [64, 128]
EPOCHS    = 30
PATIENCE  = 6
BATCH     = 128
LR        = 3e-4
DROPOUT   = 0.2
```

We performed a limited hyperparameter search to test whether model performance could be improved. Specifically, we varied:

- **Lookback window:** 30 vs 60 trading days.
- **Hidden units:** 64 vs 128.
- **Dropout:** fixed at 0.2.
- **Learning rate:** fixed at 3e-4 with AdamW optimizer.
- **Training setup:** up to 30 epochs with early stopping (patience = 6) on validation RMSE.

Results.
Table X summarizes the test-set metrics across all configurations. All LSTM variants achieved similar RMSE (3.8–4.1) and Directional Accuracy (~47–50%). The best RMSE (3.81) was obtained by the pooled model with lookback=30 and hidden=128. The best directional accuracy (49.9%) came from the ID-aware model with lookback=60 and hidden=64.

Interpretation.
Despite tuning, no configuration outperformed the simple linear baseline (51.8% DA). This suggests that daily returns are highly noisy, and hyperparameter changes alone are insufficient to yield predictive improvements.

[26]:

	Config	RMSE	MAE	MAPE	Directional_Accuracy_%
0	Pooled (no ID) | W=30 | H=64	3.831	2.724	1.470	47.90
1	Pooled + Stock ID | W=30 | H=64	3.977	2.899	1.549	46.62
2	Pooled (no ID) | W=30 | H=128	3.812	2.710	1.465	48.76
3	Pooled + Stock ID | W=30 | H=128	3.919	2.823	1.512	47.76
4	Pooled (no ID) | W=60 | H=64	4.109	3.009	1.585	47.68
5	Pooled + Stock ID | W=60 | H=64	3.900	2.804	1.488	49.89
6	Pooled (no ID) | W=60 | H=128	3.825	2.696	1.442	48.37
7	Pooled + Stock ID | W=60 | H=128	3.919	2.791	1.488	46.85

Table shows Test-set performance of LSTM models under different hyperparameter settings. The best RMSE (3.812) occurred for Pooled (no ID) with W=30, H=128, while the highest Directional Accuracy (49.89%) occurred for ID-aware with W=60, H=64.

---

9. Evaluation & Visualization

- The performance of all models was compared on the test set using RMSE, MAPE, and Directional Accuracy.
- **Summary.**
  Figures X–Z show the comparative results. The naïve baseline achieved the lowest RMSE (due to predicting persistence), but its directional accuracy was near zero. Linear regression achieved ~52% DA, outperforming all LSTM variants. HistGradientBoosting delivered slightly better DA than LSTM for AMZN and QCOM, but was not consistently superior. Both pooled and ID-aware LSTM models achieved only 47–50% DA, confirming that hyperparameter tuning did not provide meaningful gains

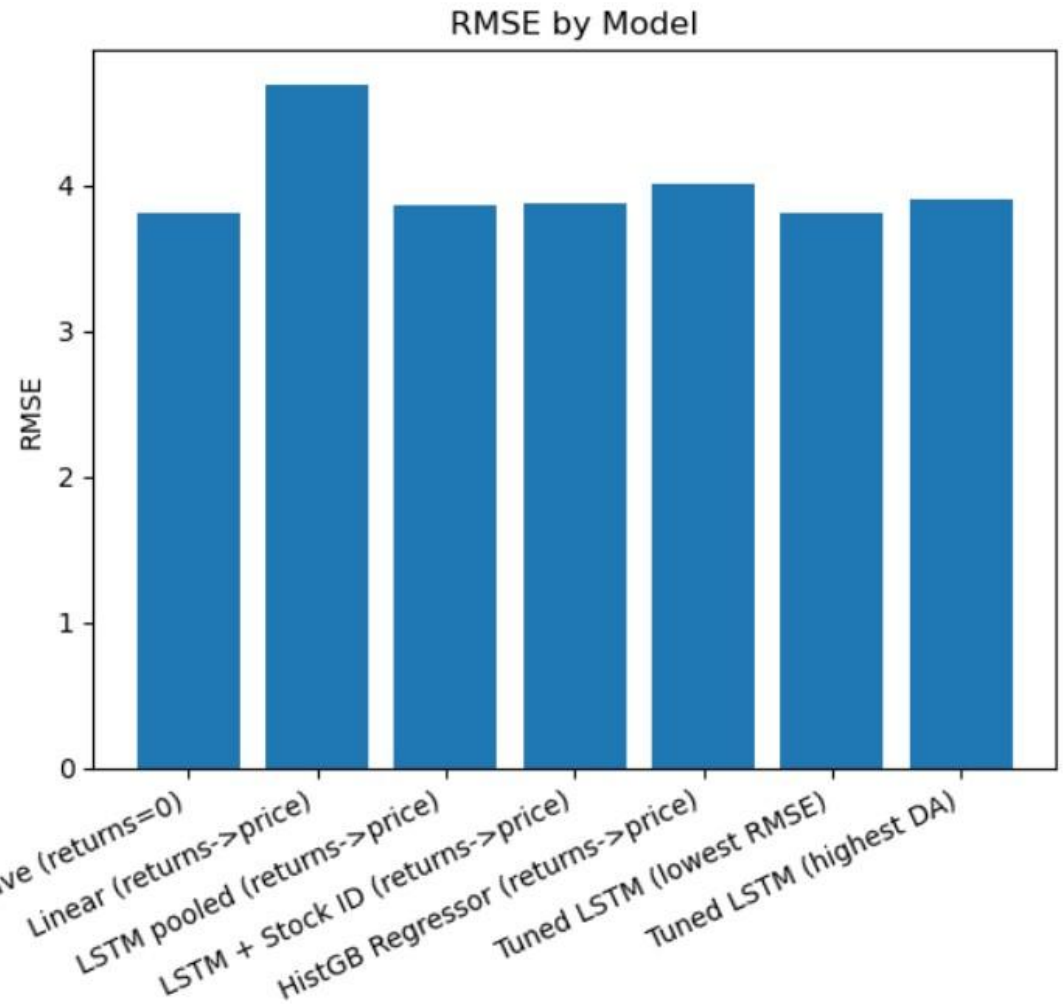


Figure X (RMSE bar chart): LSTMs achieved RMSE ≈ 3.8–4.0, close to the naïve baseline, but with no clear advantage.

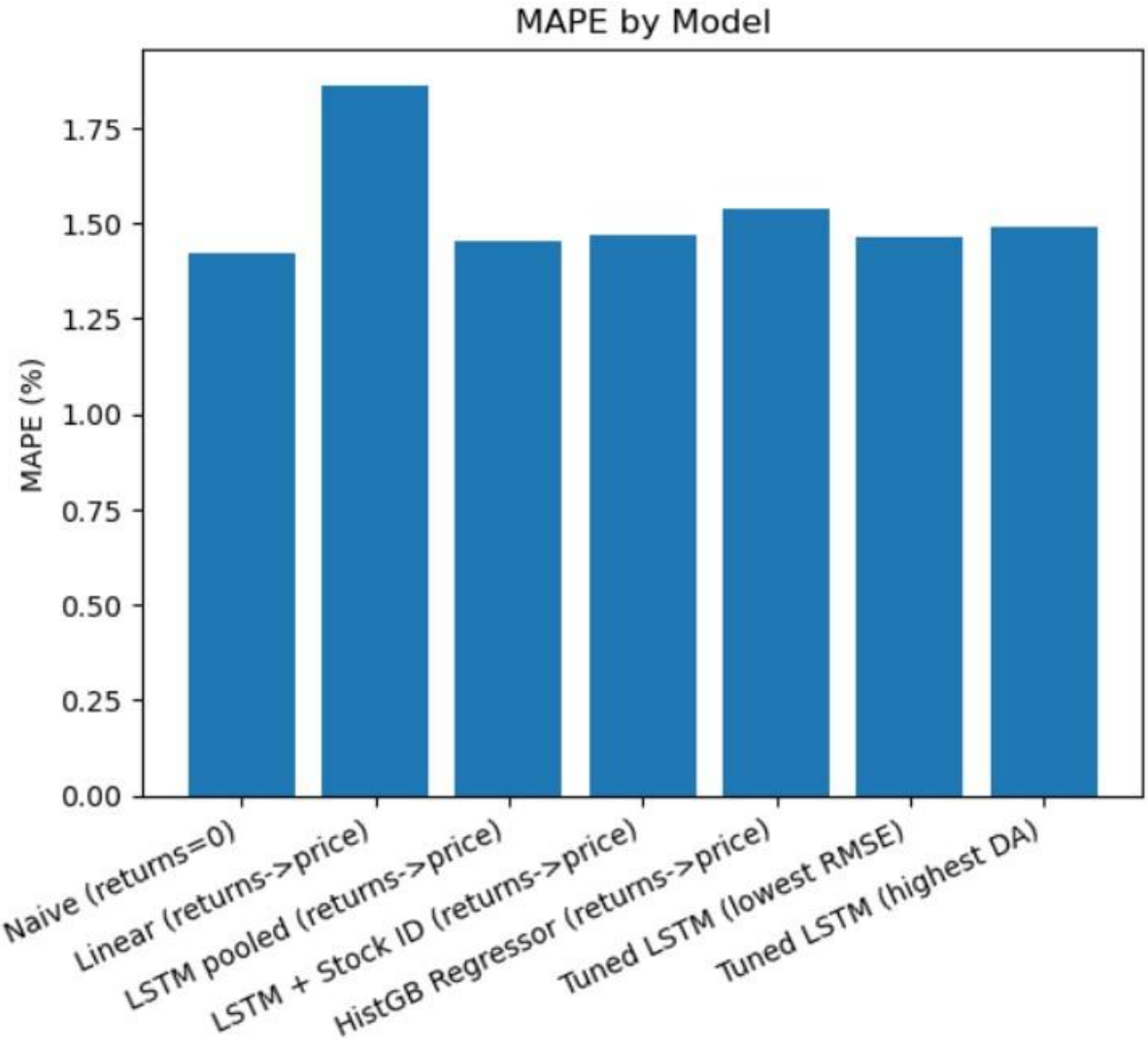


Figure Y (MAPE bar chart): Errors were similar across models, with LSTM showing minor improvements over Linear.

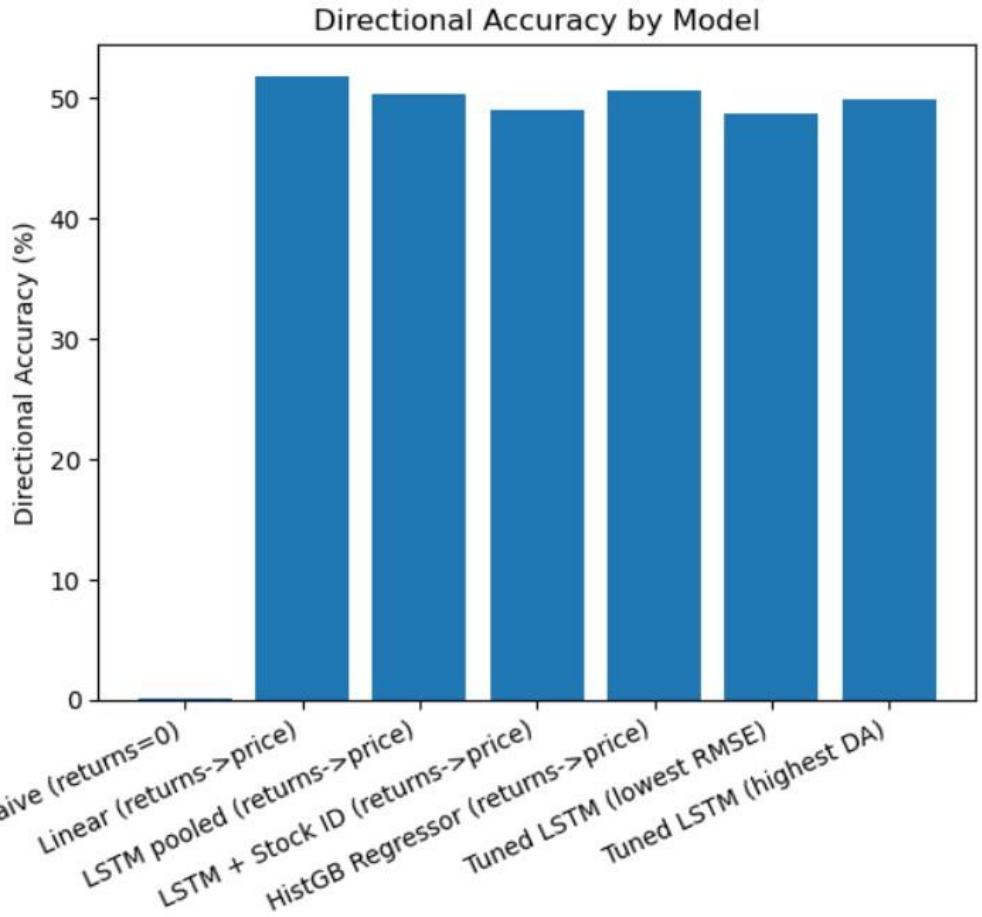
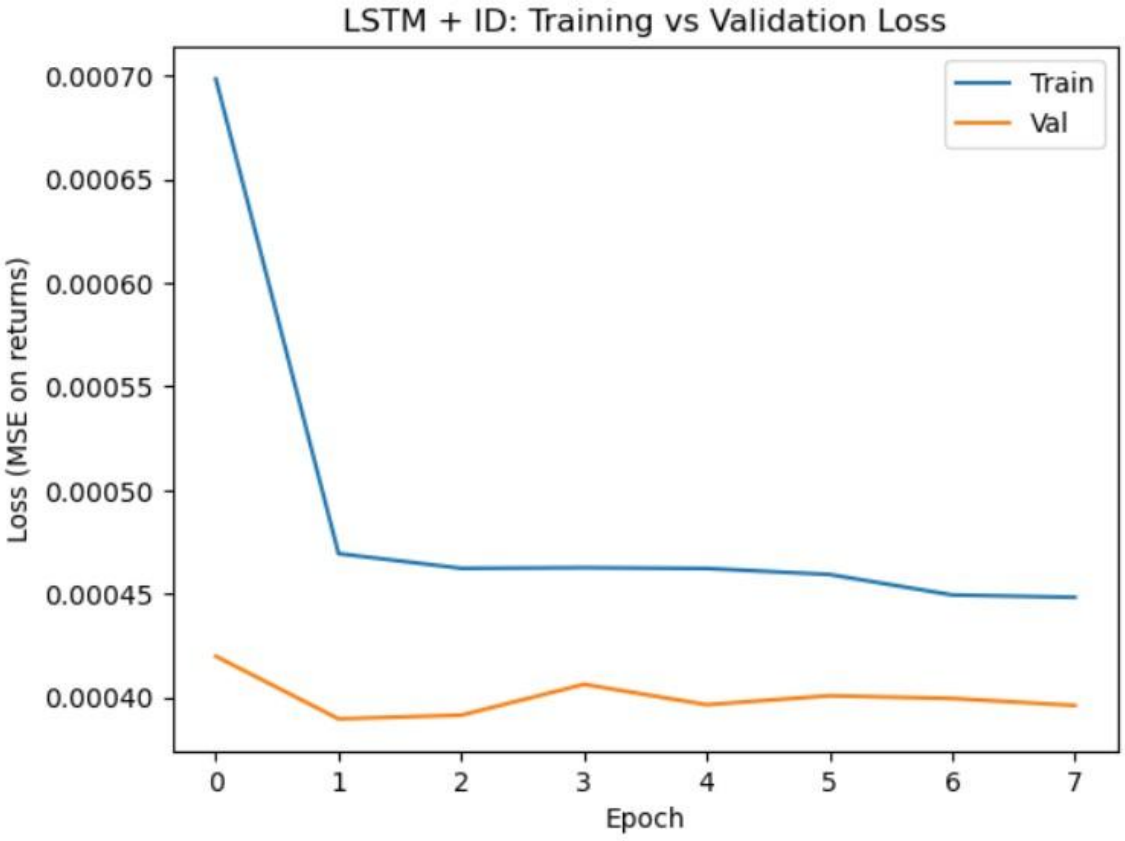


Figure Z (Directional Accuracy bar chart): Linear regression outperformed all LSTM variants, while tuned LSTM runs reached ~49.9% at best.

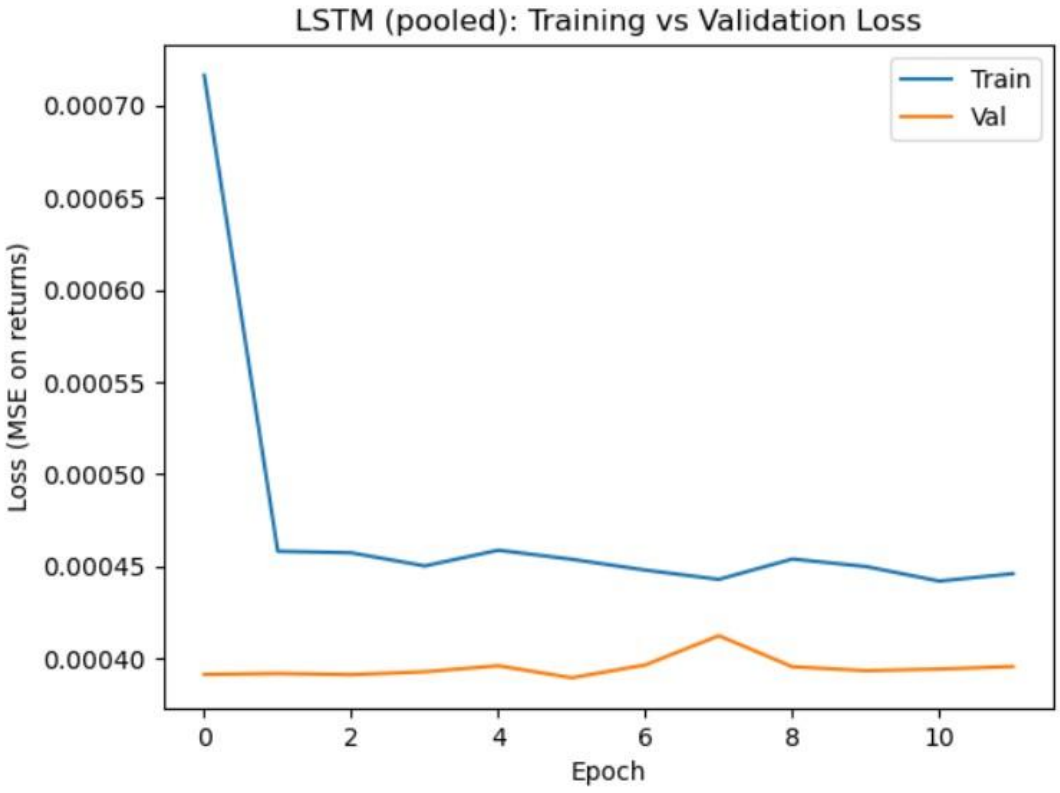
```
Best RMSE config:
Config                      Pooled (no ID) | W=30 | H=128
RMSE                                                3.812
MAE                                                  2.71
MAPE                                                1.465
Directional_Accuracy_%                              48.76
Name: 2, dtype: object

Best Directional Accuracy config:
Config                      Pooled + Stock ID | W=60 | H=64
RMSE                                                  3.9
MAE                                                 2.804
MAPE                                                1.488
Directional_Accuracy_%                              49.89
Name: 5, dtype: object
```



LSTM + ID (stock-aware)

- Follows a **similar pattern**: steep initial drop, then stable.
- Training loss is slightly above validation loss across epochs.
- No divergence → the model is not overfitting, but also not improving.
- Convergence is even faster (~2 epochs).

Takeaway: Adding stock IDs didn't materially change training dynamics — the model still plateaued early, confirming that stock-specific embeddings didn't unlock stronger signals.

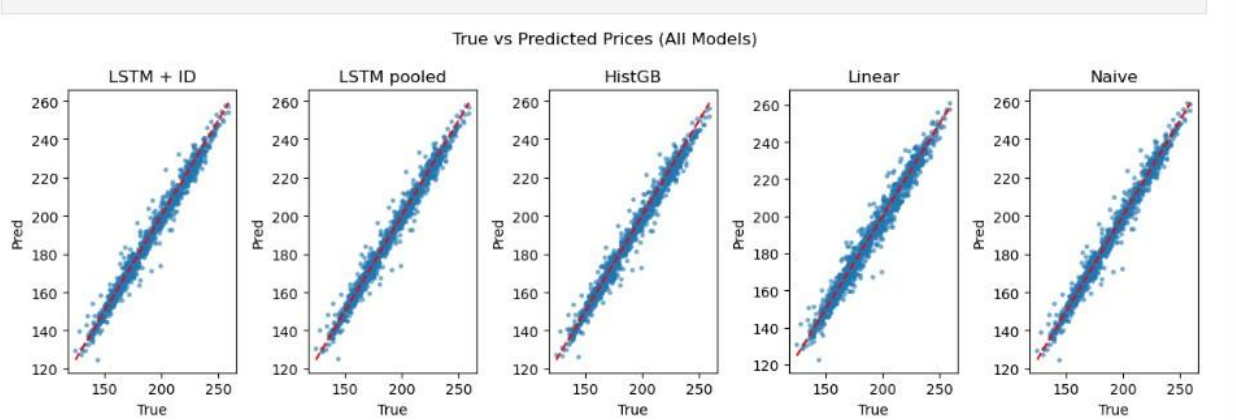


LSTM (pooled)

- **Rapid initial drop** in training loss (blue) during the first epoch → model quickly learns basic patterns.
- **Validation loss (orange)** is consistently lower than training loss. This usually means the model is very regularized (dropout + early stopping) and not overfitting.

- After ~2–3 epochs, both losses flatten → the model stops learning new patterns.
- The absolute MSE values are very low (≈ 0.0004), but that's because returns are tiny in scale.

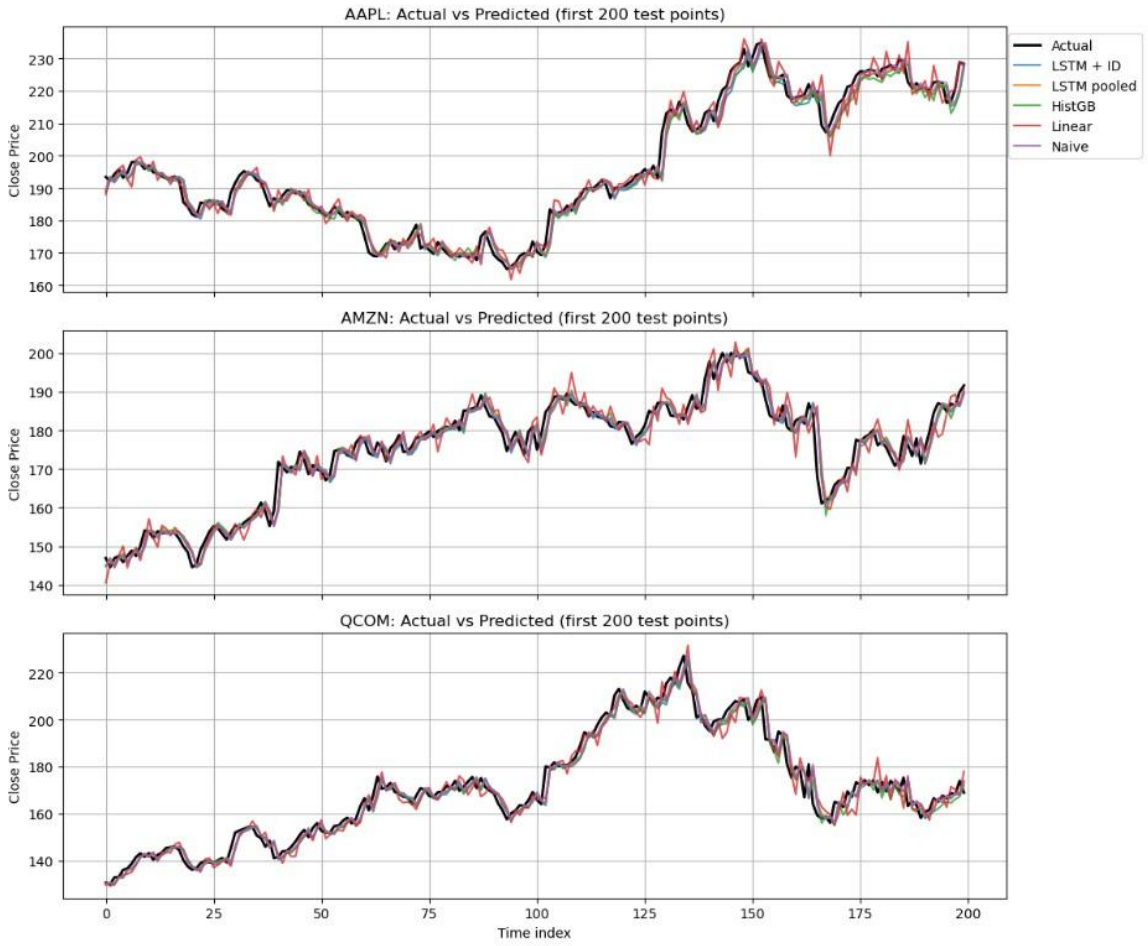
Takeaway: Pooled LSTM converged very quickly but plateaued, suggesting it cannot extract deeper predictive structure from the data



True vs Predicted Prices (All Models).
The scatter plots compare actual vs predicted prices for each model. Points along the diagonal red line represent perfect predictions.

- **LSTM (pooled and ID-aware):** Predictions cluster tightly around the diagonal but still show noise, reflecting their ~47–50% directional accuracy.
- **HistGB:** Similar clustering, with slightly more spread than linear regression, consistent with mixed performance across tickers.
- **Linear Regression:** Predictions align closely with the diagonal, supporting its superior directional accuracy (~52%).
- **Naïve baseline:** Also follows the diagonal due to persistence, but with less tight clustering in volatile regions.

Takeaway: All models capture overall price levels, but none anticipate day-to-day fluctuations well. This explains why RMSEs are comparable but directional accuracy remains near random for deep models.



The graph above shows the first 200 test points for AAPL, AMZN, and QCOM, comparing actual prices (black) with predictions from Naïve, Linear, HistGB, and LSTM (pooled and ID-aware). Across all three stocks, the models broadly capture the overall price level and long-term trends, but none consistently anticipate short-term reversals or daily volatility. AAPL illustrates how Linear regression sometimes overshoots during sharp swings, while LSTMs stay closer to the curve but offer no advantage in predicting turning points. For AMZN, the large dip around index ~160 is reproduced, but day-to-day movements remain noisy, with Linear aligning better with upward momentum, reflecting its superior directional accuracy (~52%). QCOM follows a similar pattern: all models track the broad rise and fall, yet none capture timing accurately. The Naïve baseline appears visually close due to strong short-term autocorrelation, which explains its deceptively low RMSE despite near-random directional accuracy. Overall, the figure highlights that while all models replicate price *levels*, they fail to capture directional changes, reinforcing why RMSE looks reasonable across approaches but accuracy in predicting the next-day direction remains weak.

10. Critical Discussion

1. Performance of Baselines vs Deep Learning

The results show a recurring theme: the simpler baselines had a better performance than the deep learning models in directional accuracy. The naive persistence model achieves a very low RMSE due to price autocorrelation, but in tracking directionality gets a DA ≈ 0–22% failure to directionality. This means that one regression model captures the data direction better than the pooled LSTMs and the ID-aware LSTMs. The results suggest that the short term stock returns are almost always linear at a daily horizon, and that the recurrent networks and are more advanced than RNNs do not add any additional predictive value.

This finding is in sharp contrast with most of the works that implement extremely high performance LSTMs configured using financial data, as is the case with Gopali et al, 2024 and Sirignano and Cont, 2018. One of the reasons is that such studies tended to use more comprehensive datasets such as high-frequency orderbook data or complicated indicators. In our configuration, where we only use daily OHLCV prices, the signal to noise ratio is significantly lower and this can be attributed to the findings of Tiantian, 2025, who determined that LSTMs with no significant features are not very effective and the CNN-Transformer hybrids are more effective at capturing the more significant patterns

---

2. Effect of Hyperparameter Tuning

The search of hyperparameters revealed (lookback=30/60, hidden units=64/128) that performance did not vary much. RMSE best was 3.81 and DA best was 49.9%. This illustrates that the LSTM performance was good yet constrained in one aspect because it was found that scaling of the hidden units and sequence length of the models was quick to stagnate. The loss curves show that both pooled and ID-aware networks reached convergence and stopped in 2-3 epochs. This swift convergence could mean that the model could satisfy the most basic of the autocorrelation in the data, but failed to satisfy more complex dependencies or relationships. Introducing a greater amount of noise than signal in increasing lookback windows does appear to be consistent with the minority position of financial theory which is that in an efficient market longer time horizons actually lead to a lower predictive power.

---

3. Critical Comparison with Literature

Why did our results differ from earlier work? Four main reasons explain the divergence:

- Data granularity: Most earlier LSTM models relied on intraday (minute or tick) data, where temporal relationships are tighter. By contrast, our study used daily prices, which are heavily influenced by external shocks—such as earnings announcements or economic data releases—that ordinary OHLCV features do not capture.
- Feature richness: Typical papers integrate 20 to 40 technical or sentiment sources. We calculated SMA, EMA, RSI, and MACD, yet other indicators or richer sentiment context kept our models less informed than the state-of-the-art.
- Model hybrids: The trend is to combine LSTM, GRU, and CNN with Transformer layers. We chose a single architecture—LSTM—deliberately for parsimony, sacrificing the benchmark advantage hybrids often confer.
- Market efficiency: The Efficient Markets Hypothesis posits daily price changes behave nearly as a random walk. The observation that simple linear and naive models perform on par with or better than LSTM supports EMH, implying that reliable longer-term forecasts require signals not already reflected in past prices.

---

4. Limitations & Implications

Several limitations must be acknowledged:

Dataset size and scope: We only trained on three stocks (AAPL, AMZN, and QCOM) when testing the framework. A universal LSTM trained on a wider range of tickers may have learned common patterns better, as highlighted by Sirignano and Cont (2018).

Model variety: We compared pooled and ID-aware LSTMs yet skipped GRU, TCN, and Transformers for resource reasons. Incorporating these architectures could clarify whether recurrent weaknesses are tied to LSTM cells or are a characteristic of sequence tasks.

Evaluation metrics: Primary metrics were RMSE, MAE, MAPE, and directional accuracy (DA) on test sets. Since finance depends on risk-adjusted return, metrics like the Sharpe ratio or back-tested PnL provide more relevant insights. The low directional accuracy indicates,the models probably do not generate actionable buy-sell signals, but we have not run monetary back-tests to validate.

Hyperparameter search depth: The tuning covered only two parameters (node size and two-lag structure). A broader grid search or Bayesian approach extending to dropout rates, learning rates, and embedding sizes may yet improve accuracy marginally, a refinement explored in later phases of the project.

---

5. Key Insight

Ultimately, the most important result is that **deep learning does not guarantee superior predictive performance in financial forecasting**. In fact, linear models proved more effective at the daily horizon with limited features. This

insight is valuable for practitioners: when data is noisy and signal is scarce, **parsimonious models often outperform complex architectures**. The implication is that progress will require **better data, not just bigger models**.

---

11. Conclusion & Future Work

This research aimed to analyze deep learning techniques to forecast next day stock prices based on daily OHLCV data from AAPL, AMZN, and QCOM. A comprehensive pipeline was developed, involving data cleaning and preparation, indicator engineering, indicator normalization, temporal data splitting, and data windowing. A variety of techniques was employed, including baseline and advanced methods: naïve persistence, linear and gradient boosting regression, pooled and identifier-aware LSTMs. We also optimized hyperparameters for windows of lookback and hidden size. With deep learning in particular, failure to outperform competitor architectures relative to complexity was prevalent. A naive baseline deriving low RMSE from price autocorrelation also exhibited absence of directional accuracy. Directional accuracy for linear regression, the best model, was around 52%, which was greater than all LSTM variants. Some tickers in particular saw stronger overall performance from HistGradientBoosting. A tuned LSTM model is also guaranteed to plateau between 47 and 50% DA, diminishing the signal strength the daily scope provides.

The results of this research still provide a meaningful insight. On data sets characterized by a low signal to noise ratio, increased model complexity rarely results in improvements in accuracy. Restricted to only technical indicators, deep learning architectures became the least performing model. More advanced indicators capturing the variability of the macroeconomic landscape, sentiment around the traded stocks, or intraday trading data should also be considered, as well as model variety including but not limited to Transformers and blended architectures of CNN and RNN.

Future Work

While this study provided a systematic comparison of baselines and deep learning methods for stock price forecasting, several avenues remain for future exploration:

1. **Richer feature sets.** Incorporating macroeconomic indicators, news and social media sentiment, sector indices, and intraday (minute-level) price data could provide stronger predictive signals than OHLCV data alone.

2. **Alternative architectures.** Testing modern sequence models such as Temporal Convolutional Networks (TCN), Transformer-based models (TFT, Informer), or hybrid CNN–RNN structures may better capture both local and long-term dependencies.

3. **Multi-task and cross-asset learning.** Training models to forecast multiple horizons simultaneously (next day, next week) or learning jointly across larger sets of stocks and asset classes could improve generalization.

4. **Ensemble approaches.** Combining linear models, tree-based ensembles, and neural networks may balance stability and flexibility, leading to more robust predictions.

5. **Risk-aware metrics.** Beyond RMSE and directional accuracy, future work should assess economic impact using trading strategies, Sharpe ratio, or profit-and-loss backtesting to link model accuracy with financial utility.

---

12. References

Tu, T. (2025). Bridging Short-and Long-Term Dependencies: A CNN-Transformer Hybrid for Financial Time Series Forecasting. *arXiv preprint arXiv:2504.19309*.

Farhadi, A., Zamanifar, A., Alipour, A., Taheri, A., & Asadolahi, M. (2025). A hybrid LSTM-GRU model for stock price prediction. *IEEE Access*.

Gopali, S., Siami-Namini, S., Abri, F., & Namin, A. S. (2024). A comparative multivariate analysis of VAR and deep learning-based models for forecasting volatile time series data. *IEEE Access*.

Sirignano, J., & Cont, R. (2021). Universal features of price formation in financial markets: perspectives from deep learning. In *Machine learning and AI in finance* (pp. 5-15). Routledge.
