Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

MSc Quantitative Finance and Data Science

Wintersemester 2025/2026

Forecasting Stock Prices with LSTM and GRU: Deep Learning Based Trading vs. Buy & Hold

Master Seminar Deep Learning

Eric Christopher
582715

Abstract

This study investigates the predictive performance of Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks for next-day stock price forecasting and evaluates the effectiveness of trading strategies derived from these predictions. Using daily stock data from 2015 to 2024 for ten diverse companies, technical indicators including SMA, RSI, MACD, and volatility were employed as model inputs. Both LSTM and GRU models accurately captured medium-term price trends and directional changes, with GRU consistently achieving lower forecasting errors. Trading strategies based on model predictions outperformed a passive Buy & Hold approach, generating higher returns and lower downside risk as measured by 5% Historical Value at Risk (VaR). While GRU exhibited superior predictive accuracy, LSTM achieved higher final portfolio values for most individual stocks, highlighting that lower prediction error does not always result in higher trading profits. In a multi-stock portfolio, GRU provided the most favorable capital preservation and risk-return balance. These findings show that deep learning models can effectively support profitable and risk-controlled trading decisions, with relevance for both individual stock and portfolio-level investment strategies.

Contents

List of Figures

List of Tables

1 Introduction

Financial markets are constantly changing and influenced by many factors, making stock prices difficult to predict. Prices move in response to macroeconomic events, corporate news, investor sentiment, and geopolitical developments, resulting in sequences that are highly volatile, nonlinear, and often non-stationary. For example, the MSCI Inc. (MSCI) stock has experienced substantial fluctuations over the past years, with pronounced drawdowns and recoveries that reflect the dynamic behavior of global equity markets (see Figure 1). These large swings illustrate how rapidly market conditions can evolve, challenging both traditional forecasting techniques and practical investment decision-making.

Figure 1: MSCI Inc. Stock Price over Last Years
Source: Yahoo Finance

Accurate forecasting of stock prices is important for investors, portfolio managers, and financial institutions because it can inform better trading strategies and risk management practices. However, the complexity of financial time series makes this task extremely difficult. Traditional statistical models such as autoregressive integrated moving average (ARIMA) or simple moving average approaches rely on assumptions of linear relationships and stable temporal dynamics, which are often violated in practice.

In response to these limitations, deep learning approaches have gained increasing attention in financial forecasting research. Recurrent neural networks, particularly Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures, are specifically designed to model sequential data and capture long-term temporal dependencies. Shahi et al. (2020) [SSNG2020] provide a comparative study of deep learning models for stock price prediction and demonstrate that LSTM- and GRU-based models outperform several traditional approaches in modeling nonlinear financial time series. Their work also explores the incorporation of additional information, such as sentiment data, to enhance predictive performance. More recently, Velarde et al. (2025) [VBBHL2025] present an open-source and reproducible implementation of LSTM and GRU models for stock forecasting, emphasizing methodological transparency and systematic comparison of both architectures.

While these studies provide important insights into the predictive performance of LSTM and GRU models, they mainly focus on statistical forecasting accuracy. In other words, they evaluate how well the models predict future prices, but they pay less attention to whether these predictions lead to better investment results in practice. In particular, there is limited analysis of how forecast-based trading strategies

perform compared to a passive Buy & Hold strategy.

This study addresses these gaps by investigating whether LSTM and GRU models can accurately forecast next-day closing prices for ten individual stocks using price-based technical indicators as input features. Beyond evaluating statistical prediction accuracy, the study assesses the practical relevance of the models by implementing forecast-based trading strategies and comparing their performance to a passive Buy & Hold benchmark.

To evaluate practical trading performance, we consider two portfolio scenarios. First, ten separate single-stock portfolios, and second, a multi-stock portfolio where the total capital is evenly distributed across the ten stocks. Model performance is evaluated using error metrics, final portfolio value, and 5% Historical Value at Risk (VaR). By integrating predictive evaluation with portfolio return and risk analysis, this study provides a comprehensive comparison of LSTM and GRU models in a practical trading context. The findings of this study have practical relevance for investors and portfolio managers, as they demonstrate how deep learning-based forecasts can support more informed, risk-aware trading decisions in both individual stocks and diversified portfolios.

2 Theoretical Foundation

2.1 Financial Market and Technical Analysis

cf. [M2021] Ch. 1

In financial markets, technical analysis is a widely used approach to make trading decisions. Instead of focusing on fundamental data like company earnings, technical analysis examines historical price and volume patterns to anticipate future price movements. Price is usually the most important factor, as it directly reflects market sentiment and behavior. Traders often visualize these data in charts and apply various mathematical tools, called technical indicators, to identify trends and signals.

Technical indicators can be thought of as mathematical operators applied to past prices, transforming raw price data into values that represent trends, momentum, or volatility. For instance, moving averages smooth out short-term fluctuations to reveal the underlying trend, while indicators like Moving Average Convergence-Divergence (MACD) show changes in the speed of price movements.

Two useful concepts borrowed from physics are velocity and acceleration, which describe how quickly a price changes and how that speed itself changes over time. Here, velocity is the rate of price change. A positive velocity means the price is rising, and a negative velocity means it is falling. Acceleration is the rate of change of velocity. If it is positive, upward momentum is increasing, and if negative, momentum is slowing down.

Many trading strategies are based on these ideas. For example, a simple tactic is to buy when the velocity of price is positive and sell when it is negative. Research has shown that popular indicators, such as the MACD, essentially act as velocity indicators, tracking the speed of price movements, while the MACD histogram can

be interpreted as an acceleration indicator.

By transforming historical price data into indicators that reflect trends, momentum, and volatility, models like LSTM and GRU can leverage these features to detect long-term patterns in sequential data, improving the prediction of future stock prices.

2.2 Technical Indicators for Stock Price Forecasting

cf. [L2013] Ch. 3 and 6.1, [M2021] Ch. 12.2 and 12.4

Technical indicators are essential tools in financial analysis, used to extract meaningful information from historical price data. They help identify trends, momentum, and volatility, which are crucial for predicting future price movements. When applied as input features for sequential models like LSTM and GRU, these indicators provide structured signals that allow the models to capture temporal dependencies and enhance forecasting accuracy.

2.2.1 Simple Moving Average (SMA)

The Simple Moving Average (SMA) over a period of N bars is defined as:

$$\mathrm{SMA}_N^{(t)} = \frac{1}{N} \sum_{k=0}^{N-1} P^{(t-k)},$$

where $P^{(t)}$ is the closing price at time t, N is the lookback window length, t is the current time index, and k is the lag index within the window. The SMA smooths short-term price fluctuations and highlights the underlying trend. The choice of N affects both the smoothness of the trend and the lag of the indicator. A larger N produces a smoother trend but increases lag. In this study, the SMA is computed with window lengths $N = 5$ and $N = 20$.

2.2.2 Relative Strength Index (RSI)

RSI is a momentum oscillator that measures the magnitude of recent price changes to evaluate overbought or oversold conditions. Let

$$\Delta P^{(t)} = P^{(t)} - P^{(t-1)}$$

denote the price change at time t.

The average gain and loss over N periods are:

$$\overline{G}^{(t)} = \frac{1}{N} \sum_{k=0}^{N-1} \max(\Delta P^{(t-k)}, 0), \quad \overline{L}^{(t)} = \frac{1}{N} \sum_{k=0}^{N-1} \max(-\Delta P^{(t-k)}, 0).$$

Here, $\overline{G}^{(t)}$ and $\overline{L}^{(t)}$ denote the average gain and average loss over the past N periods, respectively.

The relative strength is:

$$RS^{(t)} = \frac{\overline{G}^{(t)}}{\overline{L}^{(t)}},$$

and the RSI is:

$$\text{RSI}^{(t)} = 100 - \frac{100}{1 + RS^{(t)}}.$$

RSI values range from 0 to 100. Values above 70 indicate that the asset may be overbought, suggesting a potential price correction, while values below 30 indicate the asset may be oversold, suggesting a potential price rebound.

2.2.3 Moving Average Convergence-Divergence (MACD)

MACD is a momentum indicator that highlights changes in trend strength and direction. It is computed as the difference between a fast and a slow exponential moving average (EMA):

$$\text{MACD}^{(t)} = \text{EMA}_{\text{fast}}^{(t)} - \text{EMA}_{\text{slow}}^{(t)},$$

where the EMA is calculated as:

$$\text{EMA}_M^{(t)} = \alpha P^{(t)} + (1 - \alpha)\text{EMA}_M^{(t-1)}, \quad \alpha = \frac{2}{M+1}.$$

Here, M is the EMA period length, α is the smoothing factor, and $\text{EMA}_M^{(t-1)}$ is the EMA value at the previous time step. Typically, the fast EMA uses $M = 12$ and the slow EMA uses $M = 26$. A positive MACD indicates upward momentum, while a negative MACD indicates downward momentum. Crossovers between the fast EMA and slow EMA generate trading signals.

2.2.4 Price Volatility

Volatility measures the magnitude of price fluctuations and is important for risk assessment and forecasting:

$$\sigma^{(t)} = \sqrt{\frac{1}{N}\sum_{k=0}^{N-1}\left(r^{(t-k)} - \overline{r}^{(t)}\right)^2}, \quad r^{(t)} = \frac{P^{(t)} - P^{(t-1)}}{P^{(t-1)}},$$

where $r^{(t)}$ is the return at time t, and the mean return over the past N periods is defined as

$$\overline{r}^{(t)} = \frac{1}{N}\sum_{k=0}^{N-1} r^{(t-k)}.$$

Here, $\sigma^{(t)}$ represents the rolling standard deviation of returns over N periods. Higher volatility reflects greater uncertainty in price movements, which LSTM and GRU models can use to adjust predictions dynamically based on market risk.

2.3 Long Short-Term Memory (LSTM)

cf. [A2023] Ch. 8.5, [B2024] Ch. 10, [DTSKB2023], [HMN2020], [S2021] Ch. 4.1-4.2

The Long Short-Term Memory (LSTM) architecture, introduced by Hochreiter and Schmidhuber (1997), was specifically designed to overcome the vanishing and exploding gradient problems that typically prevent standard recurrent neural networks from learning long-range dependencies. In a simple recurrent neural network (sRNN), the hidden state is updated using a single nonlinear transformation of the current input and the previous hidden state. However, repeated multiplication by the same weight

matrices causes gradients to either shrink to zero or grow uncontrollably during back-propagation through time, making it difficult to capture dependencies that span many time steps.

The key idea behind LSTM is the use of a memory cell that can preserve information over long sequences through a structure known as the constant error carousel (CEC), which stabilizes gradient flow during training and allows the network to accumulate and store information without immediate decay. Each LSTM block consists of a cell state and three gates, the input gate, forget gate, and output gate that regulate how information enters, stays in, or leaves the memory cell. These gates are small neural subnetworks with their own weight matrices: W_* denote the input–to–gate weights, R_* the recurrent (hidden–to–gate) weights, b_* the bias vectors, and p_* the peephole weights that allow each gate to directly access the cell state for precise temporal control.

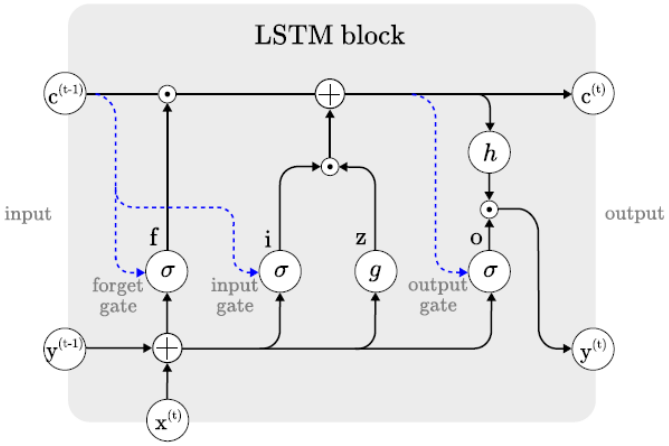

Figure 2: Long Short-Term Memory (LSTM) Cells
Source: [HMN2020]

At every time step t, the model receives an external input $x^{(t)} \in \mathbb{R}^m$ and the previous hidden state $y^{(t-1)} \in \mathbb{R}^n$. The first step is to compute the candidate cell input

$$z^{(t)} = \tanh\left(W_z x^{(t)} + R_z y^{(t-1)} + b_z\right),$$

which represents new information that could potentially be integrated into the memory cell, where $W_z \in \mathbb{R}^{n \times m}$ transforms the input, $R_z \in \mathbb{R}^{n \times n}$ processes the previous hidden state, and $b_z \in \mathbb{R}^n$ is the bias term.

The input gate controls how much of this candidate information is allowed into the cell state:

$$i^{(t)} = \sigma\left(W_i x^{(t)} + R_i y^{(t-1)} + p_i \odot c^{(t-1)} + b_i\right),$$

where W_i, R_i, and b_i are the input gate parameters, and the peephole vector $p_i \in \mathbb{R}^n$ allows the gate to observe the previous cell state $c^{(t-1)}$.

The forget gate is defined as

$$f^{(t)} = \sigma\left(W_f x^{(t)} + R_f y^{(t-1)} + p_f \odot c^{(t-1)} + b_f\right),$$

which determines how much of the previous cell state should be retained or discarded. Here, W_f maps the input to the forget gate, R_f maps the previous hidden state, b_f is the bias term, and p_f connects the cell state to the gate. This mechanism enables the LSTM to reset or preserve memory when necessary.

The cell state is then updated by combining preserved memory with newly added information:

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot z^{(t)}.$$

Next, the output gate determines which parts of the updated cell state influence the new hidden state:

$$o^{(t)} = \sigma\left(W_o x^{(t)} + R_o y^{(t-1)} + p_o \odot c^{(t)} + b_o\right),$$

where W_o, R_o, b_o, and p_o serve the same structural roles as in the other gates.

Finally, the hidden output is produced by modulating the activated cell state with the output gate:

$$y^{(t)} = \tanh\left(c^{(t)}\right) \odot o^{(t)}.$$

While the equations presented include peephole connections to demonstrate the full LSTM architecture, the LSTM models used in this work (Keras LSTM layer) do not employ them. Consequently, the gating mechanisms depend only on the current input and the previous hidden state.

Because each gate and the candidate update require separate weight matrices and biases, an LSTM has significantly more parameters than a simple RNN. For an input dimension m and hidden dimension n, the total number of parameters is

$$4(n^2 + nm + n) = 4n(n + m + 1),$$

which can exceed one million parameters in practical applications (e.g., more than 1,28 million when $m = n = 400$). Although computationally heavier, the gating structure and stable gradient flow allow LSTM to model long-term dependencies effectively, enabling successful applications in speech recognition, natural language processing, and financial time-series forecasting.

2.4 Gated Recurrent Unit (GRU)

cf. [A2023] Ch. 8.6, [B2024] Ch. 10, [DTSKB2023], [S2021] Ch. 5.1-5.2

The Gated Recurrent Unit (GRU) architecture, proposed by Cho et al. (2014), is a simplified variant of the Long Short-Term Memory (LSTM) network designed to reduce computation time and the number of parameters while retaining the ability to capture long-term dependencies in sequential data. Like LSTM, GRU addresses the vanishing and exploding gradient problems present in simple recurrent neural networks (sRNNs) by introducing gating mechanisms. However, GRU uses only two gates the reset gate $r^{(t)}$ and the update gate $z^{(t)}$, compared to the three gates in LSTM.

At every time step t, the GRU receives an external input $x^{(t)} \in \mathbb{R}^m$ and the previous hidden state $y^{(t-1)} \in \mathbb{R}^n$. The reset gate is computed as:

$$r^{(t)} = \sigma(W_r x^{(t)} + R_r y^{(t-1)} + b_r),$$

and the update gate as:

$$z^{(t)} = \sigma(W_z x^{(t)} + R_z y^{(t-1)} + b_z),$$

where $W_r, W_z \in \mathbb{R}^{n \times m}$ are the input weight matrices, $R_r, R_z \in \mathbb{R}^{n \times n}$ are the recurrent (hidden-to-gate) weight matrices, and $b_r, b_z \in \mathbb{R}^n$ are the bias vectors. The sigmoid function $\sigma(\cdot)$ restricts the gate activations to the interval $[0, 1]$, controlling the amount of information flow.

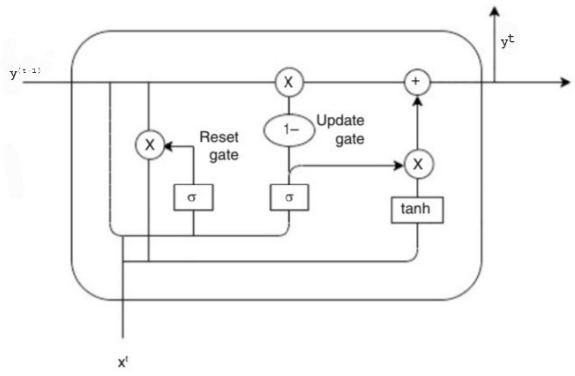

Figure 3: Gated Recurrent Unit (GRU) Cells
Source: [DTSKB2023]

Next, the candidate hidden state $\tilde{y}^{(t)}$ is computed as:

$$\tilde{y}^{(t)} = \tanh(W_y x^{(t)} + R_y(r^{(t)} \odot y^{(t-1)}) + b_y),$$

where $W_y \in \mathbb{R}^{n \times m}$ and $R_y \in \mathbb{R}^{n \times n}$ are the input and recurrent weight matrices, respectively, $b_y \in \mathbb{R}^n$ is the bias vector, and $\odot$ denotes element-wise multiplication. The reset gate $r^{(t)}$ determines which parts of the previous hidden state $y^{(t-1)}$ are ignored or reset when computing the candidate state. If $r^{(t)}$ is close to zero, the past information is largely discarded, allowing the network to focus on the current input.

The final hidden state $y^{(t)}$ is then obtained by interpolating between the previous hidden state and the candidate state using the update gate:

$$y^{(t)} = (1 - z^{(t)}) \odot y^{(t-1)} + z^{(t)} \odot \tilde{y}^{(t)}.$$

The update gate $z^{(t)}$ controls how much of the past hidden state is carried forward to the current step. For short-term dependencies, the update gate may be small, emphasizing new information from $\tilde{y}^{(t)}$, whereas for long-term dependencies, it may be larger to retain important historical information.

Compared to LSTM, GRU eliminates the separate output gate and merges the forget and input gates into a single update gate. Consequently, GRU has fewer parameters, requiring only three sets of weight matrices and biases for a hidden dimension n and input dimension m, resulting in a total parameter count of:

$$3(n^2 + nm + n) = 3n(n + m + 1),$$

which is smaller than the LSTM parameter count $4n(n+m+1)$, reflecting the elimination of the output gate and its associated parameters. For example, if $m = n = 100$,

a GRU saves $n(n+m+1) = 20.100$ parameters compared to a corresponding LSTM.

One of the key differences between LSTM and GRU is in how they expose and control information. In LSTM, the output gate regulates how much of the cell state is visible to other units, whereas in GRU, the full candidate hidden state is exposed without additional gating. Moreover, LSTM controls the new memory content to be added while preserving the previous state, whereas GRU controls the flow of past information when computing the candidate hidden state.

Both architectures use an additive property to retain important past information, preventing it from being overwritten by new input, which allows both LSTM and GRU to model long-term dependencies effectively. The simplified structure of GRU offers computational advantages and can achieve comparable performance to LSTM in many sequence modeling tasks, including language modeling and financial time-series forecasting.

3 Data and Preprocessing

3.1 Dataset Description

The empirical analysis is based on daily stock market data retrieved from Yahoo Finance, covering the period 2015-2024. The dataset includes ten companies from different industries MSCI (MSCI Inc.), MCD (McDonald's Corporation), BA (Boeing Company), PFE (Pfizer Inc.), ES (Eversource Energy), INTC (Intel Corporation), MDLZ (Mondelez International Inc.), AVY (Avery Dennison Corporation), BXP (BXP Inc.), APA (APA Corporation) to ensure diversity in market behavior, volatility, and sector-specific dynamics. This allows the forecasting models to be evaluated across a broad range of market conditions.

For each stock, the dataset includes the adjusted closing price, open, high, low, and trading volume, which serve as the foundation for deriving additional technical indicators.

To avoid look-ahead bias, the data is split chronologically. The training period spans 2015-2022, and 20% of this period is used as a validation set. The test period covers 2023-2024, representing unseen data used for model evaluation.

3.2 Feature Engineering

Several technical indicators were computed to expand the feature. These include the 5-day and 20-day Simple Moving Averages (SMA5, SMA20), which represent short- and medium-term trend dynamics; the Relative Strength Index (RSI), which measures momentum and highlights potential overbought or oversold conditions; the Moving Average Convergence Divergence (MACD), which captures changes in momentum through the interaction of fast and slow exponential moving averages; and volatility, calculated as the 20-day rolling standard deviation of daily returns, serving as a measure of market uncertainty.

To illustrate the behavior of the indicators across the full set of stocks, each stock's technical features were examined visually. Figure 4 shows the indicators for McDonald's (MCD) as a representative example. The adjusted closing price exhibits

a clear upward trend throughout the sample period, with notable corrections, such as the sharp decline in early 2020. The SMA5 closely follows short-term price movements, while the smoother SMA20 captures broader medium-term trends. Bullish and bearish SMA crossovers often align with corresponding short-term price movements, confirming the indicators' ability to capture trend momentum.

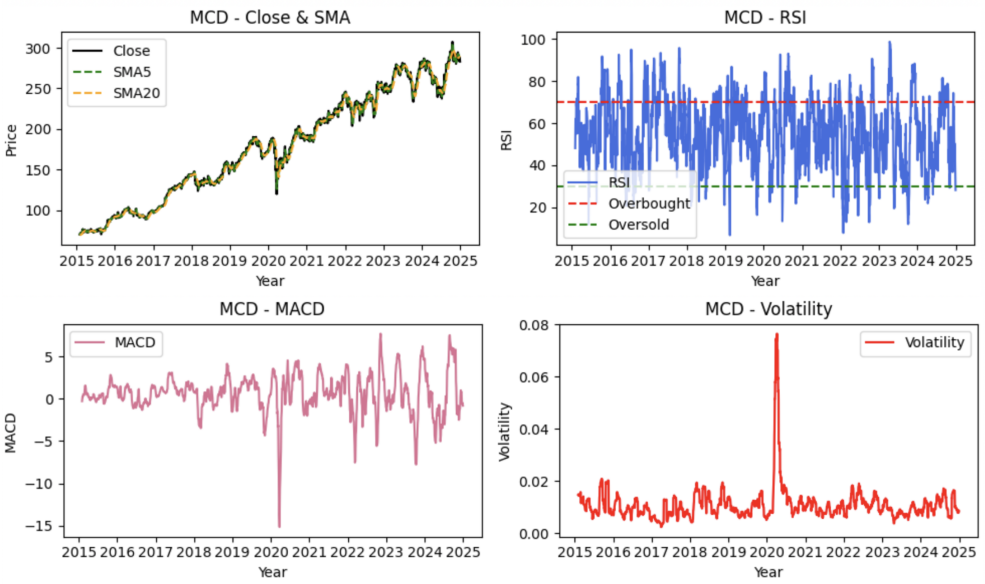

Figure 4: McDonald's Technical Indicators

Across the other stocks, similar patterns are observed, although the magnitude and frequency of movements vary according to each stock's volatility and market behavior. For instance, higher volatility stocks may display more frequent SMA crossovers and wider RSI swings, while large-cap defensive stocks like MCD show smoother trends and fewer extreme oscillations. The RSI fluctuates over time for all stocks, with values occasionally surpassing the overbought threshold of 70 during strong momentum phases and dropping below 30 during corrections. The MACD similarly reflects rising and falling momentum across the portfolio, while volatility spikes correspond to market-wide shocks, such as the COVID-19 crisis, and differ in intensity depending on individual stock characteristics.

3.3 Target Variable and Scaling

cf. [S2023] Ch. 3.7, [S2024] Ch. 5.2.3, [SSNG2020], [VBBHL2025]

The prediction objective is to forecast the next day's Adjusted Close price. Input features include Adjusted Close, Volume, SMA5, SMA20, RSI, MACD, and Volatility.

All features and the target are scaled to the interval $[0, 1]$ using Min–Max normalization. Here, x denotes the value of any input feature or the target variable before scaling.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

This ensures all variables have the same range, which is important for gradient-based models such as LSTM and GRU, as it improves convergence and prevents features with large magnitudes from dominating training.

The scaler is fitted only on the training set to avoid data leakage. Predicted values are inverse-transformed to the original price units for evaluation and comparison with benchmark strategies like Buy & Hold.

3.4 Time Series Windowing for LSTM and GRU

Deep learning models such as LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) are designed to capture temporal dependencies in sequential data. To leverage these models effectively, historical stock price data must be structured into fixed-length sequences, also called sliding windows.

Let T denote the window length, representing the number of past days used to predict the next day's closing price. Each input sequence is therefore of the form:

$$X_t = [x_{t-T}, x_{t-T+1}, \ldots, x_{t-1}]$$

and the corresponding target is

$$y_t = x_t^{\text{Adjusted Close}}$$

where $x_t^{\text{Adjusted Close}}$ is the adjusted closing price on day t. By sliding this window over the entire training period, a set of overlapping sequences is generated, which serves as input to the LSTM and GRU networks. This sequence-based structure allows the models to learn short-term trends, momentum effects, and volatility clustering.

4 Methodology

4.1 Model Architecture

cf. [SSNG2020], [VBBHL2025]

This work employs LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) networks to forecast the next day's stock price. These models are chosen because they are capable of capturing temporal dependencies in sequential data, which is essential for financial time series exhibiting trends, momentum, and volatility clustering. The input to each model consists of sequences of 60 past trading days [VBBHL2025], including features such as Adjusted Close, Volume, SMA5, SMA20, RSI, MACD, and Volatility. A 60-day lookback window provides sufficient historical context to capture short- and medium-term market dynamics, including momentum persistence, mean reversion behavior, and volatility clustering, while avoiding the influence of irrelevant long-term history.

Each model contains a single recurrent layer (LSTM or GRU) with a tunable number of hidden units of 100, 150, or 200, followed by a dropout layer with rates ranging from 0 to 0,5 (step size of 0,1) to mitigate overfitting. The chosen range of hidden units provides sufficient representational capacity to model nonlinear temporal relationships without introducing unnecessary complexity that could reduce generalization. Dropout regularization is particularly important for financial time series due

to their noisy and non-stationary nature, as it encourages the network to learn more robust and distributed feature representations.

The recurrent layers use the tanh activation function [SSNG2020] to transform the hidden states, enabling the models to capture nonlinearities and maintain values within a bounded range, which stabilizes training and improves convergence.

The output layer consists of a single dense neuron that produces the predicted next day's Adjusted Close price. Models are trained using the Mean Squared Error (MSE) loss function and optimized using the Adam optimizer [VBBHL2025]. MSE is appropriate for the regression task because it penalizes larger errors more heavily, which is important in financial forecasting where large deviations can be costly. Adam is selected due to its adaptive learning rate mechanism and robustness to noisy gradients.

A batch size of 32 is selected because it offers an effective compromise between computational efficiency and reliable gradient updates, enabling the model to generalize well without imposing high memory demands. Training is limited to a maximum of 20 epochs, which is generally sufficient for convergence given the dataset size and model complexity, while also reducing the risk of overfitting and limiting training time.

Early stopping [SSNG2020] is applied based on validation performance to prevent overfitting and improve generalization by retaining the model state with the best validation results. The training and validation sets are used for model fitting and hyperparameter tuning, while a separate test set ensures an unbiased evaluation of predictive performance.

4.2 Trading Strategy

The trading strategy is derived from model predictions to simulate actionable investment decisions. Signals are generated by comparing the predicted price with yesterday's actual closing price. A Buy signal occurs when the predicted price exceeds the previous day's close and no position is held, initiating the purchase of the maximum number of shares. A Hold signal occurs when the predicted price is higher than yesterday's close and a position is already held. A Sell signal occurs when the predicted price is lower than or equal to yesterday's close and a position is currently held, triggering the sale of all shares. No Action is taken if the predicted price is lower than or equal to yesterday's close and no position is held.

All trades are executed at the next day's opening price to reflect realistic market conditions for investors. Using the next day's open ensures that the trading strategy does not rely on future information, avoiding look-ahead bias. In practice, traders cannot transact at the same day's closing price immediately after the market closes, making the next day's opening price a feasible and actionable execution price. A fixed transaction cost of 1 USD per trade is applied, reflecting realistic brokerage fees, such as those charged by platforms like Trade Republic.

For the portfolio analysis, we implement two setups. In the single-stock portfolio, each of the 10 stocks is treated independently, with an initial capital of 5.000 USD allocated to each stock separately. This means 10 individual single-stock portfolios are simulated, one for each stock.

In the multi-stock portfolio, a total capital of 10.000 USD is equally divided across the 10 stocks, with 1.000 USD invested per stock, forming a single combined portfolio. The performance of both portfolio setups is evaluated by comparing their returns to a Buy & Hold strategy, in which the entire portfolio capital is invested at the first day's opening price and held until the end of the period.

4.3 Evaluation Metrics

cf. [A2018] Ch. 4, [B2023] Ch. 1.7.2, [S2023] Ch. 3.4

Model performance is evaluated using standard regression metrics. Let y_t denote the actual closing price and $\hat{y}_t$ the predicted closing price at time t. The metrics are defined as:

$$\text{Mean Absolute Error (MAE)} = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t|$$

$$\text{Mean Squared Error (MSE)} = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2$$

$$\text{Root Mean Squared Error (RMSE)} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}$$

MAE measures the average magnitude of prediction errors, while MSE and RMSE penalize larger deviations more strongly, which is critical in financial forecasting where large errors can have significant consequences. Predicted values are inverse-transformed from the scaled [0,1] range to their original price units for meaningful evaluation and comparison with benchmark strategies.

In addition to standard regression metrics, the risk associated with the trading strategies is quantified using Value at Risk (VaR) based on the historical simulation method. While MAE, MSE, and RMSE capture the numerical accuracy of predictions, they do not provide information about potential losses, which is a critical consideration in financial applications. To address this, predicted prices are first converted into predicted returns defined as

$$r_t = \frac{V_t - V_{t-1}}{V_{t-1}},$$

where r_t is the portfolio return at time t, V_t is the portfolio value at time t, and V_{t-1} is the portfolio value at the previous day.

Historical Value at Risk (VaR) at the 95% confidence level is then computed as the empirical 5th percentile of these daily returns:

$$\text{VaR}_{95\%} = \text{Quantile}_{0,05}(r_t).$$

This value represents a threshold such that only 5% of predicted daily returns fall below it. The historical simulation approach is chosen because it does not assume a specific distribution for returns. This provides additional insight into the potential downside risk associated with model-based forecasts, which is important for risk management and investment decision-making.

5 Empirical Results

5.1 Empirical Results for MSCI Stock

First, we will examine the MSCI stock to evaluate the predictive performance of the deep learning models and the profitability of the resulting trading strategies. Figure 5 (left) illustrates the comparison between actual and predicted closing prices, showing that both the LSTM and GRU models closely track the overall price dynamics of MSCI during the test period from 2023 to 2024. Although the predicted price series appear smoother than the actual prices and tend to underreact to abrupt short-term movements, both models successfully capture medium-term trends and directional changes.

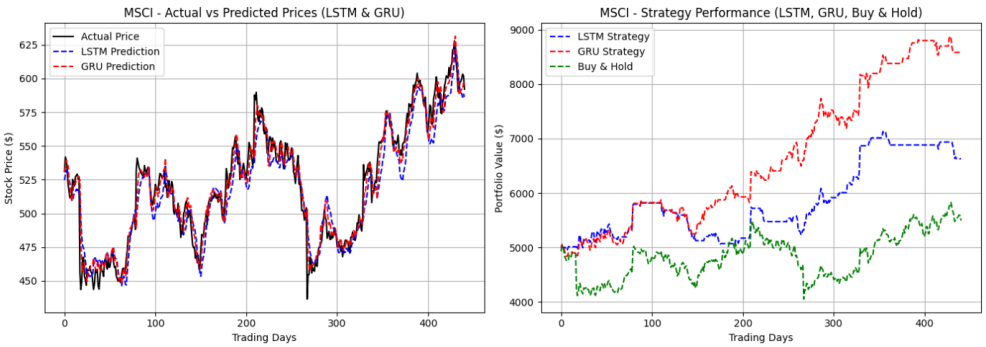

Figure 5: Actual vs. Predicted MSCI Inc. Closing Prices (left). Portfolio Values from Trading Strategies (right)

From a visual perspective, the GRU predictions exhibit a closer alignment with the actual price trajectory, particularly during periods of heightened volatility toward the end of the sample, suggesting a more stable and responsive forecasting behavior.

This visual assessment is supported by the quantitative error metrics. The LSTM model records a mean absolute error (MAE) of 10,45, a mean squared error (MSE) of 190,46, and a root mean squared error (RMSE) of 13,80. In comparison, the GRU model achieves lower errors across all measures, with an MAE of 6,88, an MSE of 99,75, and an RMSE of 9,99. Considering that the average MSCI price level during the test period ranges between approximately 450 and 600 USD, these error values correspond to relatively small deviations of about 1-2 percent. Overall, both models demonstrate satisfactory predictive accuracy, with the GRU model consistently outperforming the LSTM model.

While the trading signals generated by the LSTM and GRU models vary over time, both approaches successfully convert their forecasts into profitable trading strategies, as reflected in the final portfolio values. Starting from an initial capital of 5.000 USD, the Buy & Hold strategy results in a final portfolio value of 5.493,76 USD, corresponding to a return of approximately 9,88% over the test period. By comparison, the LSTM-based trading strategy increases the portfolio value to 6.629,70 USD, while the GRU-based strategy achieves the highest final value of 8.585,11 USD, representing returns of approximately 32,59% and 71,7%, respectively. Both deep-learning-based strategies therefore substantially outperform the passive Buy & Hold benchmark.

The risk characteristics of the strategies are captured by the 5% Historical Value at Risk (VaR), which indicates the potential worst-case daily losses with 95% confidence. The LSTM strategy has a 5% VaR of -0,0125, meaning there is only a 5% chance of a daily loss exceeding 1,25%, reflecting relatively low downside risk. The GRU strategy has a slightly higher 5% VaR of -0,0142 (1,42% daily loss), showing a marginal increase in risk compared to LSTM but achieving the higher return. In contrast, the Buy & Hold strategy exhibits the largest 5% VaR of -0,0217, implying a potential daily loss of 2,17% and showing higher downside risk.

Overall, the MSCI results demonstrate that accurate next-day price forecasting can generate profitable trading outcomes, with both deep-learning strategies offering an attractive combination of high returns and controlled risk.

5.2 Performance and Risk Analysis Across Stocks

After analyzing the MSCI stock in detail, we now compare the predictive performance and trading outcomes of the LSTM and GRU models across the other nine stocks. In terms of forecasting accuracy, measured by MAE, MSE, and RMSE, the GRU model outperforms LSTM for all 10 stocks, consistently producing lower prediction errors. For example, for BXP, GRU has an MAE of 1,19 compared to 1,50 for LSTM, while for MDLZ, GRU achieves an MAE of 0,64 compared to 0,98 for LSTM. This pattern demonstrates that GRU provides more precise next-day price forecasts across a diverse set of stocks.

Table 1: Forecasting Errors for LSTM and GRU Across Stocks

Stock	**LSTM**			**GRU**		
	MAE	MSE	RMSE	MAE	MSE	RMSE
MSCI	10,4522	190,4630	13,8008	6,8787	99,7474	9,9874
MCD	3,0745	16,4596	4,0570	2,5125	11,0638	3,3262
BA	3,8772	25,3437	5,0343	3,3641	20,4540	4,5226
PFE	0,4568	0,3378	0,5812	0,3566	0,2141	0,4627
INTC	0,9264	1,6291	1,2764	0,8428	1,3526	1,1630
MDLZ	0,9783	1,5545	1,2468	0,6435	0,7050	0,8396
AVY	3,0782	15,1623	3,8939	2,0707	7,1861	2,6807
BXP	1,5030	3,4496	1,8573	1,1939	2,2189	1,4896
ES	0,9435	1,4675	1,2114	0,7865	1,0238	1,0119
APA	0,7609	0,8950	0,9460	0,5857	0,5812	0,7624

When examining trading performance, the results are more mixed. Despite GRU's superior forecasting accuracy, the LSTM-based trading strategy achieves higher final portfolio values for seven stocks (MCD, PFE, INTC, AVY, BXP, ES and APA), while GRU leads for three stocks (MSCI, BA, MDLZ).

For instance, LSTM generates a final portfolio value of 6.582,73 USD for MCD, higher than GRU's 5.561,10 USD, whereas GRU produces 7.860,62 USD for BXP compared to 9.549,83 USD for LSTM. These results highlight that lower prediction errors do not always translate into higher trading returns, due to the nonlinear nature of the trading strategy and stock-specific market dynamics.

Table 2: Final Portfolio Values (USD) for LSTM, GRU, and Buy & Hold Strategies

Stock	**LSTM**	**GRU**	**Buy & Hold**
MSCI	6.629,70	8.585,11	5.493,76
MCD	6.582,73	5.561,10	5.437,71
BA	4.237,05	4.518,66	4.223,45
PFE	4.331,81	4.155,98	3.591,61
INTC	4.223,45	3.819,12	3.187,24
MDLZ	5.029,15	5.139,31	4.408,57
AVY	5.696,15	5.576,40	5.529,34
BXP	9.549,83	7.860,62	7.666,35
ES	4.104,09	4.035,42	3.981,07
APA	3.877,38	3.675,84	3.236,52

The risk profiles of the trading strategies, measured by the 5% Historical Value at Risk (VaR), provide clear insight into their potential downside exposure. Across the ten stocks analyzed, both LSTM and GRU strategies consistently exhibit lower VaR values than the Buy & Hold benchmark, indicating a reduced likelihood of extreme daily losses. For instance, for MSCI, the Buy & Hold strategy has a VaR of -2.17%, while LSTM and GRU reduce this to -1.25% and -1.42%, respectively. Similar patterns are observed for other stocks, with volatile stocks such as INTC and APA showing substantial reductions in potential losses under model-based strategies compared to Buy & Hold. These results show that both deep-learning strategies not only generate substantial returns but also manage downside risk more effectively than passive investment approaches.

Table 3: 5% Historical Value at Risk (VaR) for LSTM, GRU, and Buy & Hold Strategies across 10 Stocks

Stock	**LSTM VaR**	**GRU VaR**	**Buy & Hold VaR**
MSCI	-0,0125	-0,0142	-0,0217
MCD	-0,0089	-0,0094	-0,0159
BA	-0,0220	-0,0208	-0,0302
PFE	-0,0180	-0,0194	-0,0227
INTC	-0,0263	-0,0255	-0,0439
MDLZ	-0,0060	-0,0090	-0,0178
AVY	-0,0128	-0,0144	-0,0194
BXP	-0,0271	-0,0270	-0,0328
ES	-0,0185	-0,0175	-0,0241
APA	-0,0279	-0,0247	-0,0388

In summary, the results show that accurate next-day price forecasting can produce profitable trading outcomes, with both LSTM and GRU offering an attractive combination of high returns and controlled risk. While GRU consistently achieves superior forecasting accuracy, LSTM frequently achieves higher trading returns for the majority of stocks. In terms of VaR, both strategies provide better downside protection than Buy & Hold.

5.3 Multi-Stock Portfolio Results

After analyzing the performance and risk of the deep-learning strategies across individual stocks, we also examine their behavior in a 10-stock portfolio, with an initial investment of 1.000 USD in each stock (totaling 10.000 USD).

Table 4: Multi-Stock Portfolio Performance and Risk

Metric	**LSTM**	**GRU**	**Buy & Hold**
Total Portfolio Return	-5,32%	-2,80%	-6,56%
Final Portfolio Value (USD)	9.468,36	9.719,77	9.343,64
5% Historical VaR	-0,0099	-0,0106	-0,0146

For the portfolio, the GRU strategy shows superior performance, achieving a final value of 9.719,77 USD, corresponding to a total return of -2,80%. The LSTM strategy ends at 9.468,36 USD (-5,32%), while the Buy & Hold benchmark declines to 9.343,64 USD (-6,56%). These results indicate that both predictive strategies help preserve capital more effectively than a passive approach, even in a declining market, with GRU providing the best overall outcome.

The 5% Historical VaR of the portfolio further highlights the downside risk profile. LSTM has a VaR of -0,99%, GRU -1,06%, and Buy & Hold -1,46%. This shows that both deep-learning strategies reduce the potential daily losses relative to Buy & Hold.

Overall, the portfolio results reinforce the single-stock findings. Deep-learning-based forecasts can generate profitable or capital-preserving trading outcomes, while simultaneously managing risk more effectively than passive investing. The GRU strategy, in particular, consistently balances forecasting accuracy, trading returns, and controlled downside risk, making it the most favorable approach for multi-stock portfolio management.

6 Conclusion and Future Directions

This study investigated the predictive performance of LSTM and GRU models for next-day stock price forecasting and evaluated the effectiveness of trading strategies derived from these forecasts. Using daily stock data from 2015 to 2024 for ten diverse stocks, we assessed both individual single-stock portfolios and a multi-stock portfolio, considering prediction accuracy, portfolio returns, and risk measured with the 5% Historical Value at Risk (VaR).

The results demonstrate that both LSTM and GRU models can capture medium-term trends and directional changes in stock prices, with GRU consistently achieving lower prediction errors across all ten stocks. While GRU provided superior forecasting accuracy, trading performance was not always directly correlated with lower errors. LSTM achieved higher final portfolio values for seven out of ten individual stocks. Both models, however, substantially outperformed the passive Buy & Hold strategy in terms of returns and downside risk management. In the multi-stock portfolio, GRU delivered the most favorable outcome, preserving capital better than LSTM and Buy & Hold, even in a declining market environment. The VaR analysis further confirmed that deep-learning-based strategies reduce potential daily losses compared to passive investment.

These findings highlight several key insights. First, accurate next-day price forecasts can translate into profitable or capital-preserving trading strategies. Second, the choice of model should consider not only prediction accuracy but also the interaction between forecasts and trading rules, as lower errors do not always guarantee higher returns. Third, both single-stock and multi-stock portfolio evaluations are essential to understand how predictive models perform under different investment scenarios, including diversification effects.

Future research could extend this work in several directions. Incorporating additional features, such as market sentiment or macroeconomic indicators, may further improve predictive performance. Exploring more advanced architectures, including attention-based mechanisms or Transformer models, could enhance the models' ability to capture sudden market movements and complex temporal dependencies. Additionally, portfolio optimization techniques could be applied to adjust capital allocations across stocks in order to maximize returns or minimize risk, providing a more practical and potentially profitable implementation of deep learning-based trading strategies.

7 Appendix

The full implementation for this work is available in the course repository on GitHub `https://github.com/HTW-WM-FAR/DeepLearningWS2025`. The implementation can be found in the folder DeepLearning_WS2025_EricChristopher.

8 Academic Integrity Declaration

I hereby confirm that I have authored this report independently and without use of others than the indicated sources. All passages which are literally or in general matter taken out of publications or other sources are marked as such.

All verbatim or in-sentence copies and quotations, as well as all sections that were designed, written and/or edited with the help of AI-based tools, have been identified and verified.

Information on the use of AI-based tools: In the appendix of my work, I have listed all AI-based resources with product names and prompts used. I assure that I have not used any AI-based tools whose use has been explicitly excluded in writing by the examiner. I understand that AI-generated content does not guarantee correctness. I remain fully responsible for the content of this work.

Eric Christopher

Berlin, 20.02.2026

9 References

[A2018] Auer, M. (2018). Hands On Value at Risk and Expected Shortfall: A Practical Primer. Springer International Publishing. Springer. eISBN 978-3-319-72320-4

[A2023] Aggarwal, C. (2023). Neural Network and Deep Learning. Springer. eISBN 978-3-031-29642-0

[B2023] Botsch, B. (2023). Maschinelles Lernen - Grundlagen und Anwendungen. Springer. eISBN 978-3-662-67277-8

[B2024] Bhasin, H. (2024). Hands-on Deep Learning. Apress. eISBN 978-8-8688-1035-0

[DTSKB2023] Das, S., Tariq, A., Santos, T., Kantareddy, S. S., Banerjee, I. (2023). Recurrent neural networks (RNNs): Architectures, training tricks, and introduction to influential research. In Machine learning for brain disorders (pp. 117-138). Humana Press. eISBN 978-1-0716-3195-9

[HMN2020] Houdt, G. V., Mosquera, C., Nápoles, G. (2020). A Review on the Long Short-Term Memory Model. In: Artificial Intelligence Review, pp. 5929-5955, Springer Nature B. V. eISSN 1573-7462

[L2013] Lorenzo, R. D. (2013). Basic Technical Analysis of Financial Markets. Springer. eISBN 978-88-470-5421-9

[M2021]Mak, D. K. (2021). Trading Tactics in the Financial Market. Springer. eISBN 978-3-030-70622-7

[S2021] Salem, F. M. (2021). Recurrent Neural Networks, Springer. eISBN 978-3-030-89929-5

[S2023] Schonlau, M. (2023). Applied Statistical Learning, Springer. eISBN 978-3-031-33390-3

[S2024] Selle, S. (2024). Data Science Training - Supervised Learning, Springer. eISBN 978-3-662-67960-9

[SSNG2020] Shahi, T. B., Shrestha, A., Neupane, A., Guo, W. (2020). Stock Price Forecasting with Deep Learning: A Comparative Study. `https://doi.org/10.3390/math8091441`

[VBBHL2025] Velarde, G., Branez, P., Bueno, A., Heredia, R., Lopez-Ledezma, M. (2025). An Open-Source and Reproducible Implementation of LSTM and GRU Networks for Time Series Forecasting. `https://doi.org/10.48550/arXiv.2504.18185`