



Università
Ca' Foscari
Venezia

Single Cycle Degree programme in
Economics, Finance, and Sustainability

Curricula:
Quantitative Finance and Risk Management

Final Thesis in Computational Finance Lab

“Portfolio optimization with a Machine Learning model:
The LSTM Network”

Supervisor

Ch. Prof. Antonella Basso

Graduand

Marco Parisi

Matriculation number 0905951

Academic Year

2025/2026

Dichiarazione del ricorso a strumenti di Intelligenza Artificiale (IA)

Il sottoscritto Marco Parisi, matricola numero 0905951, iscritto presso Ca' Foscari al corso di Laurea magistrale in Economics, Finance, and Sustainability, curricula Quantitative Finance and Risk Management, autore della tesi di laurea "Portfolio optimization with a Machine Learning model: The LSTM Network", relatrice Antonella Basso, consapevole delle sanzioni previste dall'art. 76 del Testo unico, D.P.R. 28/12/2000 n.445, e della decadenza dei benefici prevista dall'art. 75 del medesimo Testo unico in caso di dichiarazioni false o mendaci, sotto la propria personale responsabilità ai sensi degli artt. 46 e 47 del D.P.R. 28/12/2000 n. 445,

DICHIARA

di aver fatto ricorso a tecnologie di intelligenza artificiale generativa per supportare vari aspetti della ricerca e della scrittura della mia tesi di laurea, in particolare:

- "ChatGPT".
- Riformulare alcune frasi (per verificarne la correttezza grammaticale) che avevo già scritto, che poi ho comunque rivisto e rielaborato personalmente. Come supporto per la parte di programmazione per comprendere meglio la struttura di alcune funzioni.
- In tutti i capitoli ci sono alcune frasi riformulate.
- Nessun contenuto/informazione è stato generato da zero con l'IA, nessuna frase è stata scritta da zero con l'IA, nessuna fonte è stata cercata tramite l'IA e nessuno script è stato generato dall'IA.

DICHIARA INOLTRE

che tutto il materiale generato dall'IA è stato attentamente verificato, rivisto e rielaborato dall'autore/dall'autrice, che si assume la piena responsabilità per la qualità, l'accuratezza e l'integrità scientifica del lavoro.

INDEX

INTRODUCTION	1
CHAPTER 1 - Traditional portfolio optimization model	3
1.1. <i>Markowitz model: Modern Portfolio Theory</i>	3
1.1.1. <i>Assumption of the Mean-Variance framework</i>	3
1.1.2. <i>Mathematical formulation of the Mean–Variance framework</i>	4
1.1.3. <i>Diversification and role of covariance</i>	6
1.1.4. <i>Efficient Frontier and Capital Allocation Line</i>	7
1.1.5. <i>The separation theorem</i>	10
1.1.6. <i>Risk, risk aversion and expected utility</i>	11
1.2. <i>Advantages and limitations of the Markowitz model</i>	13
1.2.1. <i>Advantages</i>	13
1.2.2. <i>Limitations</i>	14
CHAPTER 2 - New portfolio optimization models	16
2.1. <i>Introduction to: Artificial Intelligence approaches</i>	16
2.2. <i>Reinforcement Learning</i>	16
2.2.1. <i>Reinforcement Learning methods</i>	17
2.2.2. <i>Reinforcement Learning workflow</i>	18
2.3. <i>Artificial Neural Networks</i>	20
2.3.1. <i>Performance measures of Artificial Neural Networks</i>	21
2.4. <i>Deep Learning</i>	22
2.4.1. <i>Structure of Deep Learning models</i>	23
2.5. <i>Advantages and limitations</i>	24
2.5.1. <i>Reinforcement Learning</i>	24
2.5.2. <i>Artificial Neural Network</i>	25
2.5.3. <i>Deep Learning</i>	27
2.6. <i>Synthesis and implications for portfolio optimization</i>	28
CHAPTER 3 – Deep Learning architectures for financial time series forecasting	29
3.1. <i>Motivation for the use of Deep Learning models</i>	29
3.2. <i>Deep Learning models versus Traditional Forecasting models</i>	29
3.3. <i>Convolutional Neural Networks (CNNs)</i>	31
3.3.1. <i>CNN structure</i>	32
3.3.2. <i>Image recognition models with financial datasets</i>	33
3.4. <i>Long Short-Term Memory (LSTM)</i>	34
3.4.1. <i>LSTM architecture</i>	36
3.4.2. <i>LSTM framework for portfolio optimization</i>	38

CHAPTER 4 – Practical application of the LSTM model for returns

prediction	43
4.1. <i>Construction of the data set</i>	43
4.2. <i>Input features used in the model</i>	45
4.3. <i>Construction of the LSTM model used</i>	50
4.4. <i>Model's parameters</i>	54
4.5. <i>Results of the LSTM model.....</i>	57
4.6. <i>Robustness analysis of the reference-stock selection.....</i>	62

CHAPTER 5 – Portfolio Optimization and Portfolio Back-Test frameworks

and results.....	65
5.1. <i>Portfolio Optimization frameworks.....</i>	65
5.1.1. <i>Efficient Frontiers construction</i>	66
5.1.2. <i>Selection of the optimal portfolios</i>	67
5.1.3. <i>Portfolio optimization results.....</i>	68
5.1.4. <i>Overall results and conclusion on the Portfolio Optimization</i>	73
5.2. <i>Out-of-sample portfolio Back-Test.....</i>	73
5.2.1. <i>Back-Test framework</i>	74
5.2.2. <i>Back-Test functioning</i>	75
5.2.3. <i>Portfolios back-test performance measures.....</i>	77
5.2.4. <i>Back-Test performance results.....</i>	78
5.2.5. <i>Overall results and conclusion on the out-of-sample back-test.....</i>	84

CONCLUSION	85
-------------------------	-----------

BIBLIOGRAPHY	86
---------------------------	-----------

MATLAB SCRIPT AND DATA-SET	90
---	-----------

INTRODUCTION

Portfolio optimization is one of the main problems in modern finance. This because the investors aim to obtain the best possible trade-off between expected return and risk.

The traditional approach introduced by Markowitz provides a rigorous framework for portfolio construction by considering expected returns, variances, and covariances among assets; that will be analysed in the first chapter. Despite it is one of the more used method, it has some limitations, the most important is its strong dependence on historical estimates, particularly for the expected returns, which may not adequately represent future market conditions.

In recent years, Machine Learning and Deep Learning techniques have provided alternative methods for analysing financial data and for identifying complex and non-linear relationships. Among these models, Long Short-Term Memory (LSTM) networks are particularly suitable for financial time series because they are designed to process sequential data and capture temporal dependencies.

The objective of this thesis is to investigate whether LSTM-generated return forecasts can improve portfolio optimization when incorporated into a traditional Markowitz mean–variance framework. The analysis is conducted on 45 stocks from the EURO STOXX 50, using daily adjusted prices and trading volumes from January 2015 to May 2026. The LSTM model predicts cumulative three-day logarithmic returns using price, momentum, volume, and volatility-based features.

The forecasts are then used to construct two LSTM-based portfolios: a complete portfolio containing all the securities and a filtered portfolio including only stocks for which the model achieves a Directional Accuracy above 50% and a positive Correlation coefficient between predicted and realized returns. These strategies are compared with a traditional historical Markowitz portfolio, an equally weighted portfolio, and the EURO STOXX 50 benchmark.

The analysis evaluates both the predictive performance of the LSTM model and its economic usefulness through portfolio optimization and an out-of-sample back-test including transaction costs. In this way, the thesis investigates whether Machine Learning can complement traditional portfolio theory by providing more forward-looking information for investment decisions.

The thesis is organized into five chapters, following a progression from the theoretical foundations of the portfolio optimization to the empirical analysis.

Chapter 1 presents the traditional Markowitz mean–variance framework and discusses its main assumptions, advantages, and limitations.

Chapter 2 introduces alternative portfolio optimization approaches based on Artificial Intelligence, with particular attention to Reinforcement Learning, Artificial Neural Networks, and Deep Learning.

Chapter 3 focuses more specifically on Deep Learning architectures for financial time-series forecasting, and introduces the LSTM networks and its application to portfolio optimization.

Chapter 4 describes the empirical construction and implementation of the LSTM model, including the construction of the dataset, the input features, the model architecture, the selected parameters, and how predictive performances of the model are evaluated.

And finally, Chapter 5 applies the resulting forecasts to the portfolio optimization and evaluates the different investment strategies through an out-of-sample back-test, comparing the LSTM-based strategies with the historical Markowitz portfolio, the equally weighted portfolio, and the EURO STOXX 50 benchmark.

CHAPTER 1 - Traditional portfolio optimization model

1.1. *Markowitz model: Modern Portfolio Theory*

Harry Markowitz introduced Modern Portfolio Theory (MPT), also known as the mean–variance framework, in his seminal 1952 paper (Markowitz, 1952), for which he received the Nobel Memorial Prize in Economic Sciences. Prior to his contribution, investment decisions were largely driven by heuristic and non-formalized approaches, lacking a rigorous mathematical foundation. Markowitz's work established the basis for modern portfolio construction, and his framework remains one of the most influential methodologies in finance up to this day (Boyd, Johansson, Kahn, Schiele, & Schmelzer, 2024).

Modern Portfolio Theory provides a formal mathematical framework for selecting portfolios by optimizing the trade-off between risk and return. In fact, it allows investors to either maximize expected return for a given level of risk or doing the opposite, minimize risk for a given expected return. The core intuition of the model lies on the reduction of portfolio risk through the diversification, which arises from the imperfect correlation structure among asset returns. In this context, diversification enables investors to construct portfolios that achieve the same expected return with a lower level of overall risk by modifying the weights of the assets.

Within the Markowitz framework, risk is measured by the variance (or equivalently by the standard deviation) of portfolio returns, as this metric allows for a tractable analytical treatment when combining multiple assets. The model assumes that the market is formed by investors that are rational and risk-averse and therefore prefer portfolios that offer the highest expected return for a given level of risk (Michuad, 1989).

1.1.1. *Assumption of the Mean-Variance framework*

The Markowitz Mean–Variance framework is based on a set of assumptions that simplifies the real world to allow the portfolio optimization problem to be expressed in a tractable mathematical form. These assumptions are essential to derive a closed-form solution and to express the portfolio choice problem in a quadratic optimization setting. However, these assumptions represent a key source of limitations when the model is applied in real-world financial markets (Markowitz, 1989).

These assumptions are:

- The model assumes that investors are mean–variance optimizers, meaning that they make decisions based solely on the expected return and variance of portfolio returns. This assumption significantly simplifies the decision-making process, as it reduces the evaluation of investment opportunities to a two-dimensional trade-off. However, it implies that some moments of the return distribution, such as skewness and kurtosis, are ignored. In practice,

investors may exhibit preferences for asymmetric return distributions, which are not captured by the model (Campbell, Harvey, & Akhtar, 2020);

- Asset returns are typically assumed to be normally distributed or, more generally, fully described by their first two moments (the mean and the variance). This assumption simplifies the optimization process because it justifies the use of variance as a complete measure of risk. Nevertheless, empirical evidence shows that financial returns often exhibit fat tails and skewness, meaning that extreme events occur more frequently than predicted by the normal distribution. So, the model tends to underestimate the probability and impact of large losses, leading to an underestimation of risk in real-world applications (Campbell, Harvey, & Akhtar, 2020);
- The model assumes that all input parameters (expected returns, variances, and covariances) are known with certainty. Normally, these quantities must be estimated from historical data, introducing estimation error and uncertainty into the optimization process that could lead to different final result of the optimization process. For example, small estimation errors in expected returns may lead to large variations in portfolio weights, a phenomenon commonly known as the estimation error sensitivity;
- Presence of frictionless markets, where there are no transaction costs, taxes, or liquidity constraints. This assumption allows for continuous portfolio rebalancing without costs, which is unrealistic in practical investment settings (Yashaswi, 2021);
- The model assumes that investors can always borrow and lend at a risk-free rate and, in many formulations, can take unrestricted short positions. These assumptions simplify the construction of the Capital Allocation Line and the identification of the tangency portfolio. These conditions may not hold in real markets due to regulatory and institutional constraints, making difficult the application of the theoretical results;
- The Markowitz framework is inherently static and based on a single-period horizon. This assumption allows for a simplified optimization problem but fails to capture the dynamic nature of financial markets, where investment opportunities and risk profiles evolve over time. In practice, investors must continuously update their portfolios in response to new information and to their new risk profile preferences, making dynamic models more appropriate in many contexts.

Overall, while these assumptions are necessary to ensure analytical tractability and to derive a clear theoretical framework, they also explain many of the limitations of the model.

1.1.2. *Mathematical formulation of the Mean–Variance framework*

The mean–variance framework introduced by Harry Markowitz provides a formal mathematical

representation of the portfolio selection problem. Within this setting, a portfolio is defined by a vector of weights $w = (w_1, w_2, \dots, w_n)$, where each weight represents the proportion of total amount invested in each asset, considering the budget constraint that the weights' sum must be equal to one. The expected return of the portfolio is given by the weighted average of the expected returns of the single assets (eq. 1) (Boyd, Johansson, Kahn, Schiele, & Schmelzer, 2024).

$$E(R_p) = \sum_{i=1}^n w_i E(R_i) \quad (1)$$

where R_p is the return of the portfolio, R_i is the return of each single asset and w_i is the corresponding weight of the asset inside the portfolio.

This linear relationship implies that the expected return of the portfolio ($E(R_p)$) is simply the aggregation of individual expected returns ($E(R_i)$), weighted by their respective portfolio weights (w_i).

In contrast, the variance of the portfolio, which serves as a measure of risk, depends not only on the individual variances of the assets but also on the covariance structure among them. The formula of the variance of the portfolio is not a linear relationship of the variance of all the assets (eq. 2).

$$\sigma_p^2 = \sum_{i=1}^n \sum_{j=1}^n w_i w_j \sigma_{ij} \quad (2)$$

formula 2 highlights that the portfolio risk is driven by the joint behavior of asset returns. In particular, the covariance terms capture how assets co-move, playing a crucial role in determining the effectiveness of diversification. The variance of the portfolio can be expressed in a more compact way (eq. 3) by using the variance-covariance matrix (Σ), this formulation is widely used in numerical optimization methods (Boyd, Johansson, Kahn, Schiele, & Schmelzer, 2024)

$$\sigma_p^2 = w^T \Sigma w. \quad (3)$$

The portfolio selection problem can be formulated as a constrained optimization problem. In its standard form, the investor seeks to minimize the portfolio variance for a given level of expected return (eq. 4).

$$\begin{aligned} & \min_w w^T \Sigma w ; \\ & \text{s. t.} \quad w^T \mu = \mu_p ; \\ & \quad \sum_{i=1}^n w_i = 1 \end{aligned} \quad (4)$$

where μ is the vector of expected returns and μ_p is the target portfolio return.

The problem can be, also, expressed in its dual form, where the investor maximizes the expected return subject to a constraint on portfolio risk. These two formulations are equivalent and jointly define the set of optimal portfolios that constitute the efficient frontier.

This quadratic optimization structure represents one of the key innovations of the Markowitz framework, as it provides a tractable and systematic method for portfolio construction, while

explicitly incorporating the role of diversification through the covariance matrix (Boyd, Johansson, Kahn, Schiele, & Schmelzer, 2024).

1.1.3. *Diversification and role of covariance*

Within the mean–variance framework introduced by Harry Markowitz, portfolio risk can be decomposed into two distinct components: systematic risk and idiosyncratic (firm-specific) risk. Systematic risk, also known as non-diversifiable risk, arises from common factors that affect the entire market, like macroeconomic conditions, interest rate changes, or geopolitical events, in fact it cannot be eliminated through diversification.

In contrast, idiosyncratic risk is specific to individual firms and is driven by factors such as management decisions, operational performance, or firm-level shocks. Unlike systematic risk, this component can be effectively reduced by holding a sufficiently diversified portfolio of assets.

As the number of assets included in a portfolio increases, the contribution of idiosyncratic risk to total portfolio variance progressively declines. This effect is illustrated in fig. 1, where the portfolio standard deviation decreases with the number of securities held, eventually converging to a lower bound determined by the systematic risk. This asymptotic level represents the portion of total risk that remains even in highly diversified portfolios (Bodie, Kane, & Marcus, 2013).

However, the effectiveness of diversification does not depend solely on the number of assets included, but rather on the correlation structure among their returns. A portfolio composed of securities belonging to the same industry may exhibit high levels of co-movement, this will limit the benefits of diversification. Conversely, combining assets from different sectors or economic environments, which are characterized by lower covariance, allows for a more substantial reduction in overall portfolio risk (Markowitz, 1952).

The role of correlation is central in determining the extent of diversification benefits. In the special case where asset returns are uncorrelated ($\rho = 0$), the portfolio variance decreases as the number of assets increases, approaching the lower bound. When correlations are strictly positive ($\rho > 0$), diversification still reduces the risk but cannot eliminate it entirely. In the extreme case of perfect positive correlation ($\rho = 1$), all assets move together in the same way, and diversification does not produce any positive effect on risk reduction, as all risk effectively becomes systematic.

Conversely, when asset returns are negatively correlated ($\rho < 0$), diversification benefits are significantly enhanced. In the theoretical case of perfect negative correlation ($\rho = -1$), it is possible to construct a portfolio with zero variance, fully eliminating risk. Although such conditions are rarely observed in practice, they illustrate the fundamental importance of correlation in portfolio construction.

Under the simplifying assumption of assets with equal variance and constant pairwise correlation, the

portfolio variance can be expressed as shown in eq. (5).

$$\sigma_p^2 = \frac{1}{n}\sigma^2 + \frac{n-1}{n}\rho\sigma^2 \quad (5)$$

this expression clearly shows that as the number of assets n increases, the first term (the idiosyncratic risk) tends to zero, while the second term (the systematic component) converges to a positive value determined by the average correlation.

Consequently, in a well-diversified portfolio, the contribution of an individual asset to total portfolio risk depends primarily on its covariance with the other assets, rather than on its standalone variance. This insight represents one of the most important implications of Modern Portfolio Theory and provides the theoretical foundation for risk-based portfolio construction (Bodie, Kane, & Marcus, 2013).

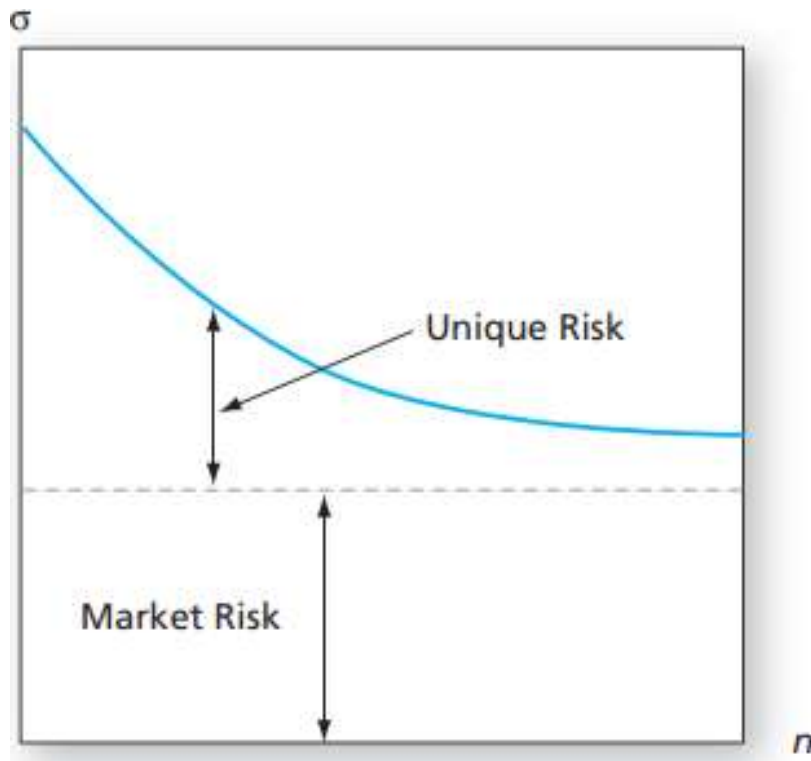


Figure 1: Relationship between the number of assets in a portfolio (n) and the total portfolio risk, measured by the standard deviation (σ). As the number of assets increases, portfolio risk decreases due to diversification effects, reflecting the reduction of idiosyncratic risk. However, this reduction occurs at a decreasing rate and converges to a lower bound, which represents systematic (market) risk. This residual level of risk cannot be eliminated through diversification (source: (Bodie, Kane, & Marcus, 2013)).

1.1.4. *Efficient Frontier and Capital Allocation Line*

Within the mean–variance framework, the investor faces a wide set of feasible portfolio combinations resulting from different allocations across risky assets. Each portfolio is characterized by a specific expected return, and level of risk which is measured by the standard deviation or the variance. The set of all the portfolios that lie on the Minimum-Variance Frontier (fig. 2) are those one that are

providing for each level of expected return, for all the combinations of all the assets, the lowest possible level of risk.

However, not all portfolios lying on the Minimum-Variance Frontier are considered efficient. Along this frontier, it is possible to observe different portfolios that share the same level of risk but offer different expected returns (fig. 2). In such cases, rational investors will choose the portfolio with the highest expected return for the same level of risk. The set of portfolios that maximizes expected return for each level of risk forms the Efficient Frontier, which corresponds to the upper segment of the Minimum-Variance Frontier (Bodie, Kane, & Marcus, 2013).

A portfolio is said to dominate another if it provides a higher expected return for the same level of risk, or a lower level of risk for the same expected return. In fact, the portfolios that lie on the Efficient Frontier dominate all the others.

The expected return of the portfolio is a linear function of portfolio weights and the corresponding expected returns of all the assets (eq. 1). The portfolio risk depends on the covariance structure among assets, and it is computed by using the variance-covariance matrix (Σ), eq. (3) (Markowitz, 1952).

Among all feasible portfolios, the one with the lowest possible variance is known as the Global Minimum-Variance Portfolio (GMVP), which represents the leftmost point on the frontier (fig. 2). All portfolios lying below the efficient frontier are suboptimal, because there always exists an alternative portfolio with a higher expected return for the same level of risk or lower risk for the same expected return.

An important implication of this framework is that individual assets are generally inefficient when considered in isolation, as diversification allows for the construction of portfolios with superior risk–return trade-offs (Bodie, Kane, & Marcus, 2013).

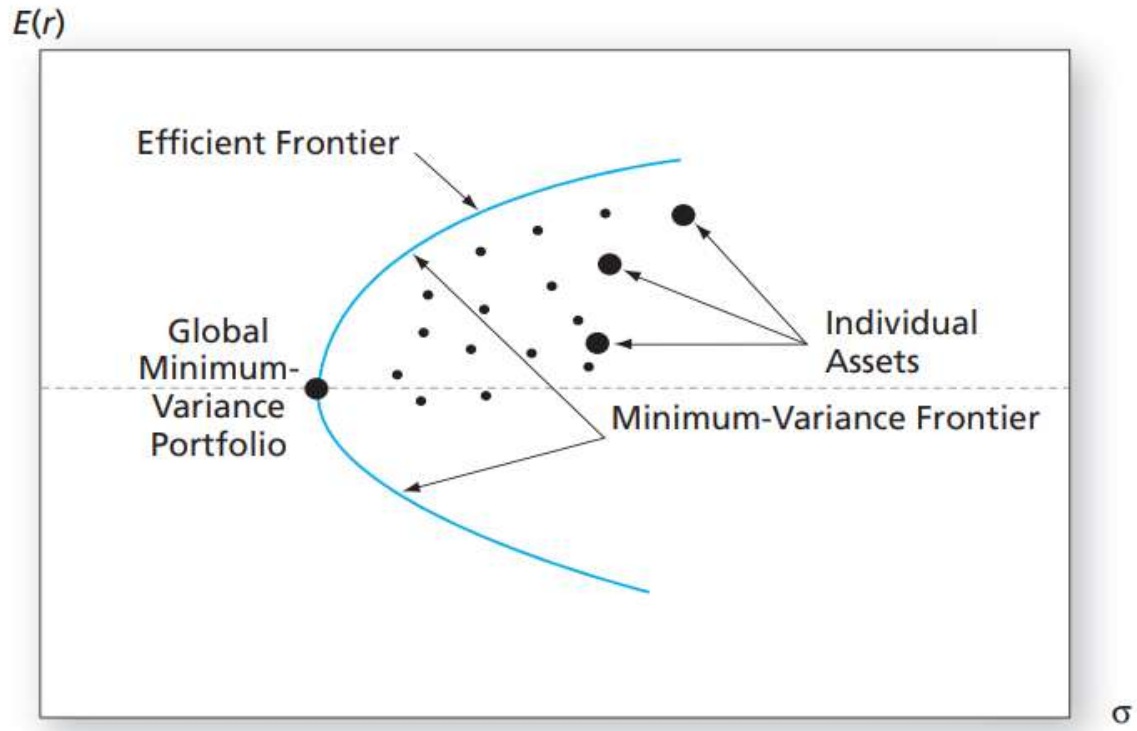


Figure 2: Minimum-Variance Frontier and Efficient Frontier in the mean–variance space. Relationship between portfolio risk (standard deviation) and expected return. The Global Minimum-Variance Portfolio (GMVP) corresponds to the portfolio with the lowest achievable risk (source: (Bodie, Kane, & Marcus, 2013)).

When the investment opportunity set includes only risky assets, the optimal portfolio selection is restricted to portfolios lying on the Efficient Frontier, the final choice among these portfolios depends on the investor's preferences, typically represented through a utility function.

The introduction of a risk-free asset fundamentally changes the portfolio selection problem. In this case, investors can combine the risk-free asset with any portfolio of risky assets, leading to a new set of feasible portfolios described by the Capital Allocation Line (CAL). The CAL represents all possible linear combinations between the risk-free asset and a given risky portfolio. Its slope corresponds to the Sharpe ratio, which measures the excess return per unit of risk, that is given by eq. (6), that is:

$$\text{Sharpe Ratio} = \frac{E(R_p) - R_f}{\sigma_p} . \quad (6)$$

Where $E(R_p)$ is the expected return of the portfolio, R_f is the return of the risk-free asset and σ_p is the standard deviation of the portfolio.

Among all possible risky portfolios of the Efficient Frontier, there exists one that maximizes the Sharpe ratio. This portfolio, known as the tangency portfolio, is located at the point where the CAL is tangent to the Efficient Frontier, it represents the optimal risky portfolio, as it offers the highest reward-to-risk ratio (fig. 3).

The CAL associated with the tangency portfolio dominates all other feasible allocation lines, as it

provides the highest expected return for any given level of risk, all investors, in fact regardless of their risk preferences, will choose to hold a combination of the risk-free asset and this optimal risky portfolio (the tangency one).

The final allocation decision then consists in selecting the appropriate mix between the risk-free asset and the tangency portfolio. More risk-averse investors will allocate a larger proportion of their wealth in the risk-free asset, while less risk-averse investors will take on greater exposure in the risky portfolio.

This separation between portfolio selection and risk preference is a key implication of Modern Portfolio Theory and is called the separation theorem (Bodie, Kane, & Marcus, 2013).

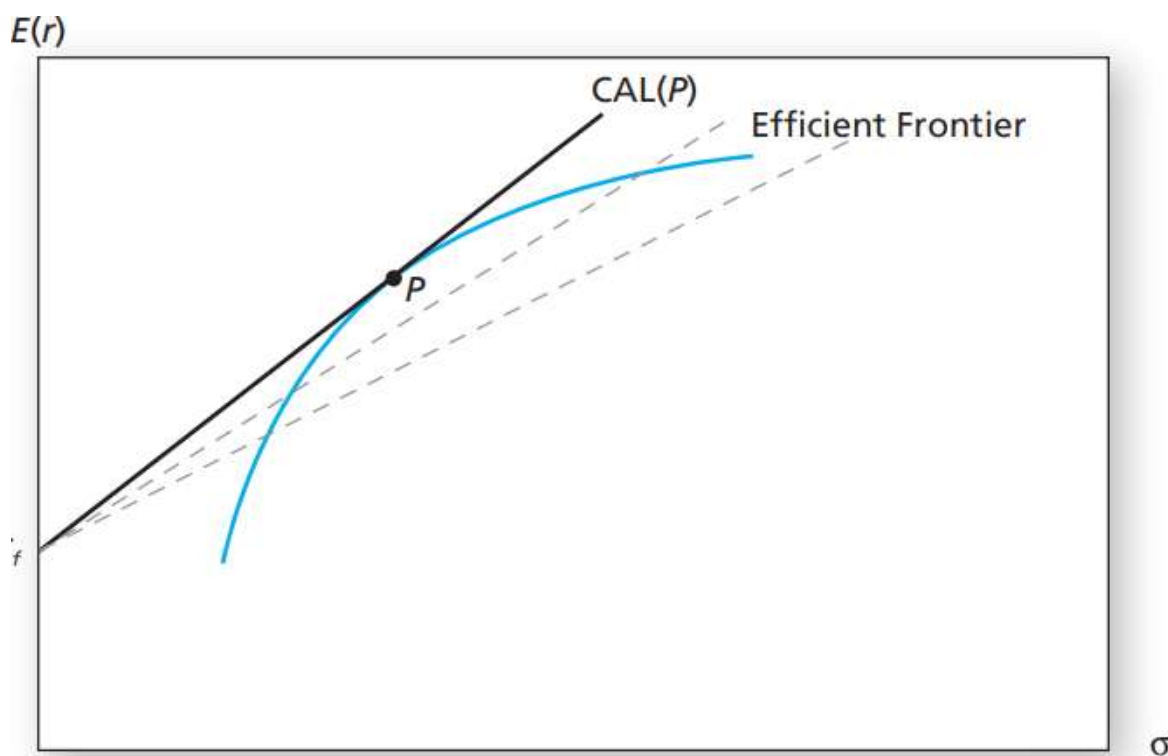


Figure 3: Efficient Frontier with the Capital Allocation Line (CAL). The straight line originates from the risk-free rate and is tangent to the efficient frontier at point P , representing the optimal risky (tangency) portfolio. The slope of the CAL corresponds to the Sharpe ratio. Portfolios along the CAL are obtained by combining the risk-free asset with the tangency portfolio, allowing investors to achieve different risk–return profiles depending on their level of risk aversion (source: (Bodie, Kane, & Marcus, 2013)).

1.1.5. *The separation theorem*

When the investment opportunity set includes both risky assets and a risk-free asset, the portfolio choice is governed by the separation theorem, one of the key implications of the mean–variance framework introduced by Harry Markowitz. This principle states that the portfolio selection problem can be decomposed into two distinct and independent decisions.

The first decision concerns the identification of the optimal risky portfolio. This step is purely technical and depends exclusively on the statistical properties of asset returns, that are: expected returns, variances, and covariances. The resulting portfolio corresponds to the tangency portfolio, which maximizes the Sharpe ratio. Importantly, this portfolio is the same for all investors, regardless of their individual risk preferences.

The second decision involves the allocation of wealth between the optimal risky portfolio and the risk-free asset. Unlike the first step, this choice depends on the investor's degree of risk aversion and is therefore subjective, each investor selects a specific point along the Capital Allocation Line (CAL), determining the final risk–return profile of the overall portfolio.

A key implication of the separation theorem is that all investors will hold the same optimal risky portfolio, differing only in the proportion of wealth allocated between this portfolio and the risk-free asset. More risk-averse investors will allocate a larger share to the risk-free asset, while less risk-averse investors will choose higher exposure to the risky portfolio.

This result has important practical consequences for portfolio management. In particular, it implies that asset management can be organized efficiently, as a single optimal risky portfolio can be offered to all clients, while customization is achieved through the capital allocation decision. This significantly reduces complexity and allows financial institutions to serve a large number of investors with relatively low incremental costs (Bodie, Kane, & Marcus, 2013).

1.1.6. *Risk, risk aversion and expected utility*

A fundamental concept in portfolio theory is the relationship between risk and return, and the way in which investors evaluate this trade-off. A commonly used definition of speculation describes it as the behavior of taking a significant investment risk in order to obtain a commensurate gain, that is, a sufficiently high expected return to compensate for the risk undertaken (Bodie, Kane, & Marcus, 2013).

From an economic perspective, a distinction can be made between speculation and gambling, while both involve risk. Speculation is undertaken with the expectation of a positive risk premium, whereas gambling typically involves risk without adequate compensation. A risk-averse investor will therefore avoid investments that do not offer a sufficient expected return relative to their risk, even if the expected return is positive. In particular, investments with zero risk premium, often called fair games, are generally rejected by risk-averse individuals.

Within the mean–variance framework introduced by Harry Markowitz, investor preferences are modeled through a utility function, which assigns a numerical value to each portfolio based on its expected return and risk. This approach allows investors to rank alternative portfolios and select the one that maximizes their level of satisfaction.

For risk-averse investors, the utility is typically represented as a function that penalizes risk. In its standard quadratic form, the utility of a portfolio is given by eq. (7).

$$U = E(r) - \frac{1}{2}A\sigma^2 \quad (7)$$

where U denotes the utility level, $E(R_p)$ is the expected return of the portfolio, σ_p^2 represents its variance, and A is the coefficient of risk aversion. A higher value of A indicates a stronger aversion for risk, leading to a greater penalty applied to the portfolio variance.

This formulation implies that investors evaluate portfolios by balancing expected return against risk. Portfolios with higher expected returns increase utility, while a higher volatility reduces it. In the case of a risk-free asset, the utility coincides with the expected return, as no risk penalty is applied ($\sigma = 0$) (Bodie, Kane, & Marcus, 2013).

Investors select the portfolio that maximizes their utility, subject to the set of feasible portfolios. Graphically, this corresponds to the point at which the highest attainable indifference curve (or utility curve) is tangent to the efficient frontier.

Different levels of risk aversion lead to different optimal choices. More risk-averse investors will select portfolios with lower variance and lower expected return, while less risk-averse investors will choose portfolios with higher expected return and higher risk. This relationship is illustrated in fig. 4, where each utility curve represents a different level of risk aversion, and the optimal portfolio is determined by the tangency point with the Efficient Frontier (Michuad, 1989).

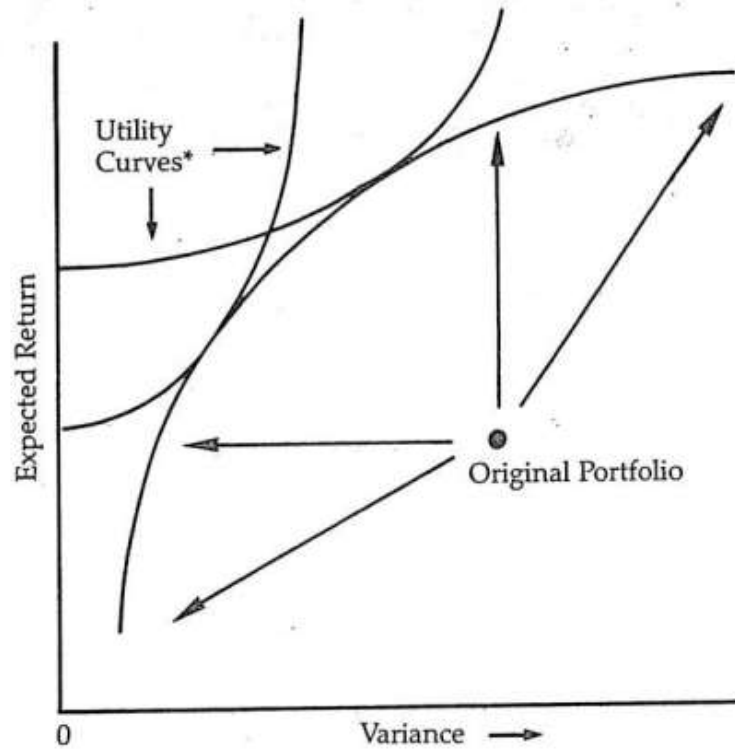


Figure 4: Efficient Frontier and investor utility curves (indifference curves, corresponding to a specific degree of risk aversion). The optimal portfolio for each investor is determined by the tangency point between the highest attainable utility curve and the efficient frontier (source: (Michuad, 1989)).

1.2. *Advantages and limitations of the Markowitz model*

1.2.1. *Advantages*

The Markowitz mean–variance model represents a fundamental framework for portfolio selection, offering several important advantages in both theoretical and practical applications. They are:

- The model allows for the formal incorporation of investor objectives and constraints. Portfolio managers can input specific requirements, such as risk tolerance, return targets, and regulatory or institutional constraints, thereby ensuring that the resulting portfolio is aligned with the investor's preferences.
- The model provides a structured approach to control portfolio risk exposure. By explicitly modeling the variance and covariance of asset returns, the optimizer enables a detailed analysis of how different sources of risk contribute to overall portfolio volatility.
- The Markowitz framework facilitates the implementation of investment style, strategic views, and market expectations. Managers can incorporate their outlook on expected returns and correlations, thereby influencing the composition of the optimal portfolio and its exposure to different risk factors.
- Efficient use of available information. The model can process large sets of financial data, combining expected returns, variances, and covariances into a single optimization problem

that identifies the most efficient allocation of capital.

- Finally, the computational nature of the model allows for rapid portfolio adjustments. As new information becomes available, the optimizer can quickly recompute optimal allocations, enabling timely and informed decision-making in dynamic market environments (Michuad & O, 1989).

1.2.2. *Limitations*

Despite its theoretical elegance, the Markowitz model is subject to several well-known limitations that significantly affect its practical implementation. They are:

- The sensitivity of the model to input assumptions. The optimal portfolio depends heavily on estimated expected returns, variances, and covariances. Even small changes in these inputs, such as variations in the estimation window or small adjustments in risk aversion, can lead to substantial differences in the resulting portfolio. This phenomenon is commonly known as output instability.
- Closely related to the previous issue is the problem of error maximization. Because the optimization process tends to favor assets with high expected returns, low variance, and favorable correlations, any estimation error in these inputs is amplified. As a result, the model may assign excessive weights to assets whose characteristics are overestimated, leading to poorly diversified and potentially unrealistic portfolios.
- The omission of relevant real-world constraints. In particular, the standard Markowitz framework often neglects liquidity considerations. However, liquidity plays a crucial role in portfolio management, especially for risk-averse investors who may require a portion of their wealth to remain easily accessible in periods of market stress. Ignoring liquidity constraints may lead to portfolios that are optimal in theory but impractical in real market conditions.
- The model may fail to adequately reflect organizational or strategic inconsistencies. If portfolio inputs, such as investment style or market outlook, are not updated consistently, the resulting allocation may diverge from the intended investment strategy, creating a mismatch between objectives and outcomes.
- The non-uniqueness and statistical uncertainty of the optimal solution. While the optimization process produces a single portfolio as the optimal outcome, this result is based on estimated parameters subject to statistical error. In reality, there exists a range of portfolios that are statistically indistinguishable in terms of expected return and risk. It is therefore more appropriate to think in terms of a confidence region of optimal portfolios, rather than a single deterministic solution.

These limitations highlight an important weakness of the Markowitz model: its strong dependence on

precise parameter estimation. In practical settings, where financial data are noisy and unstable, this reliance can significantly reduce the robustness and reliability of the model's output (Michuad, 1989). The limitations outlined above (particularly estimation error, instability, and sensitivity to input assumptions) have motivated the development of alternative approaches to portfolio optimization. In recent years, machine learning techniques, including deep learning models such as Long Short-Term Memory (LSTM) networks, have been proposed to improve the estimation of expected returns and capture complex, non-linear patterns in financial data. These approaches aim to enhance the robustness of portfolio construction by addressing some of the key weaknesses of the traditional mean–variance framework.

CHAPTER 2 - New portfolio optimization models

2.1. *Introduction to: Artificial Intelligence approaches*

In recent years, Artificial Intelligence (AI) techniques have gained increasing relevance in financial applications, particularly in portfolio optimization and asset allocation. These approaches aim to extract patterns and predictive signals from large and complex datasets, overcoming some of the limitations of traditional model-based frameworks.

AI encompasses a broad range of methodologies, among which machine learning techniques play a central role. Within this context, three major approaches have received particular attention, that are: Reinforcement Learning (RL), Artificial Neural Networks (ANNs), and Deep Learning.

Reinforcement Learning is a class of machine learning algorithms in which an agent learns to make decisions through repeated interactions with an environment. The learning process is driven by a system of rewards and penalties, allowing the model to improve its performance over time by optimizing a given objective function, this approach is particularly suitable for sequential decision-making problems, such as dynamic portfolio allocation (Sood, Papasotiriou, Vaiciulis, & Balch, 2023).

Artificial Neural Networks (ANNs) are computational models inspired by the structure and functioning of the human brain. They consist of interconnected nodes (neurons) organized in layers, which process information and learn complex relationships between inputs and outputs, ANNs are especially effective in capturing non-linear patterns in financial data. These data are often not adequately modeled by traditional linear approaches (Kufel, et al., 2023).

Deep Learning represents an advanced class of neural network models characterized by multiple hidden layers, enabling the extraction of hierarchical features from data. By learning representations at different levels of abstraction, deep learning models are capable of identifying complex structures and dependencies in financial time series, this ability makes them particularly suitable for tasks such as return prediction and risk modelling (Goodfellow, Bengio, & Courville, 2016).

A key element underlying all AI approaches is the availability of data. Financial markets generate large volumes of high-frequency and heterogeneous data, which can be used to train models and improve predictive accuracy. Unlike traditional models, which rely on strong assumptions about data distributions, AI methods are typically data-driven and can adapt to complex and evolving market dynamics.

2.2. *Reinforcement Learning*

Reinforcement Learning (RL) is a class of machine learning techniques in which an agent learns to perform a task through repeated interactions with an environment. The objective of the agent is to

identify an optimal decision-making strategy, known as a policy, which maps observed states of the environment to actions in order to maximize a cumulative measure of reward.

At the beginning of the learning process, the agent typically explores the environment by taking random or suboptimal actions, through a trial-and-error mechanism it gradually learns to associate observations with actions and their corresponding outcomes. This process allows the agent to improve its policy over time, balancing exploration of new actions with exploitation of previously learned strategies.

The reinforcement learning framework can be explained through a simple example, such as autonomous vehicle parking. In this case, the agent corresponds to the control algorithm, the environment includes all external factors (such as vehicle dynamics, obstacles, and weather conditions), and the observations are provided by sensors such as cameras or GPS. The agent selects actions (steering or braking), receives feedback in the form of rewards, and iteratively refines its policy until it can successfully complete the task.

More generally, reinforcement learning problems are formalized using the Markov Decision Process (MDP), which provides a mathematical framework for sequential decision-making under uncertainty. An MDP is defined by (S, A, P, R, γ) , where S is the set of possible states, A is the set of available actions, $P = P(s_{t+1} = s' \mid s_t = s, a_t = a)$ represents the transition probability from state s to state s' given action a , $R(s, a, s')$ denotes the reward associated with the transition, and $\gamma \in [0,1]$ is the discount factor that determines the relative importance of future rewards (MathWorks).

The solution to a reinforcement learning problem consists in identifying an optimal policy π that maximizes the expected cumulative discounted reward over time, this maximization function is given by eq. (8), where the expectation is taken over the sequence of states and actions generated by the policy.

$$\mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \right] \quad (8)$$

In financial applications, reinforcement learning has been increasingly used for dynamic portfolio allocation and trading strategies, where the agent sequentially adjusts portfolio weights in response to changing market conditions. The ability of RL models to learn from interaction and adapt over time makes them particularly suitable for problems characterized by uncertainty, feedback, and sequential decision-making (Sood, Papasotiriou, Vaiciulis, & Balch, 2023).

2.2.1. Reinforcement Learning methods

Reinforcement Learning (RL) methods can be broadly classified into three main categories: Dynamic Programming (DP), Monte Carlo (MC) methods, and Temporal Difference (TD) learning. These approaches differ in the way they estimate value functions and improve policies, as well as in their

requirements regarding the knowledge of the environment (Andrew, 1999).

Dynamic Programming (DP) methods constitute a class of algorithms that compute the optimal policy assuming full knowledge of the environment's dynamics, requiring access to the transition probabilities and reward function, which allows the agent to evaluate all possible state–action pairs. The key idea underlying DP is the recursive decomposition of the value function through Bellman equations, enabling an iterative improvement of policies by updating value estimates in a systematic and exhaustive manner. DP methods are often impractical in real-world applications due to the need for a complete model of the environment.

Monte Carlo (MC) methods estimate value functions by averaging the returns obtained from complete sampled episodes. In contrast to DP methods, MC can estimate value functions without requiring an explicit knowledge of the environment's transition probabilities, provided that complete episodes are obtained directly from the environment. A key characteristic of MC methods is the fact that updates are performed only after the completion of an entire episode, as the return must be fully observed before it can be used for learning, in fact the MC approaches are limited to episodic tasks and may exhibit slower convergence due to the delayed nature of updates.

Temporal Difference (TD) learning combines some features of both DP and MC methods. Standard TD methods are typically model-free, since they can learn directly from sampled transitions without requiring explicit knowledge of the environment's transition probabilities, so like DP, TD methods update value estimates incrementally at each time step, without waiting for the end of an episode, and similarly to model-free MC methods, they learn directly from experience. TD learning achieves this by using bootstrapping. This means that current value estimates are updated based on other learned estimates rather than solely on observed returns, this allows for more efficient and flexible learning, especially in non-episodic or continuing tasks environments where complete episodes may not be naturally available.

Overall, these three classes of methods represent different trade-offs between model dependence, computational efficiency, and learning speed. In financial applications, where the underlying environment is complex and only partially observable, reinforcement learning methods that can learn from sampled data without a fully specified transition model may be particularly useful. For example, approaches like Monte Carlo prediction and Temporal Difference learning can be implemented in a model-free setting, as they can adapt to data without requiring a fully specified model of market dynamics (Andrew, 1999).

2.2.2. Reinforcement Learning workflow

The training of a Reinforcement Learning (RL) agent follows a structured workflow that involves: the definition of the environment, the specification of the reward function, the estimation of value

functions, and the iterative improvement of the policy. These components interact dynamically during the learning process, allowing the agent to progressively refine its decision-making strategy.

The first step consists in *defining the environment*. The environment represents the context in which the agent operates, it can be either simulated or real, although simulation-based environments are typically preferred during the development phase, as they allow for controlled experimentation and reduce the risks associated with real-world implementation.

The second step involves the *specification of the reward function*, which defines the objective of the agent and assigns a scalar value to each state or state–action transition, reflecting the desirability of a given outcome. Designing an appropriate reward function is a critical and non-trivial task, as it directly determines the behavior that the agent will learn. The policy is defined as the mapping from states to actions. The meanings of state and action are respectively: a state represents the information available to the agent at a given time; while an action corresponds to a decision taken by the agent based on that information (Andrew, 1999).

The third step concerns the *definition of the learning representation*, which may include the estimation of a value function depending on the selected RL algorithm. The value function estimates the expected cumulative reward associated with a given state under a specific policy. It is continuously updated based on the observed consequences of the agent's actions. In episodic tasks, this return is computed over the remaining duration of the episode possibly with discounting, while in continuing tasks (where there is no natural terminal state) it is usually represented as a discounted sum of future rewards. The discount factor gives lower weight to rewards received further in the future and allows the agent to balance immediate and long-term outcomes.

The fourth step consists in *designing and configuring the agent*, which includes the choice of the learning algorithm. A central challenge in this phase is balancing exploration (trying new actions to improve knowledge of the environment) and exploitation (leveraging current knowledge to maximize rewards). The learning process is characterized by the interdependence between policy improvement and value estimation: the value function depends on the policy, while the policy is updated based on value estimates (Andrew, 1999).

The fifth step involves *training and evaluation* of the agent. In this step the agent repeatedly interacts with the environment, observes the consequences of its actions, receives rewards, and updates its policy and/or value estimates according to the selected learning algorithm. During this phase, the training parameters, specified during the configuration of the process (such as stopping criteria, discount factor and learning rates) are monitored throughout learning. The evaluation is particularly important because reinforcement learning performance can not be assessed only by observing the training reward. The learned policy should be tested on separate episodes, simulated scenarios or out-of-sample data in order to verify whether it generalizes beyond the training environment. In financial

applications, this validation step usually requires a chronological out-of-sample backtesting procedure, since random data splitting may introduce look-ahead bias and lead to unrealistic performance estimates.

Finally, once the training process is complete, the learned policy can be deployed as a standalone decision-making system, capable of operating independently within the defined environment (MathWorks).

In financial applications, reinforcement learning can be applied to portfolio optimization by framing the investment problem as a sequential decision-making process. In this setting, the state typically includes market information such as asset prices, returns, or macroeconomic indicators, while actions correspond to portfolio allocation decisions (selection of asset weights). After each decision, the agent observes the evolution of the market (the next state) and receives a reward based on portfolio performance.

The reward function is often defined using financial performance measures such as cumulative returns, the Sharpe ratio, or the Sortino ratio. The objective of the agent is to learn a policy that maximizes long-term portfolio performance by dynamically adjusting asset allocations in response to changing market conditions (Benhamou, Saltiel, & Abh, 2020).

2.3. Artificial Neural Networks

Artificial Neural Networks (ANNs) are computational models inspired by the structure and functioning of the human brain. They are designed to learn complex patterns from data and are particularly effective in identifying non-linear relationships, even in the presence of noise and incomplete information.

An ANN is composed of interconnected processing units, called neurons, organized in layers. The overall structure of the network, that is known as its topology, defines how these neurons are arranged and connected. The network typically consists of an input layer, which receives external data, an output layer, which produces the final predictions, and one or more hidden layers, which process intermediate representations of the data.

The connections between neurons are characterized by weights, which determine the strength of the signal transmitted between nodes. These weights, during the learning process, are iteratively adjusted in order to improve the accuracy of the model.

Neural Networks can be classified based on their topology. In Feedforward Network, information flows in a single direction, from the input layer to the output layer, without forming cycles. In contrast, Recurrent Neural Networks (RNNs) include feedback connections, allowing information to persist across time steps and making them particularly suitable for sequential data (Fadlalla & Lin, 2001).

The learning process in ANNs is governed by optimization algorithms that aim to minimize a

predefined loss (or cost) function. A commonly used training procedure is backpropagation, in which the error between predicted and actual outputs is propagated backward through the network to update the weights. This process is repeated iteratively over the training dataset, enabling the network to progressively improve its predictions (Siami Namin & Siami Namin, 2018).

Once trained, neural networks can be applied to various tasks such as regression, classification, and pattern recognition, using new and unseen data. Their ability to approximate complex functions makes them particularly useful in financial applications, where relationships between variables are often non-linear and difficult to model using traditional approaches (Tu, 1996).

In the context of portfolio optimization, Artificial Neural Networks have been used to enhance traditional models by improving the estimation of expected returns and risk measures, some studies have proposed hybrid approaches combining ANNs with classical optimization frameworks, replacing variance-based risk measures with alternative indicators such as semi-absolute deviation. These approaches relax the assumption of normally distributed returns and provide more flexible and robust solutions, particularly in large-scale portfolio optimization problems (Ma, Han, & Wang, 2020).

2.3.1. Performance measures of Artificial Neural Networks

The performance of Artificial Neural Networks can be evaluated using different metrics. The one chosen depends on the specific task and application context of ANN. In financial applications, performance measures are typically grouped into three main categories: predictive accuracy, profitability, and risk-related metrics.

Predictive accuracy refers to the ability of the ANN to correctly estimate or classify outcomes. In regression tasks, this is commonly assessed using error-based measures such as Mean Squared Error (MSE) or Mean Absolute Error (MAE), while in classification problems it is evaluated through metrics such as accuracy, precision, or directional accuracy. This category captures how well the model fits the data, but it does not necessarily reflect its economic value.

Profitability measures evaluate the financial performance of the decisions based on the model's outputs. These metrics assess whether the predictions generated by the neural network translate into economically meaningful gains, by looking at cumulative returns, average returns, and return on investment. In portfolio management, profitability is often evaluated through performance ratios such as the Sharpe ratio, which relates the excess returns to the level of risk taken.

Risk measures focus on the potential losses associated with the use of the model. In financial contexts, it is not sufficient for a model to generate high returns, but it must also control the level of risk. Risk can be assessed using metrics such as volatility, Value at Risk (VaR), or the number and magnitude of losses. These measures provide insight into the stability and robustness of the model's

performance, under different market conditions (Fadlalla & Lin, 2001).

A comprehensive evaluation of neural network performance should consider all three dimensions simultaneously. In particular, a model that exhibits high predictive accuracy may still be unsuitable for financial applications if it does not lead to profitable decisions or if it exposes the investor to excessive risk, so balancing accuracy, profitability, and risk is essential when assessing the effectiveness of ANN-based models in portfolio optimization.

2.4. Deep Learning

Deep Learning is an advanced subset of machine learning that enables models to learn complex representations of data through multiple layers of abstraction. Unlike traditional machine learning approaches, which often rely on manually engineered features, Deep Learning models automatically extract relevant features directly from raw data, allowing for a more flexible and scalable learning process.

A key characteristic of deep learning is its ability to model data through a hierarchical structure of representations; with this framework simple patterns are learned at lower layers and progressively combined into more complex features at higher levels. This hierarchical learning process allows Deep Learning models to capture intricate structures and dependencies in large and high-dimensional datasets (Goodfellow, Bengio, & Courville, 2016).

Deep Learning models are typically implemented using Deep Neural Networks (DNNs), which extend traditional Neural Networks by increasing the number of hidden layers. This increased depth enhances the model's capacity to approximate complex, non-linear functions, making it particularly suitable for applications involving large-scale and highly structured data.

Several architectures have been developed to address different types of data and learning tasks, that are (Zou, Huss, Abid, & Moha, 2024):

- *Feedforward Neural Networks* (FNNs), in these models the information flows in a single direction from input to output layers. They are widely used for general prediction tasks where no specific structure is assumed in the data;
- *Convolutional Neural Networks* (CNNs), which are designed to process data with spatial structure. By applying convolutional filters across input matrices, CNNs can detect local patterns and achieve invariance to certain transformations, making them particularly effective in image and signal processing tasks;
- *Recurrent Neural Networks* (RNNs), which are specifically designed for sequential and time-series data. Unlike feedforward networks, RNNs include feedback connections that allow information to persist across time steps, this enables the model to capture temporal dependencies by incorporating past information into current predictions.

However, standard RNNs suffer from important limitations, particularly in learning long-term dependencies. Due to issues such as vanishing and exploding gradients, these models tend to retain information only over short time horizons, limiting their effectiveness in applications involving long sequences.

To address this limitation, more advanced architectures such as Long Short-Term Memory (LSTM) networks have been developed. LSTMs introduce a memory mechanism that allows the network to selectively retain and update information over longer time periods, significantly improving performance in time-series modeling tasks (Siami Namin & Siami Namin, 2018).

In financial applications, deep learning models, especially RNNs and LSTMs, have proven particularly effective in modeling financial time series, where data exhibit temporal dependencies, non-linearity, and high levels of noise. These characteristics make Deep Learning a powerful tool for return prediction and dynamic portfolio optimization.

2.4.1. Structure of Deep Learning models

Deep learning models are designed to learn complex input–output relationships directly from data through multiple layers of non-linear transformations. A crucial step in building such models is the preparation of an appropriate training dataset, which is used to estimate the parameters of the network, and the selection of suitable evaluation metrics to assess performance.

The term “deep” refers to the presence of multiple processing layers in the network. A typical Deep Neural Network consists of an input layer, which receives the data, several hidden layers, which progressively transform the input through non-linear operations, and an output layer, which produces the final prediction, the number of units in the output layer depends on the specific task, such as regression or classification (Ahmed, et al., 2023).

Each hidden layer applies a linear transformation followed by a non-linear activation function, enabling the model to capture complex relationships in the data. Formally, the transformation at layer l can be written as eq. (9).

$$Z^{(l)} = f^{(l)} \left(W^{(l)} Z^{(l-1)} + b^{(l)} \right) \quad (9)$$

where $Z^{(l)}$ denotes the output of layer l , $W^{(l)}$ is the weight matrix, $b^{(l)}$ is the bias vector, and $f^{(l)}$ is the activation function (Zrara, 2021).

By stacking multiple layers, deep learning models construct hierarchical representations of the input data. Each layer extracts increasingly abstract features. This hierarchical structure allows the model to learn complex patterns that are difficult to capture with shallow architectures.

The learning process is based on the minimization of a loss function $[L(Y, \hat{Y})]$. This loss function measures the discrepancy between the predicted output and the true target values. A commonly used

loss function in regression tasks is the squared error, and it is represented by eq. (10) (Zrara, 2021)

$$L(Y, \hat{Y}) = \|Y - \hat{Y}\|_2^2 \quad (10)$$

where Y is the actual true value observed, $\hat{Y}$ is the predicted value from the model and $\|\dots\|_2^2$ represents the sum of the squares of the differences.

The optimization is typically performed using the backpropagation algorithm, which computes the gradient of the loss function with respect to the model parameters by applying the chain rule. These gradients are then used to update the weights iteratively. This allows the model to improve its predictions over time (Ahmed, et al., 2023).

Given an input vector X , the deep learning model defines a mapping of the form of eq. (11)

$$\hat{Y}(X) = F(X) \quad (11)$$

where $F(\cdot)$ represents a composition of multiple non-linear transformations across layers. The training dataset can be represented as a collection of input–output pairs $\{(X^{(i)}, Y^{(i)})\}_{i=1}^T$, which are used to estimate the parameters of the model.

To evaluate the generalization ability of the model, a separate testing dataset is employed. The predictions generated by the trained model are compared with the observed values to assess performance and ensure that the model does not overfit the training data.

Overall, the layered structure, combined with non-linear activation functions and gradient-based optimization, allows deep learning models to approximate highly complex functions, making them particularly suitable for high-dimensional and non-linear problems such as financial time series analysis.

2.5. Advantages and limitations

2.5.1. Reinforcement Learning

Reinforcement Learning (RL) presents several advantages that make it particularly suitable for complex and dynamic decision-making problems.

One of its main strengths is its versatility and adaptability, RL agents learn directly from interactions with the environment, allowing them to operate effectively even in settings where the underlying system is not fully specified. This characteristic makes RL particularly appealing in dynamic environments, such as financial markets, where conditions continuously evolve (Srinivasan, 2023).

Another important advantage is the ability of RL models to support continuous learning and generalization. In fact, agents can progressively improve their behavior as new information becomes available, making them well-suited for long-term and sequential decision-making tasks. This adaptability is crucial in applications where the environment is non-stationary and subject to structural changes over time.

A key feature of reinforcement learning is the management of the exploration–exploitation trade-off. At each step, the agent must decide whether to explore new actions in search of potentially higher rewards or to exploit previously learned strategies that have proven effective, and the ability to balance these two aspects is fundamental for identifying optimal policies and achieving long-term performance improvements (Srinivasan, 2023).

Despite these advantages, RL also presents several limitations that affect its practical implementation. A major drawback is sample inefficiency, as RL algorithms often require many interactions with the environment to learn effective policies. This can be computationally expensive and, in real-world applications, may be impractical or even infeasible.

Another critical issue concerns the reward design and training stability. The specification of an appropriate reward function is a complex task, and poorly designed rewards may lead the agent to learn suboptimal or unintended behaviors, resulting in instability during training and unreliable outcomes.

Reinforcement learning methods also suffer from the curse of dimensionality. This happens especially when dealing with large state and action spaces. As the dimensionality increases, the number of possible states grows exponentially, making it more difficult for the agent to explore the environment efficiently and learn accurate value functions.

Furthermore, safety and ethical concerns arise in real-world applications, where RL agents may learn behaviors that are unsafe or undesirable. This is especially relevant in sensitive domains such as robotics, healthcare, or automated trading systems, where incorrect decisions can have significant consequences (Srinivasan, 2023).

Finally, it is important to note that some Reinforcement Learning approaches, particularly those based on dynamic programming, rely on the assumption of a complete and known model of the environment. In practice, this assumption is rarely satisfied, and such methods quickly become computationally intractable as the complexity of the system increases (Andrew, 1999).

Overall, while Reinforcement Learning provides a powerful and flexible framework for sequential decision-making, its practical application requires careful consideration of the computational constraints, model design, and real-world implementation challenges.

2.5.2. Artificial Neural Network

Artificial Neural Networks (ANNs) offer several advantages that make them particularly suitable for modeling complex relationships in financial data.

One of their primary strengths is the ability to approximate highly non-linear functions without requiring an explicit specification of the model structure in advance. This flexibility allows ANNs to adapt to a wide range of problems, especially in contexts where the underlying relationships between

variables are unknown or difficult to model analytically (Livingstone, Manallack, & Tetko, 1997). Another important advantage is their capability to capture complex interactions among input variables. Using hidden layers and non-linear activation functions, ANNs can implicitly model interactions and dependencies between predictors, often outperforming traditional linear or parametric models when significant non-linearity is present in the data.

ANN models can be developed relatively efficiently once an appropriate dataset is available. While training may be computationally intensive, the model-building process does not require strong statistical assumptions, such as linearity or normality of residuals. This makes ANNs particularly attractive in applications where traditional econometric models may be too restrictive.

A further advantage lies in the availability of multiple training algorithms and architectures. This provides flexibility in adapting the model to different types of problems. However, this flexibility also introduces challenges, as selecting the most appropriate architecture and training procedure often requires empirical testing and model validation (Tu, 1996).

Despite these strengths, Artificial Neural Networks also present several limitations.

One of the main drawbacks is their limited interpretability. Compared to traditional models such as logistic regression, ANNs are often considered “black-box” models, as it is difficult to identify the contribution of individual variables to the final prediction. This lack of transparency can be problematic in applications where interpretability and explainability are important.

Another limitation concerns the difficulty in establishing causal relationships. While ANNs are effective at capturing correlations and patterns in the data, they do not inherently provide insights into causal mechanisms, making them less suitable for explanatory analysis compared to traditional statistical models.

Furthermore, ANN models typically require significant computational resources and training time, especially when dealing with large datasets or complex architectures. The optimization process may be time-consuming and sensitive to the choice of hyperparameters, such as learning rate, number of layers, and network size.

Finally, the development of ANN models is often characterized by a high degree of model uncertainty. Given the wide range of possible architectures and training algorithms, it is difficult to determine *ex ante* which configuration will perform best for a given problem. So, model selection relies heavily on empirical experimentation, and there are limited theoretical guidelines to guarantee optimal performance (Tu, 1996).

Overall, while Artificial Neural Networks provide a powerful and flexible framework for modeling complex data, their practical use requires careful consideration of interpretability, computational cost, and model selection challenges.

2.5.3. Deep Learning

Deep learning models offer several advantages that make them particularly suitable for high-dimensional and complex data environments.

One of their primary strengths is the ability to process large volumes of heterogeneous data, including structured and unstructured inputs such as images, text, and time-series data. This scalability allows deep learning models to perform effectively in applications where traditional methods struggle, especially when dealing with massive datasets.

Another key advantage is the capability of automatic feature extraction. Unlike traditional machine learning approaches, which often rely on manually engineered features, deep learning models learn hierarchical representations directly from the data, and by combining simple features into increasingly complex ones across multiple layers, these models can capture intricate patterns without requiring explicit human intervention. This significantly reduces the need for domain-specific feature design and can improve learning efficiency (Ahmed, et al., 2023).

In financial applications, this ability is particularly valuable. Deep learning models can integrate diverse sources of information, such as historical price data, macroeconomic indicators, and even textual data from news or social media, to identify complex relationships and improve predictive performance, this makes them suitable for tasks such as return forecasting and portfolio optimization, where interactions between variables are often non-linear and difficult to model analytically.

Despite these advantages, deep learning models also present several limitations.

One of the main drawbacks is their lack of interpretability. These models are often considered “black-box” systems. Meaning that it is difficult to provide clear explanations for the decisions they produce. This lack of transparency can be problematic in financial contexts, where understanding the drivers of a model’s predictions is often essential (Kufel, et al., 2023).

Another important limitation is the difficulty of human intervention and model debugging. Deep learning architectures do not easily allow practitioners to trace the reasoning behind specific outputs. So, correcting errors or improving model performance may require extensive retraining and experimentation.

Furthermore, Deep Learning models require significant computational resources, in fact training these models typically involves high-performance hardware, such as GPUs, and large amounts of memory and storage. In addition, training times can be substantially longer compared to traditional machine learning techniques, particularly when large datasets are required to achieve superior performance (Ahmed, et al., 2023).

Overall, while Deep Learning provides a powerful framework for modeling complex and high-dimensional data, its practical application requires careful consideration of interpretability, computational cost, and data availability constraints.

2.6. Synthesis and implications for portfolio optimization

This chapter has presented an overview of modern machine learning approaches applied to complex decision-making problems, with a particular focus on Reinforcement Learning, Artificial Neural Networks, and Deep Learning models.

These methodologies provide a flexible and data-driven framework capable of capturing non-linear relationships and processing large volumes of heterogeneous data. They have solved some of the limitations of traditional model-based approaches.

Reinforcement Learning offers a powerful paradigm for sequential decision-making. They make the agent able to learn optimal strategies through interaction with the environment.

Artificial Neural Networks extend this capability by modeling complex functional relationships.

On the other hand, Deep Learning architectures further enhance predictive performance through hierarchical feature extraction and representation learning.

Despite their advantages, these models also present important challenges. They are showing: limited interpretability, high computational requirements, and sensitivity to model design and data quality. These limitations highlight the need for careful model selection and adaptation, particularly in financial applications where data are noisy, non-stationary, and characterized by temporal dependencies.

In this context, the choice of an appropriate model architecture becomes crucial. Among Deep Learning approaches, particular attention has been given to models specifically designed for sequential data, which are able to capture temporal dynamics and long-term dependencies more effectively.

The next chapter focuses on Long Short-Term Memory (LSTM) networks, a class of Recurrent Neural Networks that has proven particularly effective in modeling financial time series. These models will be analyzed in detail, with the aim of evaluating their potential in improving portfolio optimization compared to traditional approaches.

CHAPTER 3 – Deep Learning architectures for financial time series forecasting

3.1. Motivation for the use of Deep Learning models

Overall, the adoption of Machine Learning techniques represents a shift from model-driven to data-driven portfolio optimization, aiming to overcome the limitations of classical frameworks and provide more reliable investment strategies in complex and dynamic environments (Gu, Kelly, & Xiu, 2020). While Reinforcement Learning and Artificial Neural Networks provide valuable frameworks for prediction and decision-making, Deep Learning models are particularly suitable for financial time series analysis because of their hierarchical structure and their ability to automatically extract complex features from data.

In fact, among all the different Machine Learning approaches presented in the previous chapter, now the focus will be only on Deep Learning models, because these models have shown ability to process large quantities of data, identify non-linear relationships, and model sequential dependencies within time series.

Within the broad family of Deep Learning models, particular attention is given to Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks.

CNNs are introduced because of their capability to extract local patterns and relevant features from structured datasets. However, despite their effectiveness in feature extraction tasks, CNNs are not specifically designed to model long-term temporal dependencies, and this characteristic is something very important to be captured for portfolio optimization activity. For this reason, the focus of this chapter will be especially on Long Short-Term Memory networks.

LSTM is a specialized form of Recurrent Neural Networks, that are a specific kind of Deep Learning models, explicitly developed to process sequential and time-dependent data. Thanks to their memory structure, they are able to retain information over longer horizons and overcome some of the limitations of traditional models, like the vanishing gradient problem.

For this reason, the empirical work of the thesis will be developed using the LSTM structures for the financial forecasting of the returns that will be used for the portfolio optimization.

3.2. Deep Learning models versus Traditional Forecasting models

Time series forecasting is traditionally referred to stochastic and econometric models designed to capture temporal dependence in observed data. The most used approaches are:

- The Auto-Regressive Moving Average (ARMA) model; that combines autoregressive components with moving-average components.
- The Auto-Regressive Integrated Moving Average (ARIMA) model, that extends the ARMA

model by incorporating the differentiation of the dataset to handle non-stationary time series. In recent years, Machine Learning, especially Deep Learning techniques, have introduced alternative approaches for forecasting problems. Unlike traditional models, which are usually based on predefined statistical assumptions and linear structures, Deep Learning models can learn complex and non-linear relationships directly from data through multiple layers of representation without needing additional information from the users. This characteristic makes them particularly useful in financial applications, where time series often show noise, non-linearity, volatility clustering, and changing market dynamics.

Deep Learning architectures such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Convolutional Neural Networks (CNNs) have gained increasing attention in forecasting applications, including finance.

RNNs and LSTMs are especially suitable for sequential data. This because they are designed to capture temporal dependencies.

On the other hand, CNNs can be used to identify local patterns and structures in financial time series. Several studies have compared traditional forecasting models, like ARIMA, with deep learning approaches, particularly LSTM networks. Some empirical studies have reported substantially lower forecasting errors for LSTM models compared with ARIMA. However, these results should not be interpreted as universal evidence of LSTM superiority, since forecasting performance depends on the dataset, the forecasting horizon, the model specification, and the evaluation procedure (Zhang & et al, 2019).

The predictive accuracy of forecasting models is commonly evaluated by using error-based measures, like: the Root Mean Squared Error (RMSE) and the Mean Absolute Percentage Error (MAPE).

RMSE is used to measure the average magnitude of prediction errors and assigns greater weight to larger errors, it is computed by eq. (12).

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \hat{x}_i)^2} \quad (12)$$

where N is the number of observations, x_i is the actual value, and $\hat{x}_i$ is the predicted value. A key advantage of RMSE is that it penalizes large forecasting errors more heavily (Siami Namin & Siami Namin, 2018).

MAPE measures forecasting accuracy in percentage terms by comparing the absolute prediction error with the actual value, it is computed by eq. (13).

$$MAPE = \frac{1}{N} \sum_{i=1}^N \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100 \quad (13)$$

where y_i is the observed value, $\hat{y}_i$ is the predicted value, and N is the size of the test set. MAPE is

particularly useful because it expresses forecast errors as percentages, making results easier to interpret and compare within the same empirical context (Zhang & et al, 2019).

The comparison between Traditional Forecasting models and Deep Learning models is not only based on theoretical differences, but also on empirical performance; while ARMA and ARIMA models remain useful benchmarks due to their simplicity and interpretability, Deep Learning models offer greater flexibility in capturing complex patterns in financial data, which is one of the reasons why LSTM networks are particularly relevant for portfolio optimization applications.

The LSTM models solve some of the key weaknesses of traditional portfolio optimization approaches by improving return prediction and enhancing the robustness of portfolio construction.

3.3. *Convolutional Neural Networks (CNNs)*

Convolutional Neural Networks (CNNs) are one of the most widely used Deep Learning architectures. They were originally developed for processing grid pattern data, such as images, but they have also been applied to other types of structured data, including natural language and financial time series.

Usually, in time-series forecasting, CNNs can be used to identify local patterns within the data, such as short-term movements, repeated structures, or localized dependencies. This characteristic can be particularly useful in financial applications, where relevant information may be found in short-term windows of past prices, returns or volatility. One of the main advantages of CNNs is that they can reduce the number of parameters compared to fully connected networks, making the learning process more efficient.

A typical CNN architecture is composed of three main components: a convolutional layer, a pooling layer, and a fully connected layer (their functioning will be seen later).

The main operation in a CNN is called convolution. In this operation, the model uses small filters, also called kernels, to scan the input data. Each kernel looks only at a small part of the input data at a time and checks whether a specific pattern is present. For example, in a financial time series, a kernel may detect short-term price movements or repeated return patterns. As the kernel moves across the input, it produces a new representation called a feature map. This feature map shows where and how strongly a certain pattern appears in the data.

In this sense, different kernels can be used as different pattern detectors: each one learns to identify a particular characteristic of the input data (Ahmed, et al., 2023).

The convolution operation is represented by eq. (14).

$$f_1^k(p, q) = \sum_c \sum_{x,y} i_c(x, y) \odot e_1^k(u, v) \quad (14)$$

where $f_1^k(p, q)$ represents the output feature map generated by the k -th convolution operation at the

layer l ; $i_c(x, y)$ denotes the input tensor element at position (x, y) in the c -th input channel, where $c = 1, \dots, C_l$ indexes the input channels of layer l ; $e_l^k(u, v)$ is the k -th convolutional kernel associated with the corresponding output feature map and the c -th input channel; and $\odot$ is the element-wise multiplication operator (Ahmed, et al., 2023).

CNNs rely on two key principles that reduce the number of parameters and improve computational efficiency, that are:

- The local connectivity, meaning that each neuron processes only a local region of the input data rather than the entire input space.
- The weight sharing. This means that the same kernel is applied across different regions of the input, this allows the network to detect the same pattern in different positions while using the same set of parameters (Gao, Zhang, & Yang, 2020).

In financial applications, CNNs may be useful for detecting local structures in price movements or return sequences, but unlike recurrent architectures they do not explicitly model long-term temporal dependencies. For this reason, in fact, CNNs are often useful for feature extraction, while models such as LSTM networks are generally more appropriate when the main objective is to capture sequential dependence over time.

3.3.1. CNN structure

After having introduced the general logic of the Convolutional Neural Networks, this section describes the main components of a typical CNN structure.

A CNN is composed of multiple layers that progressively process the input data to extract features that describe them. In a CNN architecture, the output generated by one layer becomes the input of the following layer, allowing the network to learn increasingly complex representations of the data, and through repeated convolution and pooling operations, the model can extract relevant features and reduce unnecessary information (Gao, Zhang, & Yang, 2020).

CNN architectures are mainly composed of three different types of layers:

- *Convolutional layer*. It is the core component of a CNN, it applies a set of filters, also called kernels, to the input data to detect local patterns and relevant features. Each filter scans the input and produces a feature map that highlights specific characteristics of the data. The filters are not predefined but are learned automatically during the training process. In fact, the network is adapted automatically to the structure of the dataset. By combining multiple filters, the network can progressively construct more complex and global representations from local information.
- *Pooling layer*. It is generally introduced after convolutional layers. The objective of this layer

is to reduce the dimensionality of the feature maps. Meaning that it tries to decrease the number of parameters and computational complexity of the model. This reduction also helps in limiting overfitting and improving the robustness of the network by retaining only the most relevant information extracted from the previous layers.

- *Fully connected layer.* After several convolutional and pooling operations, the extracted features are transformed into a one-dimensional vector and passed to the fully connected layers. In these layers, each neuron is connected to all neurons of the following layer through changeable weights. These layers use the extracted features to produce the final prediction or classification result. When the problem requires a classification result it is usually applied the SoftMax activation function that will generate a probability distribution over the possible output classes (Gao, Zhang, & Yang, 2020).

Despite their effectiveness, CNN models also present some limitations.

One of the main drawbacks is the large number of trainable parameters, especially in deep architectures, by increasing the number of layers and parameters of the model, its computational cost raises and may lead to overfitting when the available training dataset is not sufficiently large. To reduce this problem, regularization techniques (like dropout) are commonly used. The dropout technique randomly and automatically deactivates a subset of neurons during training, reducing the dependence between nodes and improving the generalization ability of the network (Ahmed, et al., 2023).

3.3.2. Image recognition models with financial datasets

Recent studies have proposed approaches that transform one-dimensional financial time series into two-dimensional image-like representations in order to exploit the feature extraction capabilities of Convolutional Neural Networks (CNNs).

CNN models can be applied to algorithmic trading and financial forecasting problems, by converting financial data into structured visual representations (Chen, Chen, Huang, Huang, & Chen, 2016).

Several parameters are required to construct this type of model, that are:

- *Two-dimensional data size (D).* First, the image size must be defined. If the image dimension is $D \times D$, each generated image will contain D^2 financial observations arranged in a two-dimensional structure. The choice of image size directly affects the quantity of information processed by the CNN.
- *Training iterations (N).* This parameter defines the number of times the model updates its weights during the training process. The appropriate number of training iterations depends on the size of the dataset, the complexity of the input patterns, and the architecture of the CNN model. In general, larger and more complex datasets may require a higher number of training

iterations to allow the model to learn meaningful representations. However, an excessive number of iterations may lead to overfitting, meaning that the model performs well on the training data but poorly on unseen data. For this reason, the number of iterations is usually selected empirically and monitored, and if necessary changed, through validation performance.

- *Prediction horizon (F)*. This parameter defines the future time interval over which the predictions are made. In financial applications, selecting an excessively short prediction horizon may expose the model to high short-term market noise and fluctuations, for this reason, intermediate horizons are often preferred.
- *Threshold value*. In classification problems, CNNs usually produce an output score for each possible class. This score indicates the confidence of the model in assigning the input observation to a specific category. A threshold value can be introduced to accept only predictions with sufficiently high confidence (higher score means higher confidence). Predictions below the threshold may be discarded or classified as uncertain, reducing the impact of low-confidence signals and limiting unreliable trading decisions.
- *Training period (TP)*. It defines the amount of historical data used to train the model. While larger training datasets may improve learning accuracy, the optimal training period must generally be determined empirically through several trials.
- *Time frame (TF)*. It determines the sampling frequency of the financial data. The dataset may include variables such as opening price, closing price, highest price, and lowest price, collected at different intervals (for example: daily or intraday frequencies such as 5-minute intervals).

Before training, the two-dimensional data are typically standardized to improve numerical stability and model performance. The most used normalization procedure is the one represented by eq. (15).

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (15)$$

where X is the observed value, X_{min} is the minimum value observed in the sample, X_{max} is the maximum value observed in the sample and X_{new} is the final value normalized.

This transformation is necessary to rescale the data into a normalized range, improving the efficiency of the learning process and reducing the effect of scale differences among variables (Chen, Chen, Huang, Huang, & Chen, 2016).

3.4. Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a specific type of Recurrent Neural Network (RNN), it introduces a memory cell, which allows the network to retain, update, or discard information over

time. This cell is necessary for the model to assess whether information is relevant and should be retained over time and, for this reason, LSTM networks are particularly effective in learning long-term dependencies within sequential data (Zhang & et al, 2019).

A standard LSTM unit is composed of a cell that includes three main components: an input gate, an output gate, and a forget gate. These gates interact with each other and regulate the flow of information into and out of the cell. Also, the memory cell is able to retain values over specific time intervals, allowing the network to preserve useful information across different steps of the sequence. The key element of the LSTM architecture is the cell state (C_t). The cell state is transmitted through the network mainly through element-wise operations, which allow information to flow across time steps without being repeatedly transformed by nonlinear functions. At the same time, it can, if the model considers it important, remove or update information contained in the cell state through the gate mechanism.

The gate mechanism enables information to pass selectively through the network, and it is based on the sigmoid activation function (how it works will be explained in the next section) which produces values between 0 and 1 to determine which portion of the previous cell state should be transferred or removed or added to the next state. In this way, the LSTM can control what information must be forgotten, what information must be stored, and what information must be passed forward.

LSTM networks have been widely applied to sequential data, including text classification and time series prediction. They were specifically developed to overcome some of the main limitations of traditional RNNs, especially their difficulty in capturing long-term dependencies due to vanishing and exploding gradient problems (Gao, Zhang, & Yang, 2020).

After the construction of the LSTM model, training and parameter optimization represent two fundamental stages.

The *training phase* is necessary for the model to learn directly from the data by iteratively adjusting its internal weights in order to minimize prediction error, this process allows the network to progressively improve its ability to identify patterns and dependencies within the input time series.

The *Parameter optimization* stage aims, instead, to identify the most appropriate configuration of the model, thereby improving its predictive performance.

To properly evaluate the model, the financial time series dataset will be divided into three separate subsets: a training set, a validation set, and a testing set. Their roles are:

- The *training set* is used to fit the model;
- The *validation set* supports hyperparameter tuning and helps reduce the risk of overfitting.
- The *testing set* is used to evaluate the final performance of the model on data that were not previously observed during the training process.

One of the most relevant hyperparameters is the learning rate, which controls the magnitude of the

weight updates during the optimization process. The learning rate determines how quickly the model adjusts to the error signal, if it is too low, the training process may become very slow and require a long time to converge. Conversely, if the learning rate is too high, the optimization process may become unstable, causing the model to overshoot the optimal solution or fail to converge (Zhong, 2024).

3.4.1. LSTM architecture

The LSTM architecture relies on three types of gates to regulate the information contained in the cell state. In general, each gate can be expressed through a sigmoid transformation of the form of eq. (16), that is:

$$\sigma(W_i X + b_i) . \quad (16)$$

The elements of eq. 16 will be seen later for each different type of gate.

The output of the sigmoid function ranges between 0 and 1, and this value determines the proportion of information from the previous cell state (C_{t-1}) that should be allowed to pass to the next state (C_t). If the output is close to 0, little or no information is retained; if the output is close to 1, most or all of the information is preserved.

The range between 0 and 1, for the final value of the gate, is guaranteed by the sigmoid activation function, that is represented by eq. (17)

$$\sigma(z) = \frac{1}{1+e^{-z}} . \quad (17)$$

In fact, eq. (17) transforms any real-valued input (z) into a value between 0 and 1 (Gao, Zhang, & Yang, 2020).

The three main gates of the LSTM model are: the forget gate, the input gate, and the output gate.

The *forget gate* is a sigmoid layer that allows the LSTM model to determine which information should be removed from the previous cell state. Its output is a vector with values between 0 and 1, computed on the basis of the previous hidden state (h_{t-1}) and the current input (x_t). Values closer to 1 indicate that a larger portion of the previous cell state should be retained, on the other hand values closer to 0 indicate that a larger portion should be dropped.

The forget gate is expressed by eq. (18).

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (18)$$

where f_t represents the output of the forget gate, σ is the sigmoid activation function, W_f is the weight matrix of the forget gate, and b_f is its bias vector. The term $[h_{t-1}, x_t]$ denotes the concatenation of two vectors: the previous hidden state (h_{t-1}), which contains short-term information from the previous time step; and the current input (x_t). The weight matrix contains the parameters learned during training, while the bias vector shifts the result of the linear transformation and allows the

network to learn more flexibly (Gao, Zhang, & Yang, 2020).

The *input gate* determines how much new information should be added to the cell state. This process is divided into two parts, that are:

- First, a sigmoid layer produces values between 0 and 1, determining the portion of the new information that should enter the cell state;
- Second, a hyperbolic tangent layer creates a candidate value ($\tilde{C}_t$), which represents the new potential content to be added to the cell state.

This gate is described by 3 equations, that are: eq. (19), eq. (20) and eq. (21).

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) ; \quad (19)$$

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) ; \quad (20)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t . \quad (21)$$

Eq. (19) defines the output of the input gate (i_t), which controls how much of the new information should be included in the cell state. In this equation, W_i is the weight matrix of the input gate, b_i is its bias vector, and $[h_{t-1}, x_t]$ is the concatenation of the previous hidden state and the current input.

Eq. (20) computes the candidate cell state ($\tilde{C}_t$), which represents the new information that may be added to the cell state. In this case, W_c is the weight matrix associated with the candidate cell state, b_c is the corresponding bias vector, and $\tanh$ is the hyperbolic tangent activation function, which transforms values into the range between -1 and 1.

Eq. (21) represents the update of the cell state at time t . The updated cell state (C_t) depends on two components. The first component ($f_t \odot C_{t-1}$) determines which part of the previous cell state should be retained; the second component ($i_t \odot \tilde{C}_t$) determines which part of the new candidate information should be added to the current cell state. The symbol $\odot$ represents the element-wise product.

The *output gate* determines the output of the LSTM unit on the basis of the updated cell state. Since the output of the LSTM block is represented by the hidden state (h_t), a further fully connected layer should then be applied to the hidden states in order to generate the final prediction.

The equations that describe the output gate are: eq. (22) and eq. (23).

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) ; \quad (22)$$

$$h_t = o_t \odot \tanh(C_t) . \quad (23)$$

Eq. (22) defines the result of the output gate (o_t), which determines how much of the cell state should be shown as output. In this equation, σ is the sigmoid activation function, W_o is the weight matrix of the output gate, and b_o is the bias vector. As in the previous gates, $[h_{t-1}, x_t]$ denotes the concatenation of the previous hidden state and the current input.

Eq. (23) gives the hidden state (h_t), which represents the actual output of the LSTM block at the current time step. This hidden state is then passed to the next time step and can also be used to generate

the final prediction. The term $\tanh(C_t)$ transforms the cell state into values between -1 and 1 (Gao, Zhang, & Yang, 2020).

In the LSTM model the cell state (C_t) represents the internal memory of the LSTM unit. On the other hand, the hidden state (h_t) represents the output produced at each time step. The cell state allows the network to preserve information over time, whereas the hidden state transfers the relevant representation of the current time step to the following step and, eventually, to the final prediction layer.

3.4.2. LSTM framework for portfolio optimization

The traditional Modern Portfolio Theory (MPT) provides a rigorous, quantitative framework for portfolio construction that emphasizes diversification and the trade-off between risk and return. While it has become an important starting point of modern investment theory, its assumptions of normal returns and constant correlations can limit its practical application in volatile or non-normal market conditions.

Despite the traditional approach has been used for a very long period, it has the drawback that it is too much related to the historical information. For this reason, nowadays, the new Machine Learning models (like the LSTM) have been implemented inside the Markowitz method to use, like input, predicted data from the historical one. So, the final model is constructed in two steps; the first one involves predicting the stock returns with the LSTM model, the second step involves the formation of the optimal portfolio by using the predicted returns and the historical standard deviation with the traditional Markowitz approach (Rather, 2021).

The LSTM model is a very valid choice for its ability of learning long-term dependencies and of solving the vanishing gradient problem of the traditional RNN models. This model achieves these goals by using gates that regulate the flow of information in the network (Habib & Meryem-Nadjat, 2025).

The LSTM model is well-equipped for identifying, analyzing, and predicting time-dependent data, thanks to the structures of gates, LSTMs are distinguished by their extensive memory (Pokhrel, 2021/2022).

The implementation of the LSTM model in the portfolio optimization is based on its capacity in predicting the next future returns that will be used, with the historical variance-covariance matrix, inside the Markowitz model, and then the Efficient Frontier is built. To find the best portfolio with the best return-risk trade-off, among those one that lie on the Efficient Frontier, it will be used the Sharpe ratio (given by eq. (6)) replacing the expected return of the portfolio with the predicted return of the LSTM model and leaving the historical standard deviation of the portfolio (Sen, Dutta, &

Mehtab, 2021).

The model for the portfolio optimization is not taking into account the investors' risk preferences or risk-free assets (only the risk-free asset return will be used for the computation of the Sharpe ratio) (Priya, Manoj, Mohamed, Mohamed, & Thippa reddy, 2023).

In the portfolio optimization process, risk is represented by the daily standard deviation of each stock, that is consistent with the use of the daily returns. The standard deviations are estimated from historical daily returns and, together with the correlations among assets, are used to construct the variance-covariance matrix.

The construction of the LSTM model starts with the collection of the historical financial data. In particular, daily historical stock prices have been downloaded and subsequently processed in order to compute the daily returns and divided in three different sets for the training, validation, and testing of the model (in the next chapter there will be a focus on the kind of data downloaded).

Since the model is applied to financial time series, this division is performed by preserving the chronological order of the observations, rather than randomly splitting the data. To have consistent prediction results among all the stocks, the division of the dataset is the same for all the stocks (Abdellilah, Abdellah, & Abdellah, 2023).

The training set is used to train the network and to iteratively adjust the weights of the parameters in order to minimize the loss function.

The validation set is used to validate the prediction capacity of the model, in fact, after each epoch, meaning one complete pass through the training samples, the network predicts the values of the validation set and a validation loss is computed. When the validation loss no longer decreases, the training process is stopped, and the weights corresponding to the lowest validation loss are retained (Fischer & Krauss, 2017).

The test set is used to evaluate the trained model to get a better idea of how well the model will perform on new data and how it will behave in a production environment. To have a better final result, the test data should be similar to what is expected to be seen in the production environment.

As optimizer for the model clustering the Adam was chosen (Habib & Meryem-Nadjat, 2025).

After the input data have been specified, it is crucial to define the input variables, especially for problems involving timeseries prediction. A forecast model that takes into account only one attribute may not produce reliable predictions. The effectiveness of forecasting algorithms may be greatly improved by financial aspects, such as fundamental and technical indicators that examine the pattern and value of stocks (like momentum, variance of the volume etc...) (Priya, Manoj, Mohamed, Mohamed, & Thippa reddy, 2023). The next chapter will present the input variables used in the LSTM model.

After the input data have been split in three sets, before the LSTM can start the learning process it is

important that the data set is normalized to improve the numerical stability of the data for the learning phase and consequently to improve the predictive performance of the LSTM model (Chen, Zhou, & Dai, 2015).

Normalization is necessary because the network is based on gradient-based optimization and uses nonlinear activation functions inside its recurrent gates, so, if the input variables are expressed on very different numerical scales, the learning process may become unstable, and the model may assign excessive importance to variables with larger magnitudes. The normalization improves numerical stability, accelerates convergence, and allows the LSTM to learn temporal relationships more effectively. In this study, the normalization parameters are computed only on the training set and then applied to the training, validation and test sets, in order to avoid data leakage.

Data leakage occurs when the model receives information that it should not know during training, meaning that information from the future or from the test/validation set accidentally enters inside the training process. When this happens, the model looks better during the validation or testing phase, but the final result is misleading because, in a real forecasting situation, that information would not be available. For this reason, the normalization parameters take into account only the training set to not have inside information from the subsequent data.

This step is particularly relevant in financial applications, where raw prices, returns, volatility measures and technical indicators can have very different magnitudes. Moreover, stock prices are often non-stationary and may follow an upward or downward trend over time. The normalization procedure reduces the numerical impact of these scale differences, although it does not fully solve the non-stationarity problem. For this reason, predicting returns rather than price levels can be more appropriate when the final objective is portfolio optimization, because returns are usually more stationary than price levels and are more directly suitable for portfolio optimization.

The normalization procedure was performed with eq. (24), that is:

$$x_{norm} = \frac{x - \mu_{train}}{\sigma_{train}} \quad (24)$$

where x_{norm} is the normalized value, x is the actual value and μ_{train} and σ_{train} are respectively the mean and standard deviation of each input feature computed exclusively on the training set.

These parameters are used to standardize the training, validation, and test sequences. The same approach was applied to the target variable, whose mean and standard deviation were also estimated only from the training target values. This procedure prevents data leakage, since no information from the validation or test periods is used to determine the normalization parameters.

The application of LSTM models to financial data must be interpreted carefully, since financial markets are highly noisy, non-stationary and affected by unpredictable external factors. When the model is too complex and its learning depends too much on the training set, the problem of overfitting could appear. Overfitting occurs when a model becomes excessively complex and fits the training

data too closely, this could lead to poor generalization (Hawkins & Douglas, 2003).

So, the model memorizes the past information instead of learning a useful pattern that can work for future predictions. In other words, the model learns not only the underlying relationships of the data but also the noise contained in the sample. This means that the network may become very good at fitting the training period, but when it is tested on the validation set or the test set, its performance becomes worse (that is the poor generalization effect of the model). In fact, the performance of the model should be evaluated on the validation and test sets, rather than on the training set alone.

To avoid the problem of overfitting, inside the LSTM model, it should be used the dropout technique. With the dropout the model randomly disables part of the neurons of the network during training, this forces the model not to rely too much on specific neurons and helps generalization and learning common patterns among the data (Rather, 2021).

The opposite of overfitting is underfitting, and it happens when the model is too simple and can not learn the pattern of the data.

In financial forecasting, overfitting is especially problematic because stock prices and returns are noisy and non-stationary. A pattern learned during the training period may not persist in the future.

To reduce even more the overfitting of the model, the early stopping method can be used. With it the model automatically, when has reached a sufficient number of epochs without seeing an improvement on the loss function, ends the training phase (Martínez-Barbero, Cervelló-Royo, & Ribal, 2025).

For this reason, the use of a validation set, early stopping and appropriate hyperparameter tuning is essential to improve the robustness of the model.

In order to define the hyperparameters and input features of the LSTM model, one stock from the dataset was selected as a representative asset for the preliminary tuning phase. The model was trained and adjusted on this stock by monitoring its performance on the validation set, until a stable and acceptable configuration was obtained. Once a stable configuration was identified, the same model structure was applied to the remaining stocks in the dataset (Van-Dai, Chuan-Ming, & Direselign, 2020).

Another relevant aspect of the structure of the model, is the characteristics of the output result. For example, being the output a return, it is important to set the correct forecasting horizon, that defines the future period that the model aims to predict (Abdellilah, Abdellah, & Abdellah, 2023).

In financial applications, the target variable may refer to the next time step (usually a day) or to a longer prediction horizon (the next three or five days for example). The choice of the forecasting horizon affects the learning task, since short-term and longer-term predictions may involve different levels of uncertainty.

For the predicted return the forecasting horizon can refer to: the single future daily return or the cumulative return over the selected future period.

Therefore, a one-day horizon ($t + 1$) corresponds to the return observed on the following trading day (single daily return), while three-day and five-day horizons ($t + 3; t + 5$) could refer either to the cumulative return over the following three and five trading days respectively, or the single predicted return of the third and of the fifth day from the end of the data set.

In this work, the LSTM model is going to predict the cumulative return of the next 3 days, then it will be converted into a daily return and used inside the Markowitz approach to compute the best composition of the portfolio that maximizes the Sharpe ratio.

In the next chapter there will be a focus on the hyperparameters used, the input variables chosen and the general structure of the model (the composition of the data set and the general functioning of the model's framework).

CHAPTER 4 – Practical application of the LSTM model for returns prediction

4.1. *Construction of the data set*

The data used to train the model have been obtained from the Bloomberg Terminal provided by the university. The sample covers the period from the beginning of January 2015 to the end of May 2026 and consists of daily adjusted closing prices and the trading volumes (necessary for some of the features used in the model) for each of the selected stock.

The data were arranged chronologically, from the oldest observation to the most recent one. This is particularly important in financial forecasting, where each observation depends on the information available in the previous period. In fact, the LSTM model has been chosen for this reason, because it is specifically designed to learn from ordered sequences to predict future values.

The use of the adjusted closing prices is particularly relevant in financial time series analysis because it allows the construction of a more consistent historical price series, and this kind of price is originated from a manipulation of the stock's closing price.

The original closing price represents the final price at which the stock has been traded during regular market hours on a specific trading day.

However, although the closing price reflects the actual market price at that date, it may not be fully comparable over time when corporate actions occur, because it does not provide the most accurate valuation of the stock since it will not account for those actions that could have caused the price to shift. Events such as dividend payments, stock splits and reverse stock splits can generate mechanical changes in the observed stock price that do not necessarily reflect a real change in the economic value of the company. For this reason, adjusted closing prices are preferred when computing historical returns, as they reduce artificial discontinuities in the price series and make the data more suitable for quantitative modelling.

The main corporate actions that are incorporated in the adjusted closing prices are:

- *The payment of dividends.* A dividend is the distribution of a part of the profits earned by a company to its shareholders. When a stock goes ex-dividend, its market price generally decreases, this happens to reflect the fact that the right to receive the dividend is no longer attached to the share. If this effect is not adjusted, the price series could incorrectly show a decline that is not entirely due to market performance of the company.
- *The stock splits.* A stock split is a corporate action that takes place when a company, to increase the number of outstanding shares, will split the existing shares into a larger number of shares. This procedure results in a decrease in the market price of the individual shares and an increase in the number of the outstanding shares. Usually, it is made with the attempt to

decrease the market price of individual shares for investors. It is important to specify that, when the stock split is made, the overall economic value of the investor's position remains unchanged since he/she does not sell or buy units of that stock;

- *The reverse stock splits.* This happens when the company want to increase the price of each single stock by reducing, proportionally, the number of the already outstanding shares. Like in the previous case, the economic value of the investor's position is not directly altered by the operation itself, if the investor leaves his/her position unchanged.

The adjusted closing price is important because it gives investors a more current and accurate idea of the stock's price. It informs investors of any calculations after a corporate action meaning that it is a more reliable basis for calculating daily returns, which are then used as inputs for the LSTM model. (CFI-Team, 2020).

This is important because the LSTM model is trained to learn patterns from historical financial data, and artificial jumps, that are related only to corporate actions, in the price series could distort the learning process. By using adjusted prices, the return series will better reflect the actual performance of the stocks over time.

The selected stocks are those included in the Eurostoxx50 index. The data have been downloaded at the beginning of June 2026 from the Bloomberg Terminal. At that time the index was composed by the following stocks: ING Groep NV, Koninklijke Philips NV, Stellantis NV, BASF SE, BAYER AG, Allianz, Adidas AG, Siemens Ag Class N, Deutsche Post AG, Mercedes Benz Group AG, Deutsche Boerse AG, Air Liquide SA, Axa SA, Danone SA, BNP Paribas SA, LVMH Moet Hennessy Louis Vuitton SE, L'Oréal SA, Schneider Electric SE, EssilorLuxottica SA, Pernod Ricard SA, Airbus Group SE, Sanofi SA, TotalEnergies SE, Vinci SA, Banco Bilbao Vizcaya Argentaria, Industria de Diseno Textil SA, Iberdrola, Banco Santander, Kone Corporation, Bayerische Motoren Werke AG, Deutsche Telekom AG, Infineon Technologies AG, Munchener Ruck AG, SAP SE, Kering SA, Koninklijke Ahold Delhaize NV, ASML Holding NV, Intesa Sanpaolo SpA, Enel, Eni SpA, Hermes International SCA, Safran SA, Anheuser Busch Inbev SA NV, Volkswagen AG Vz, Argenx SE and Rheinmetall AG.

5 stocks have been removed for missing data, they are: Flutter Entertainment PLC, Ferrari NV, Adyen NV, Volkswagen AG NV and Prosus NV.

To avoid the problem of different trading calendars, that could affect the construction of the multi-asset return matrix for the covariance estimation, only for this part the script will retain only the common trading dates among all the stocks. This procedure ensures that the covariances are estimated using returns observed on the same calendar days across all the assets. To do this the return series were converted into a MATLAB timetable and synchronized using the intersection of their dates, leaving only the common trading dates among all the stocks.

For the training of the LSTM model, instead, cross-stock date alignment was not required. This because the model was trained separately for each stock, preserving its own chronological order. In this case, each stock was treated as an individual time series.

Therefore, while the temporal sequence of observations remains essential for the LSTM learning process, it is not necessary that all stocks share the same trading dates during the individual training phase.

The stocks were ordered alphabetically, at the beginning of the process, only for reporting purposes, in order to ensure a consistent and easily comparable structure across the different output tables. This ordering is very important also to be sure to have the correct match between the covariance matrix and the predicted returns for the estimation of the stocks' weights for the portfolio optimization.

This ordering has no impact on the training process or on the predictive performance of the LSTM model.

Since the objective of the model was the portfolio optimization (that is based on the predicted returns and the historical standard deviation of each stock) the model was built to predict next return directly for all the stocks. So, the adjusted closing prices were first transformed into logarithmic returns.

This transformation was introduced, also, because financial prices are generally non-stationary and often display strong upward or downward trends over long periods, making the prediction difficult. If the model were trained directly on prices, the result could be that it learns the scale and level of the training period instead of learning the underlying short-term dynamics of the stock. This would make the model less able to generalize the future periods, especially when the validation and test prices are higher or lower than those observed during the training window.

Returns, instead, are generally more stationary than price levels. They tend to fluctuate around a relatively stable mean rather than exhibiting the persistent trends observed typically in stock prices. Although their variance may still change over time, returns are less dependent on the absolute price level and therefore represent a more appropriate target for short-term financial forecasting.

4.2. Input features used in the model

The input variables of the LSTM model were constructed from the stock price and trading-volume series.

Firstly, let (P_t) denote the price of a stock at trading day (t) , and let (V_t) denote its corresponding trading volume.

The model used eight input features designed to capture recent price movements, momentum over different time horizons, changes in market activity, and time-varying volatility.

Each feature was calculated for every available trading day, and the resulting feature observations were subsequently organized into rolling sequences of five consecutive trading days.

This means that each input sequence provided to the LSTM model contained eight variables observed over five-time steps, which are necessary for the model to understand correctly the movement of the data.

The eight features used in the model are the following.

One-day logarithmic return. The first feature is the most recent daily logarithmic return, that is defined by eq. (25).

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right) \quad (25)$$

where P_t is the price at time t , and P_{t-1} is the price at time $t - 1$. Eq. (25) being a logarithmic return can be rewritten as eq. (26), that is:

$$r_t = \ln(P_t) - \ln(P_{t-1}) . \quad (26)$$

This variable measures the proportional change in the stock price between two consecutive trading days, with a positive value it indicates that the price has increased, while with a negative value it indicates that the price has decreased.

The daily logarithmic return provides the model with information about the most recent price movement, and this may be useful because short-term stock behaviour can exhibit both continuation and reversal effects. Logarithmic returns were preferred to raw price changes because they are independent of the absolute price level of the stock and they are additive. So, they can be used consistently across securities with very different price levels.

Five-day momentum. The five-day momentum feature is defined as the sum of the daily logarithmic returns observed over the most recent five trading days. It is given by eq. (27), that is:

$$MOM_{5,t} = \sum_{j=0}^4 r_{t-j} . \quad (27)$$

Where $\sum_{j=0}^4 r_{t-j}$ is the cumulative logarithmic return of the last 5 available days.

Being the returns logarithmic, eq. (27) can be written as eq. (28), that is:

$$MOM_{5,t} = \ln\left(\frac{P_t}{P_{t-5}}\right) . \quad (28)$$

This feature measures the cumulative price movement over a short five-day period, a positive value indicates that the price at time (t) is higher than it was five trading days earlier, or a negative value indicates an overall decline. It summarizes the recent direction of the stock more effectively than a single daily return because it reduces the importance of isolated one-day fluctuations.

Ten-day momentum. The ten-day momentum feature is the analogous to the five-day momentum feature, with the difference that it considers the last 10 available trading days. It is calculated by eq. (29) that can be rewritten as eq. (30):

$$MOM_{10,t} = \sum_{j=0}^9 r_{t-j} ; \quad (29)$$

$$MOM_{10,t} = \ln \left(\frac{P_t}{P_{t-10}} \right) . \quad (30)$$

This variable measures the cumulative logarithmic return over the most recent ten trading days, and it captures a slightly longer price trend than the five-day momentum indicator.

The simultaneous use of five-day and ten-day momentum allows the network to compare price movements over different short-term horizons. For example, a stock may show a positive five-day momentum but a negative ten-day momentum, this configuration could indicate that a recent recovery is occurring within a broader declining trend.

Twenty-day momentum. Approximately twenty trading days correspond to one trading month, and this feature is used to represent the medium-term price behaviour relative to the shorter five-day and ten-day indicators.

The twenty-day momentum feature is defined as eq. (31) and it can be rewritten as eq. (32):

$$MOM_{20,t} = \sum_{j=0}^{19} r_{t-j} ; \quad (31)$$

$$MOM_{20,t} = \ln \left(\frac{P_t}{P_{t-20}} \right) . \quad (32)$$

A positive twenty-day momentum value indicates that the stock experienced an overall increase during the recent twenty-day window, while a negative value indicates an overall decrease.

By including this variable, the LSTM can determine whether short-term price movements are consistent with or opposed to the longer monthly trend. For example, negative five-day momentum combined with positive twenty-day momentum may represent a temporary decline within a more persistent upward movement.

Sixty-day momentum. The longest momentum measure is calculated over sixty trading days, and it summarizes the price movement over an approximate three-month trading period.

It is expressed by eq. (33) or by the equivalent eq. (34):

$$MOM_{60,t} = \sum_{j=0}^{59} r_{t-j} ; \quad (33)$$

$$MOM_{60,t} = \ln \left(\frac{P_t}{P_{t-60}} \right) . \quad (34)$$

The inclusion of sixty-day momentum gives the model information about the position of recent price movements within a broader historical context. While five-day momentum reacts rapidly to new price movements, sixty-day momentum changes more gradually and is less affected by isolated daily fluctuations.

Change in logarithmic trading volume. Before constructing the volume-based feature, the trading volume needs to be transformed logarithmically (eq. (36)), and then the change in the logarithmic volume is computed by eq. (37).

$$L_t = \ln(\max(V_t, 1)) ; \quad (36)$$

$$\Delta L_t = L_t - L_{t-1} . \quad (37)$$

In eq. (36) the logarithmic value of the volume is computed as the maximum between the observed volume and one, this method is used to prevent computational problems if a zero-volume observation is provided, since the logarithm of zero is undefined. A positive value indicates that the trading volume has increased relative to the previous day, while a negative value indicates that the trading activity has decreased.

The logarithmic transformation is used to reduce the influence of extremely large observations and makes relative changes more comparable across stocks. For example, an increase from one million to two million traded shares has the same logarithmic change as an increase from five million to ten million shares, in this way it is considered the proportion of the change in the traded amount and not the absolute amount. The information contained in the change of the volume are related to the intensity and credibility of price movements. A large return accompanied by a substantial increase in volume may reflect stronger market participation than an equivalent return generated under unusually low trading activity.

Exponentially weighted volatility. The model includes an exponentially weighted measure of recent return volatility. The conditional variance is updated recursively according to eq. (38), that is:

$$\sigma_{EWMA,t}^2 = \lambda \sigma_{EWMA,t-1}^2 + (1 - \lambda) r_t^2 . \quad (38)$$

Where r is the return at time t , and λ is the decay parameter and it is equal to 0.94. The decay factor was chosen by following the Risk-Metrics methodology developed by J.P. Morgan, that estimated 0.94 as the decay factor for daily financial return data, while a value of 0.97 was adopted for monthly data. The value of 0.94 provides a compromise between persistence and responsiveness, this because most of the previous variance estimate is retained, while recent squared returns are receiving sufficient weight to allow the volatility estimate to react to changes in market conditions (J.P.Morgan/Reuters, 1996). The initial value of the variance at time $t = 0$ is equal to eq. (39):

$$\sigma_{EWMA,1}^2 = r_1^2 . \quad (39)$$

Where λ tells us that 94% of the previous variance estimate is retained at each update, while 6% of the new estimate is determined by the most recent squared return, so the EWMA procedure assigns greater importance to recent observations, while the influence of older returns declines exponentially over time. This feature is included because financial returns often exhibit volatility clustering. A volatility cluster is a period that is characterized by large price movements that tend to be followed by other periods of high volatility, while calm periods tend to be followed by other calm periods.

EWMA volatility provides the LSTM with information about the current risk regime of the stock and adjusts more rapidly to recent market shocks than a simple long-window standard deviation. A high EWMA volatility value indicates that the stock has recently experienced relatively large price fluctuations, on the other hand a low value indicates that recent returns have been comparatively stable.

Relative twenty-day trading volume. This input feature compares the current logarithmic volume with its average value over the most recent twenty-day window, and it is given by eq. (40) and eq. (41):

$$L_t = \ln(\max(V_t, 1)); \quad (40)$$

$$VolRel_{20,t} = L_t - \frac{1}{20} \sum_{j=0}^{19} L_{t-j} . \quad (41)$$

Where L_t is the current logarithmic volume, and eq. (41) is the difference between L_t and the average value of the last 20 days window of logarithmic volumes.

A positive value indicates that the current logarithmic trading volume is higher than its average level within the twenty-day window, and a negative value indicates that current activity is lower than the recent reference level.

This feature differs from the daily change in logarithmic volume, because the daily volume change compares the current observation only with the immediately preceding day, whereas the relative volume compares the current trading activity with a longer twenty-day benchmark.

For example, the trading volume may increase relative to the previous day but still remain below its twenty-day average level.

The final feature set was designed to represent four main dimensions of stock-market behaviour.

The one-day logarithmic return captures the most recent price shock.

The choice of using momentum indicators with four different horizons allows the network to observe the term structure of recent price performance. The LSTM can therefore distinguish situations in which short-term, medium-term, and longer-term trends point in the same direction from situations in which they provide conflicting signals.

The two volume-based indicators measure both immediate changes in market participation and deviations from the recent average level of trading activity.

And finally, the EWMA volatility represents the current level of time-varying risk.

The presence of several momentum horizons allows the model to observe whether recent trends are consistent across time or whether short-term movements diverge from longer-term behaviour. Similarly, the simultaneous use of volume change and relative volume allows the model to distinguish between a daily acceleration in trading activity and an unusually high or low level of activity relative to the recent historical average.

For every forecasting observation, the LSTM receives these eight features for each of the five trading days included in the rolling sequence. The input associated with one observation can therefore be represented as an (8×5) matrix, that is equal to table 1.

$$X_t = \begin{bmatrix} r_{t-4} & r_{t-3} & r_{t-2} & r_{t-1} & r_t \\ \text{MOM}_{5,t-4} & \text{MOM}_{5,t-3} & \text{MOM}_{5,t-2} & \text{MOM}_{5,t-1} & \text{MOM}_{5,t} \\ \text{MOM}_{10,t-4} & \text{MOM}_{10,t-3} & \text{MOM}_{10,t-2} & \text{MOM}_{10,t-1} & \text{MOM}_{10,t} \\ \text{MOM}_{20,t-4} & \text{MOM}_{20,t-3} & \text{MOM}_{20,t-2} & \text{MOM}_{20,t-1} & \text{MOM}_{20,t} \\ \text{MOM}_{60,t-4} & \text{MOM}_{60,t-3} & \text{MOM}_{60,t-2} & \text{MOM}_{60,t-1} & \text{MOM}_{60,t} \\ \Delta L_{t-4} & \Delta L_{t-3} & \Delta L_{t-2} & \Delta L_{t-1} & \Delta L_t \\ \text{EWMAVol}_{t-4} & \text{EWMAVol}_{t-3} & \text{EWMAVol}_{t-2} & \text{EWMAVol}_{t-1} & \text{EWMAVol}_t \\ \text{VolRel}_{20,t-4} & \text{VolRel}_{20,t-3} & \text{VolRel}_{20,t-2} & \text{VolRel}_{20,t-1} & \text{VolRel}_{20,t} \end{bmatrix}$$

Table 1: Each row represents one input feature, while each column corresponds to one of the five consecutive trading days included in the input sequence, from $t - 4$ to t .

The LSTM model does not observe only the latest value of each indicator, instead it receives the recent five-day evolution of all eight variables, allowing the LSTM to process both the level of each indicator and its temporal development before producing the forecast of the cumulative logarithmic return over the following three trading days.

4.3. Construction of the LSTM model used

The configuration of the LSTM model adopted in this study was determined through an iterative empirical calibration procedure conducted on one reference stock from the dataset, that is: ASML Holding NV. The choice of this stock was not driven by any specific financial reason; instead, it followed a neutral and reproducible selection rule, as ASML Holding NV was the first security in the alphabetically ordered list of stocks. The purpose of this choice was to define a common model configuration that could then be applied consistently to all the stocks in the sample.

The construction of the model followed a sequential procedure composed of different steps, that are: data preparation (explained in the previous paragraphs), feature engineering, target variable definition, data-set splitting, normalization, model architecture design, hyperparameter calibration, training, and final evaluation.

The main characteristics of the model were selected through an iterative trial-and-error analysis.

During the tuning phase, different combinations of input features and model hyperparameters were tested. In particular, the analysis considered alternative configurations of input features variables, different numbers of LSTM hidden units, different maximum numbers of epochs, and different training settings (forecasting horizon or input sequence-length). The objective was not to perfectly optimize the model on a single stock, but to identify a configuration that provides a reasonable balance

between predictive performance, stability, and generalization capacity. The final configuration was then applied to the entire dataset in order to evaluate the performance of the LSTM model across different stocks under the same methodological framework.

The model was trained using the Adam optimization algorithm, which is commonly used in Neural Network applications because it adapts the learning rate during training and generally provides stable convergence.

Early stopping was also included through the validation set, and the model corresponding to the best validation performance was retained. This mechanism is particularly relevant in financial applications, where overfitting is an important problem. Since stock returns are noisy and non-stationary, a model may easily learn patterns that are specific to the training period but do not persist in the future, the use of a validation set, dropout, normalization, and early stopping was therefore intended to improve the robustness of the model.

After the data had been collected (explained in the previous paragraph), the next step was the construction of the input variables. The final feature set consisted of eight variables: the most recent daily logarithmic return; cumulative momentum over the previous 5, 10, 20, and 60 trading days; the daily change in logarithmic trading volume; an exponentially weighted volatility measure; and a relative volume indicator comparing current trading volume with its average level over the previous 20 trading days. These selected variables were intended to provide the LSTM with information concerning: recent price movements, short-term and medium-term momentum, changes in market activity, and time-varying volatility.

The selected model should not be interpreted as the only configuration examined during the analysis, but it is the specification that has provided the most satisfactory balance between predictive information, model complexity, stability across random initializations, and portfolio performance. The portfolio performance had been measured by using: RMSE and MAE values, the Directional Accuracy and the Correlation coefficient between the predicted returns and the actual returns (the performance measures will be analysed more in detail later).

The target variable was defined as the cumulative logarithmic return over the chosen forecasting horizon (3 days). In this way, the model was trained to predict the future return of the stock over a short forward-looking window, instead of predicting the next price level. This approach is more coherent with the financial objective of the study, because investment decisions are usually based on expected returns and risk rather than on the absolute level of the price. Once the cumulative predicted log-return is obtained, it is converted into an equivalent daily simple return, to make the interpretation of the results more intuitive, useful for the portfolio optimization because in this way the expected return and the historical risk measure are on a consistent daily scale.

More specifically, the cumulative log-return is first divided by the prediction horizon and

subsequently transformed into a simple return according to eq. (42):

$$r_{daily} = e^{\frac{R_3^{log}}{3}} - 1. \quad (42)$$

This represents the constant daily simple return that, if compounded over the forecasting horizon, would generate the same cumulative return implied by the model prediction.

Since the LSTM receives sequences as input, the dataset was transformed into rolling windows. Each input observation was composed of a fixed number of previous trading days (the “*sequenceLength*” that is equal to 5 days), while the corresponding output was the future cumulative return over the selected horizon of 3 days. Meaning that, for each stock, the model used a sequence of past information (5 days) to estimate the cumulative return (of 3 days) that would occur after the end of that sequence. The use of rolling windows allowed the model to exploit the temporal structure of the data and to learn possible relationships between past market conditions and future movements.

The data for each stock were then divided into training, validation, and test sets according to a chronological order. The first 80% of the observations were used as the training set, while the remaining 20% were equally divided between the validation and test sets, each containing 10% of the total observations.

Before training the LSTM, the input variables were normalized. The normalization parameters (that are the mean and the standard deviation) were computed only on the training set and then applied also to the validation and test sets, and the same logic was also applied to the target variable. This procedure was necessary to avoid data leakage, and it is particularly important for Neural Networks, because variables measured on different scales may lead to unstable gradients and inefficient learning. The architecture adopted in this study is based on a standard LSTM regression structure, and it is the following one.

The first layer, called input layer, is a sequence input layer. It receives the selected financial features for each time step of the rolling window. This layer is, then, followed by an LSTM layer, whose role is to process the temporal dependencies contained in the input sequence, and it is composed of 128 hidden units. The LSTM layer was set with output mode equal to “last”, meaning that only the final hidden state of the sequence is passed to the following layers. This choice is appropriate because the aim of the model is to produce one prediction for each input sequence, that is the cumulative logarithmic return over the following three trading days

A dropout layer, with a dropout probability of 0.1, was then introduced in order to reduce the risk of overfitting by deactivating part of the network during training. More specifically, during training the dropout layer randomly set a proportion (10%) of the activations produced by the LSTM layer to zero. In practical terms, some of the information extracted by the LSTM was temporarily ignored before being passed to the fully connected layer. The set of deactivated activations changed across

training iterations, meaning that the network could not consistently rely on the same internal signals to generate its predictions. This mechanism encouraged the LSTM model to distribute the learned information across different hidden units and to develop more robust representations of the relationships between the input features and predicted returns. As a result, the dropout reduces the probability that the model will memorize patterns that are specific to the training sample, thereby improving its ability to generalize previously unseen observations. The dropout mechanism was active only during the training phase and did not alter the network output during validation, testing, or forecasting.

Since the output generated by the LSTM layer did not directly represent the final forecast it needs to be processed more. In the final hidden state of the LSTM, with 128 hidden units, it is composed of 128 numerical values summarizing the temporal information extracted from the five-day input sequence.

This representation is first passed to a fully connected hidden layer containing 64 units, which role is to combine and reprocess the information generated by the different LSTM hidden units, allowing the model to identify which temporal patterns were most relevant for forecasting the future return.

A ReLU activation function was subsequently applied to introduce an additional nonlinear transformation, allowing the network to represent more complex relationships between the temporal information extracted by the LSTM and the future behaviour of the stock return.

Finally, a second fully connected layer reduced the 64 hidden activations to a single numerical value, corresponding to the predicted cumulative logarithmic return over the following three trading days.

After the training phase, the model was evaluated on the test set. The evaluation was based on different performance metrics in order to capture both the numerical accuracy and the financial usefulness of the predictions. RMSE and MAE were used to measure the average prediction error, while the Directional Accuracy was used to evaluate the ability of the model to correctly predict the sign of future returns, and the Correlation between predicted and actual cumulative returns was computed in order to assess whether the model was able to capture the movement of the target variable over time. These metrics were calculated out-of-sample, meaning that they were obtained on observations that had not been used during training or model selection.

Overall, the LSTM model was constructed with the aim of learning short-term temporal relationships between past market information and future stock returns. The final model was not designed to memorize the behaviour of one specific stock, but to provide a consistent forecasting framework applicable to all the securities included in the dataset. The use of return-based targets, rolling input sequences, technical and volatility-based features, chronological splitting, normalization, validation-based model selection, and out-of-sample testing allowed the analysis to reduce the risk of data leakage and to obtain a more realistic assessment of the predictive capacity of the model.

4.4. Model's parameters

The final LSTM model is defined through a set of parameters that set the structure of the input data, the architecture of the Neural Network, and the training process, and these parameters are constant across all the stocks included in the dataset so that the predictive results will be compared under the same methodological framework.

The *sequence length* was set equal to five trading days. It determines how much historical information the LSTM receives before producing a forecast. In this case, the model used the information available from $(t - 4)$ to (t) to predict the return following the end of the sequence.

A relatively short input window (one week of trading activities) was used to make the model able to focus on recent market conditions and short-term relationships. So, each observation provided to the LSTM model consists of eight selected financial features measured over five consecutive time steps. The *forecasting horizon* was set equal to three trading days. The target variable of the LSTM model is the cumulative logarithmic return observed during the three trading days immediately following the input sequence. The model does not predict three separate daily returns, but it generates one numerical value representing the total log-return over the complete forecasting horizon. The predicted cumulative log-return is subsequently converted into a cumulative simple return and an equivalent daily simple return for financial interpretation and portfolio construction.

The dataset was divided, by following the chronological order, into *three subsets*. The first 80% of the observations formed the training set, the following 10% formed the validation set, and the final 10% formed the test set. The training set was used to estimate the network parameters, while the validation set was used to monitor the model during training and determine which network should be retained. The test set was kept separate and was used only for the final out-of-sample evaluation. Since financial data are ordered through time, the observations are not randomly shuffled, preserving their chronological order.

Before training, the data had been *normalized*. Both the input variables and the target variable had been standardized. The mean and the standard deviation used for normalization had been calculated exclusively from the training set and were then applied to the training, validation and test sets. Since normalization was performed before the data entered the network, the normalization option of the sequence input layer was set to “*none*”, this is necessary to prevent the data from being normalized a second time.

The LSTM layer's *number of hidden units* was set equal to 128. This number determines the size of the internal representation created by the recurrent layer, each hidden unit contributes to summarizing information extracted from the five-day sequence, and the complete final hidden state therefore contains 128 numerical activations. A larger number of hidden units increases the capacity of the

model to learn complex temporal relationships, although it also increases the number of parameters built and the potential risk of overfitting. In this case, in fact, a larger number of hidden units was unnecessary to improve the extraction of information of the model, reducing the prediction accuracy of the model.

The *output mode* of the LSTM layer is set equal to “last”. This means that only the output associated with the final time step of the sequence is transferred to the subsequent layers. This setting was chosen because the model was designed as a sequence-to-one forecasting model: each five-day input sequence was associated with one cumulative three-day return.

A *dropout rate* of 0.1 was applied after the LSTM layer. During training, the dropout layer randomly set 10% of the activations generated by the LSTM to zero, and the set of omitted activations changes across training iterations.

The output of the LSTM was then passed to a *fully connected hidden layer* containing 64 units. This layer combines and reprocesses the 128 activations generated by the previous LSTM layer, producing a more compact representation of the temporal information considered useful for the forecast.

A *ReLU activation function* is then applied after the fully connected layer. The ReLU function introduces nonlinearity by retaining positive activation values and setting negative values to zero, enabling the network to represent more complex relationships between the temporal patterns extracted by the LSTM and the future stock return.

After the ReLU was set a *second fully connected layer* with one output unit. Its role is to transform the 64 hidden activations into a single numerical value corresponding to the predicted cumulative logarithmic return over the following three trading days.

After that, the *final layer* had the role to make the final prediction understandable. To this layer, in fact, had been specified the fact that the target is a continuous numerical variable. It has, also, the role to evaluate the difference between the predicted and the actual returns during the training process.

The model was trained using the *Adam optimization algorithm*, which updates the network weights by combining information about the average gradient and the average squared gradient, and this allows the effective learning rate to adapt across network parameters and generally provides a stable optimization procedure for neural networks.

The *initial learning rate* was set equal to: $1e^{-4}$. The learning rate controls the magnitude of the changes applied to the weights at each training step. A value that is too high may cause unstable training or prevent convergence, while a value that is too low may make the learning process excessively slow. The selected value allowed the model parameters to be updated gradually during training.

The *mini-batch size* was set equal to: 32. This means that the network processed 32 input sequences before updating its weights, their role during the training represents a compromise between

processing one sequence at a time and using the entire training set for every update. It reduces computational requirements while retaining a certain degree of variability in the gradient estimates. The maximum number of *training epochs* was set equal to 200. One epoch corresponds to one complete pass through the entire training set, so the maximum number establishes an upper limit on the duration of training, although the procedure could terminate earlier because of the early-stopping mechanism.

The *validation-frequency parameter* was set equal to 30 training iterations. This means that the validation loss had been evaluated every 30 training iterations.

The *validation patience* was also equal to 30, meaning that the training of the model could be stopped if the validation performance failed to improve for 30 consecutive validation checks. This is the early stopping criteria, and it limits the probability that the model would continue adapting to the training sample after its out-of-sample performance had begun to deteriorate.

The final network was selected using the best-validation-loss criterion, consequently the retained model was not necessarily the network obtained after the final training iteration, but it is the one with the lowest loss value. In fact, the LSTM model had returned the network configuration associated with the lowest validation loss observed during training, and this ensured that the final model corresponded to the point at which the validation performance was strongest.

A *gradient threshold* equal to 1 had been introduced to reduce the risk of exploding gradients. During Recurrent Neural Network training, gradients may occasionally become excessively large, producing unstable weight updates, so the gradient threshold limits their magnitude and helps stabilize the optimization process.

In the model had been used the *L2 regularization*, that was set equal to $1e^{-4}$. It introduces a penalty for excessively large network weights in the objective function, so this discourages the model from relying on extreme parameter values and provides an additional mechanism for reducing overfitting. Finally, the complete estimation procedure was repeated using *ten random seeds*, numbered from 1 to 10. Neural Network training is stochastic because the initial values of the weights and some training operations depend on random-number generation, so the same architecture may produce slightly different results across independent training runs. Repeating the procedure with ten seeds made it possible to evaluate the stability of the predictive results.

Performance measures (MSE and MAE values, the Directional Accuracy and the Correlation coefficient) were first calculated separately for every stock and seed. Their mean and standard deviation were subsequently computed to summarize average performance and sensitivity to random initialization.

For the portfolio optimization the resulting predictions for every stock and seed was used as the expected return estimate by making the mean of the predictions for each stock and seed concluding

with a single value predicted for each stock. This has reduced the dependence of the final results on a single training run.

For the portfolio back-test, the out-of-sample predictions obtained from the ten training runs were combined into a single ensemble forecast. More specifically, for each stock and for each identical pair of forecast and target dates, the model produced ten predicted returns, one from each random seed. The forecast date represented the last trading day included in the five-day input sequence, and the target date represented the final day of the subsequent three-day forecasting horizon.

The ten predictions were averaged only when they referred to the same stock and exactly the same forecasting period, otherwise the predictions were referred to the same stocks but to different dates. For stock (i), forecast date (t), and target date (T), the mean prediction was calculated by eq. (43), that is:

$$\bar{\hat{r}}_{i,t,T} = \frac{1}{10} \sum_{s=1}^{10} \hat{r}_{i,t,T}^{(s)} . \quad (43)$$

Where $\hat{r}_{i,t,T}^{(s)}$ is the return predicted by the model trained with seed (s) and $\bar{\hat{r}}_{i,t,T}$ is the average value of all the returns predicted for the same stock, for the same date but for different seeds.

For example, if the ten models have generated ten different predictions for stock “A” using information available up to 10 April 2025 and forecasting the cumulative three-day return ending on 15 April 2025 (12 and 13 April 2025 were Saturday and Sunday, so both are not in the sample), those ten values are averaged to obtain one final prediction for stock “A” for that specific period, and the same procedure was repeated independently for every stock and every available forecast-target date pair.

The resulting ensemble prediction was then used as the expected returns estimate for the back-test of the portfolio. For the back-test it was compared the result of the portfolios updated continuously by following: the LSTM model prediction, the historical average value and an equally weighted portfolio that will not change the weights of the assets (Abdellilah, Abdellah, & Abdellah, 2023).

The results of the portfolio optimization and of the back-test will be provided in the next chapter.

Overall, the selected parameters define a relatively compact sequence-to-one LSTM model designed to capture short-term temporal relationships while controlling overfitting through dropout, L2 regularization, chronological validation, gradient clipping, early stopping, and averaging across multiple random seeds.

4.5. Results of the LSTM model

The predictive performance of the LSTM model has been evaluated on the chronologically separated test set using four complementary measures: Root Mean Squared Error (RMSE), Mean Absolute

Error (MAE), Directional Accuracy, and the Correlation between predicted and actual cumulative three-day logarithmic returns.

Since the training procedure was repeated using ten seeds (from 1 to 10), the metrics were first calculated separately for every stock and seed and then were averaged across the ten training runs, the results are shown on table 2.

The *Root Mean Squared Error*, that is provided by eq. (44), measures the square root of the average squared difference between the predicted and actual returns.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2} \quad (44)$$

where $\hat{y}_i$ is the predicted value computed by the LSTM model and y_i is the actual return of the test set.

Since larger errors are squared the RMSE assigns greater importance to unusually inaccurate forecasts. Across the 45 stocks included in the dataset, the average RMSE is approximately 0.0301, while the median is 0.0286, and the values ranges from 0.0168 for Iberdrola to 0.0573 for Stellantis. This indicates that on average the prediction error is approximately three percentage points in terms of the cumulative three-day logarithmic return, although the magnitude of the error varies considerably across all the stocks.

On the other hand, the *Mean Absolute Error* provides a more direct interpretation of the average forecasting error because it does not assign disproportionate weight to the largest deviations, it is computed by eq. (45).

$$MAE = \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i| \quad (45)$$

where $\hat{y}_i$ is the predicted value computed by the LSTM model and y_i is the actual return of the test set. So, the MAE is the summation of the absolute differences between $\hat{y}_i$ and y_i .

The mean MAE across stocks is approximately 0.0227, with a median value of 0.0220. Meaning that, on average, the predicted cumulative three-day logarithmic return differs from the observed return by approximately 2,30 %. The lowest MAE was recorded for Iberdrola, with a value of approximately 0.0131, whereas the highest value was observed for Stellantis, with a value of approximately 0.0415. These error measures should not be interpreted as pure indicators of forecasting ability of the LSTM model across stocks. Both are scale-dependent metrics, for example they are naturally higher for securities characterized by greater return volatility. This means that a lower error for a relatively stable stock does not necessarily imply that the model learned its dynamics more effectively than those of a more volatile stock. This can partly explain why stocks such as Iberdrola, Air Liquide, and Enel displayed relatively low error measures (these stocks are more stable), while more volatile

securities such as Stellantis, Rheinmetall, and Infineon Technologies produced higher RMSE and MAE values.

Directional Accuracy was used to evaluate whether the model correctly predicted the sign of the future cumulative return, without considering the exact magnitude of the returns. Meaning that it does not distinguish between a forecast that predicts the correct sign but that is numerically close to the actual return or one that overestimates or underestimates the magnitude of the actual return.

The mean Directional Accuracy across stocks and seeds was 51.64%, while the median was 51.09%. Thirty of the 45 stocks obtained an average Directional Accuracy above 50%, and only four securities exceeded 55%. The highest values were observed for Axa, Schneider Electric, Enel, and Iberdrola. In particular, Axa achieved an average Directional Accuracy of approximately 57.94%, while Schneider Electric reached approximately 57.17%.

These results indicate that the model displayed a limited but positive ability to identify the direction of future returns. In fact, an accuracy slightly above 50% suggests that the forecasts contain some directional information, but the improvement relative to a random directional prediction remained modest for most stocks.

The *Correlation coefficient* was used to assess whether predicted and actual returns moved together over time. The average stock-level correlation across the ten seeds was approximately 0.0387, with a median of 0.0283. Thirty-five of the 45 stocks presented a positive average correlation, but only six obtained a value equal to or greater than 0.10. The highest correlations were recorded for Axa and Enel, both at approximately 0.143, followed by Schneider Electric, Eni, BNP Paribas, and Banco Bilbao Vizcaya Argentaria.

The predominantly positive but relatively low correlation values suggest that the model was able to capture part of the temporal variation in future returns, but its ability to reproduce their exact movements and magnitudes was weak.

This result is not unexpected in short-term financial forecasting. This is because stock returns are highly noisy and are affected by information that cannot be inferred exclusively from historical prices, volume, momentum, and volatility indicators. Despite this, a positive correlation is equally relevant because it indicates that the predicted returns magnitude is following that of the actual returns.

The results also reveal substantial differences across stocks; for example, Axa and Enel performed relatively well according to both Directional Accuracy and Correlation, suggesting that the model captured a more consistent predictive signal for these stocks. On the other hand, some securities presented Directional Accuracy below 50% or negative correlations, indicating that the learned relationships did not generalize equally well to every stock.

Probably this difference in the prediction capacity of the LSTM model is due to the use of a common architecture and set of hyperparameters for all securities to provide a methodological consistency, but

it did not guarantee identical predictive performance across companies with different volatility levels, and market dynamics.

One other important measure is the standard deviation calculated across the ten seeds, it provides further information about the model stability.

The average standard deviation across seeds was approximately 0.00030 for RMSE and 0.00027 for MAE. These relatively small values indicate that the overall magnitude of the forecasting errors was stable across alternative random initializations.

By contrast, the average standard deviation was approximately 3,68 % for Directional Accuracy and 9,30 % for correlation, meaning that the measures related to the direction and magnitude of the predictions were more sensitive to random initialization than the error measures.

This result is important because it is showing that from different training runs the LSTM model usually produces similar average prediction errors, but it is not always able to identify the same directional patterns. A model, in fact, can show a stable MAE and RMSE while generating Correlations or Directional Accuracy values that varied more considerably across seeds, and this finding supports the decision of using 10 seeds to avoid relying on a single random initialization.

For the portfolio back-test, the ten predictions associated with the same stock, forecast date, and target date were averaged to obtain one ensemble forecast. When the performance metrics were recalculated using these averaged predictions, the mean stock-level Directional Accuracy increased to approximately 52.21%, while the mean Correlation increased to approximately 0.063, and the average RMSE and MAE were approximately 0.0300 and 0.0227, respectively.

The improvement was more evident for some individual securities. For example, the ensemble prediction for Enel achieved a Directional Accuracy of approximately 64.79% and a Correlation of approximately 0.290; Iberdrola achieved a Directional Accuracy of approximately 61.19%, while Intesa Sanpaolo and ING approached or exceeded 59%.

Overall, 15 stocks obtained an ensemble Directional Accuracy of at least 55%, and 15 obtained an ensemble correlation of at least 0.10.

The ensemble procedure had a greater effect on the directional and correlation measures than on RMSE and MAE. However, averaging does not necessarily generate a large reduction in the absolute prediction error when all individual networks already produce forecasts concentrated near the mean return.

Overall, the results indicate that the LSTM did not predict the exact magnitude of short-term stock returns with high precision.

These findings are especially relevant in the context of portfolio construction. Since the objective of the model was not necessarily to forecast every individual return with perfect numerical accuracy, but to generate differentiated expected-return estimates that could be combined with the covariance

matrix in the portfolio optimization process. Even a modest predictive signal may contribute to portfolio allocation if it helps distinguish securities with relatively more favourable expected returns from those with less favourable forecasts.

Assets	RMSE	MAE	Directional Accuracy	Correlation
ASMLHoldingNV	0,0407	0,0315	54,4056	-0,0232
AdidasAG	0,0375	0,0273	48,0282	-0,0262
AirLiquideSA	0,0187	0,0140	54,2657	0,0290
AirbusGroupSE	0,0286	0,0226	52,9371	0,0038
Allianz	0,0207	0,0157	54,7887	0,0711
AnheuserBuschInbevSA NV	0,0283	0,0199	48,9161	0,0624
ArgenxSE	0,0357	0,0264	52,5874	0,0182
AxaSA	0,0222	0,0166	57,9371	0,1429
BASFSE	0,0294	0,0227	48,6620	0,0016
BAYERAG	0,0413	0,0309	54,0141	0,0898
BNPParibasSA	0,0295	0,0220	54,3007	0,1051
BancoBilbaoVizcayaArg entaria	0,0303	0,0226	54,4755	0,1015
BancoSantander	0,0298	0,0227	50,1399	0,0165
BayerischeMotorenWerk eAG	0,0316	0,0242	50,5282	0,0283
DanoneSA	0,0242	0,0178	49,1958	0,0812
DeutscheBoerseAG	0,0245	0,0196	50,8803	-0,0021
DeutschePostAG	0,0264	0,0202	50,9155	-0,0019
DeutscheTelekomAG	0,0270	0,0199	51,2324	0,0411
Enel	0,0204	0,0149	55,2113	0,1421
EniSpA	0,0255	0,0198	49,0845	0,1066
EssilorLuxotticaSA	0,0331	0,0243	48,8112	-0,0424
HermesInternationalSCA	0,0324	0,0241	46,2937	0,0485
INGGroepNV	0,0265	0,0208	54,4755	0,0243
Iberdrola	0,0168	0,0131	55,2098	0,0056
IndustriaDeDisenoTextil SA	0,0282	0,0208	54,0559	0,0987
InfineonTechnologiesAG	0,0464	0,0358	54,4718	0,0412
IntesaSanpaoloSpA	0,0254	0,0195	53,2042	-0,0607
KeringSA	0,0418	0,0321	53,4266	0,0532
KoneCorporation	0,0228	0,0170	49,3594	-0,0185
KoninklijkeAholdDelhaiz eNV	0,0227	0,0159	48,0420	-0,0213
KoninklijkePhilipsNV	0,0301	0,0232	53,4965	0,0597
L Or alSA	0,0274	0,0209	50,8042	0,0147
LVMHMoetHennesyLo uisVuitto	0,0345	0,0257	48,7063	0,0233
MercedesBenzGroupAG	0,0274	0,0218	48,5563	0,0345
MunchenerRuckAG	0,0241	0,0180	51,0915	0,0011

PernodRicardSA	0,0319	0,0257	47,7972	0,0724
RheinmetallAG	0,0515	0,0398	48,9437	0,0936
SAPSE	0,0354	0,0258	47,0877	0,0264
SafranSA	0,0309	0,0225	52,5175	0,0810
SanofiSA	0,0262	0,0195	50,5944	-0,0157
SchneiderElectricSE	0,0305	0,0237	57,1678	0,1162
SiemensAGClassN	0,0313	0,0240	53,8380	-0,0059
StellantisNV	0,0573	0,0415	49,8592	0,0220
TotalEnergiesSE	0,0230	0,0181	51,0140	0,0796
VinciSA	0,0250	0,0186	52,6923	0,0217

Table 2: This table is providing the average values of the 10 seed for each stock of all the performance measures used. These values have been computed by using the LSTM model built using as reference the stock ASMLHoldingNV.

Table 2 shows that the magnitude of RMSE and MAE is relatively consistent across all the assets, suggesting that the LSTM specification initially developed by using a single stock can be applied to the remaining stocks without producing systematically abnormal prediction errors. The choice of using one representative stock for the initial model-development phase was made to make the computation faster and to build a single model that can be used for all the stocks avoiding the need to perform of a model-selection procedure independently for every asset. However, this does not imply that the same model would necessarily have been selected if the procedure had been performed on a different stock, since each asset has different return dynamics. But the results provide evidence that the selected LSTM architecture is sufficiently robust to be applied across the considered set of stocks.

4.6. Robustness analysis of the reference-stock selection

The initial LSTM specification adopted in this study was developed through a trial-and-error procedure using ASML Holding NV as the reference stock that produced the results of table 2.

As previously presented, the choice of ASML was not motivated by any particular statistical characteristic of the company, but simply by its position in the alphabetically ordered dataset.

Since the model specification was made once for all the stocks in the data set, it was important to check if the use of a different reference asset would have led to a different model specification.

For this purpose, the model-selection procedure was repeated using Air Liquide SA as an alternative reference stock. The trial-and-error analysis resulted in a partially different model specification, the different elements are: 256 LSTM hidden units, no dropout, an initial learning rate of 1×10^{-5} and a mini-batch size of 128. The remaining elements of the models, including also the set of input features, the three-day prediction horizon, the data splitting procedure and the general network structure remained unchanged.

Once the alternative specification had been determined, the model was trained using ten different

random seeds (from 1 to 10) and applied to the same complete set of 45 stocks considered for the original ASML-based model, this ensured that the two specifications could be compared consistently. The results (table 3) indicate that the use of Air Liquide as the reference stock did not produce substantial changes in the overall magnitude of the prediction errors. Across the 45 assets, the original ASML-based (table 2) specification generated an average RMSE of 0.030114 and an average MAE of 0.02274, compared with 0.030119 and 0.022749 respectively for the Air-Liquide-based specification (table 3). The differences are extremely small, showing that the ability of the LSTM to approximate the magnitude of future returns remained essentially unchanged when the model-development procedure was performed on a different reference stock.

The important difference was for the directional measures. The ASML-based specification achieved an average Directional Accuracy of 51.64%, whereas the alternative specification achieved 50.88%. Similarly, the average correlation between predicted and realised returns decreased from 0.0387 under the ASML-based model to 0.0170 under the Air-Liquide-based model.

Although both specifications produced similar prediction errors, the model built by using ASML as reference showed a slightly stronger ability to capture the direction movement of future returns and the correlation between actual and future returns. Furthermore, under the original specification, 30 out of 45 stocks achieved a Directional Accuracy above 50% and 35 stocks showed a positive correlation, compared with 28 and 31 stocks respectively under the alternative specification. And when both conditions are considered simultaneously, 24 stocks satisfied both criteria under the ASML-based model, compared with 22 stocks under the Air-Liquide-based model.

Overall, the robustness analysis suggests that the results of the study are not materially dependent on the arbitrary selection of ASML Holding NV as the initial reference stock. Despite repeating the model-development had produced small differences in the selected hyperparameters, this because the optimal specification can depend on the return dynamics of the stock used during the model development, when the two resulting specifications were applied to the complete data set, their prediction errors can be considered identical and their overall performance remained broadly comparable. Moreover, the original ASML-based specification performed slightly better on average, particularly in terms of Directional Accuracy and correlation, and these findings provide additional support for retaining the original ASML-based LSTM specification as the common model applied to the entire set of stocks.

Assets	RMSE	MAE	Directional AccuracyPct	Correlation
ASMLHoldingNV	0,0406	0,0315	54,7902	0,0182
AdidasAG	0,0373	0,0270	47,3592	0,0050
AirLiquideSA	0,0187	0,0140	52,4825	0,0127
AirbusGroupSE	0,0285	0,0226	52,1678	0,0203

Allianz	0,0207	0,0157	54,7887	0,0485
AnheuserBuschInbevSANV	0,0285	0,0201	44,4755	0,0060
ArgenxSE	0,0357	0,0264	52,2727	0,0077
AxaSA	0,0224	0,0168	55,9790	0,0723
BASFSE	0,0293	0,0227	48,8732	-0,0356
BAYERAG	0,0414	0,0311	51,0211	0,0540
BNPParibasSA	0,0296	0,0222	51,1538	0,0546
BancoBilbaoVizcayaArgentaria	0,0304	0,0227	52,3776	0,0555
BancoSantander	0,0300	0,0228	47,8322	-0,0420
BayerischeMotorenWerkeAG	0,0315	0,0241	50,8803	0,0348
DanoneSA	0,0241	0,0175	50,1748	0,0709
DeutscheBoerseAG	0,0242	0,0193	50,1408	0,0329
DeutschePostAG	0,0264	0,0202	51,3028	-0,0172
DeutscheTelekomAG	0,0271	0,0199	49,3662	0,0241
Enel	0,0205	0,0150	54,8239	0,0279
EniSpA	0,0254	0,0197	52,3592	0,0357
EssilorLuxotticaSA	0,0328	0,0240	48,9860	-0,0351
HermesInternationalSCA	0,0324	0,0241	46,2238	0,0572
INGGroepNV	0,0266	0,0209	51,3636	0,0023
Iberdrola	0,0168	0,0131	54,3357	-0,0271
IndustriaDeDisenoTextilSA	0,0282	0,0209	52,9371	0,0912
InfineonTechnologiesAG	0,0468	0,0361	51,3732	-0,0710
IntesaSanpaoloSpA	0,0254	0,0196	50,6690	-0,0614
KeringSA	0,0419	0,0322	52,3427	0,0148
KoneCorporation	0,0228	0,0170	50,4270	-0,0490
KoninklijkeAholdDelhaizeNV	0,0226	0,0158	47,9720	0,0010
KoninklijkePhilipsNV	0,0298	0,0232	53,3916	0,0710
L Or alSA	0,0275	0,0210	48,8462	0,0014
LVMHMoetHennesseyLouisVuitto	0,0346	0,0257	48,1469	-0,0053
MercedesBenzGroupAG	0,0274	0,0218	48,0986	0,0010
MunchenerRuckAG	0,0242	0,0181	48,9437	-0,0623
PernodRicardSA	0,0318	0,0256	48,2168	0,0716
RheinmetallAG	0,0514	0,0397	49,4718	0,0831
SAPSE	0,0354	0,0258	46,3860	-0,0037
SafranSA	0,0310	0,0224	52,6573	0,0859
SanofiSA	0,0262	0,0194	49,6853	-0,0084
SchneiderElectricSE	0,0305	0,0237	57,4126	0,1218
SiemensAGClassN	0,0312	0,0239	51,7254	0,0019
StellantisNV	0,0574	0,0416	49,0845	-0,0356
TotalEnergiesSE	0,0230	0,0180	54,6503	0,0590
VinciSA	0,0250	0,0187	51,6084	-0,0232

Table 3: This table is providing the average values of the 10 seed for each stock of all the performance measures used. These values have been computed by using the LSTM model built using as reference the stock Air Liquide SA.

CHAPTER 5 – Portfolio Optimization and Portfolio Back-Test frameworks and results

5.1. Portfolio Optimization frameworks

After the stock-return forecasts had been generated through the LSTM model, a mean–variance portfolio optimisation procedure was implemented to assess whether the expected returns produced could improve the portfolio construction by comparing this result to a traditional historical approach. Four investment strategies have been considered: the complete LSTM portfolio, the filtered LSTM portfolio, the historical Markowitz portfolio, and the equally weighted portfolio. All of them have been compared with the market benchmark, that is the EURO STOXX 50.

Like common rules for all the portfolios built, I implemented:

- No-short selling constraints. Meaning that the weights of the stocks inside the portfolios are always positive (Martínez-Barbero, Cervelló-Royo, & Ribal, 2025).
- Maximum weight for a single stock equal to 0.25. This had been implemented to reduce the concentration into one single stock.

Before the optimization was performed, the historical daily returns of the 45 stocks were aligned to the common set of trading dates, and the EURO STOXX 50 returns were subsequently matched with the stock-return dataset. In this way the benchmark and the portfolios were evaluated using the same observation period, so this alignment prevented differences in trading dates or missing observations from affecting the comparison.

Portfolio risk was estimated through the covariance matrix computed on the aligned historical daily stock returns, equal to $cov(R)$ where R represents the matrix containing the historical daily returns of the stocks. The same complete covariance matrix was used for both the complete LSTM portfolio and the historical Markowitz portfolio, consequently, the main difference between the two optimisation approaches concerned the estimation of the expected returns rather than the estimation of the portfolio risk.

Three different sets of expected returns were considered to build the different portfolios.

For the complete LSTM portfolio, the expected return of each stock was represented by the average daily return predicted by the LSTM model. The daily return was generated 10 times, once for different random seeds, these 10 values were aggregated to obtain one expected-return estimate per stock.

For the filtered LSTM portfolio, the expected returns were also obtained from the LSTM forecasts, but the investment universe was restricted to the stocks for which the model has satisfied two conditions, that are eq. (46) and eq. (47).

$$\text{Directional Accuracy}_i > 50\% ; \quad (46)$$

$$\text{Correlation}_i > 0 . \quad (47)$$

Eq. (46) required the model to predict the sign of the direction of the future return correctly in more than half of the comparison between the predicted returns and the actual returns of the test set. Eq. (47) required the relationship between predicted and realised returns to be positive. From the 45 stocks included in the complete dataset, only 24 satisfied both conditions and were therefore included in the filtered portfolio. For this portfolio also the covariance matrix had been reduced considering only the 24 stocks extracted.

For the historical Markowitz portfolio, the expected returns were calculated as the arithmetic mean of the historical daily returns, computed by eq. (48).

$$\hat{\mu}_i = \frac{1}{T} \sum_{t=1}^T r_{i,t} \quad (48)$$

where $r_{i,t}$ represents the historical daily return of stock i at time t .

For the equally weighted portfolio was not performed any optimisation, because the same weight was assigned to each of the 45 stocks, by eq. (49).

$$w_i = \frac{1}{45} \approx 2.22\% . \quad (49)$$

And finally, the EURO STOXX 50 was included as an external market benchmark.

5.1.1. *Efficient Frontiers construction*

The efficient frontiers of the complete LSTM, filtered LSTM, and historical Markowitz portfolios were constructed by solving a sequence of quadratic programming problems. The efficient frontier was built by following the procedure already explained in chapter 1. The procedure for the optimization was the same for all the three portfolios, the only difference was the kind of returns used to solve the quadratic programming problem.

For each target return, portfolio variance was minimised according to eq. (53), that is subject to different constraints, given by eq. (50), eq. (51) and eq. (52).

$$\mu'w \geq \mu^* ; \quad (50)$$

$$1'w = 1 ; \quad (51)$$

$$0 \leq w_i \leq 0.25 ; \quad (52)$$

$$\min_w w' \Sigma w . \quad (53)$$

In these expressions, w is the vector of portfolio weights, Σ is the covariance matrix, μ is the relevant expected-return vector, and μ^* is the selected target return.

The objective function (eq. 53) is to minimise the portfolio variance for each target return. The first constraint (eq. 50) requires the portfolio to achieve at least the selected target return, while the second (eq. 51) ensures that the portfolio is fully invested, meaning that the sum of the weights is equal to 1. Short selling was not permitted, since each weight was required to be non-negative (eq. 52), and

moreover, a maximum weight of 25% was imposed on each stock to limit excessive concentration. This limit was added because the predicted returns used in the LSTM were uncertain, so at least four different securities were required to construct a fully invested portfolio.

For each investment universe, the lower endpoint of the efficient frontier was identified through the global minimum-variance portfolio, and it was obtained by minimising the portfolio variance without imposing a target-return constraint. On the other side, the upper endpoint was determined by identifying the maximum feasible expected return under the same investment constraints seen before. Then one hundred equally spaced target returns were generated between the expected return of the global minimum-variance portfolio and the maximum feasible return, to build the points of the Efficient Frontier. For each target return, the quadratic optimisation problem was solved using the MATLAB's "*quadprog*" function, in this way the set of minimum-variance portfolios obtained formed the Efficient Frontier. This procedure had been made for all the different kind of portfolios computed, and the Efficient Frontiers and the corresponding optimal portfolios and benchmark are shown in fig. 5.

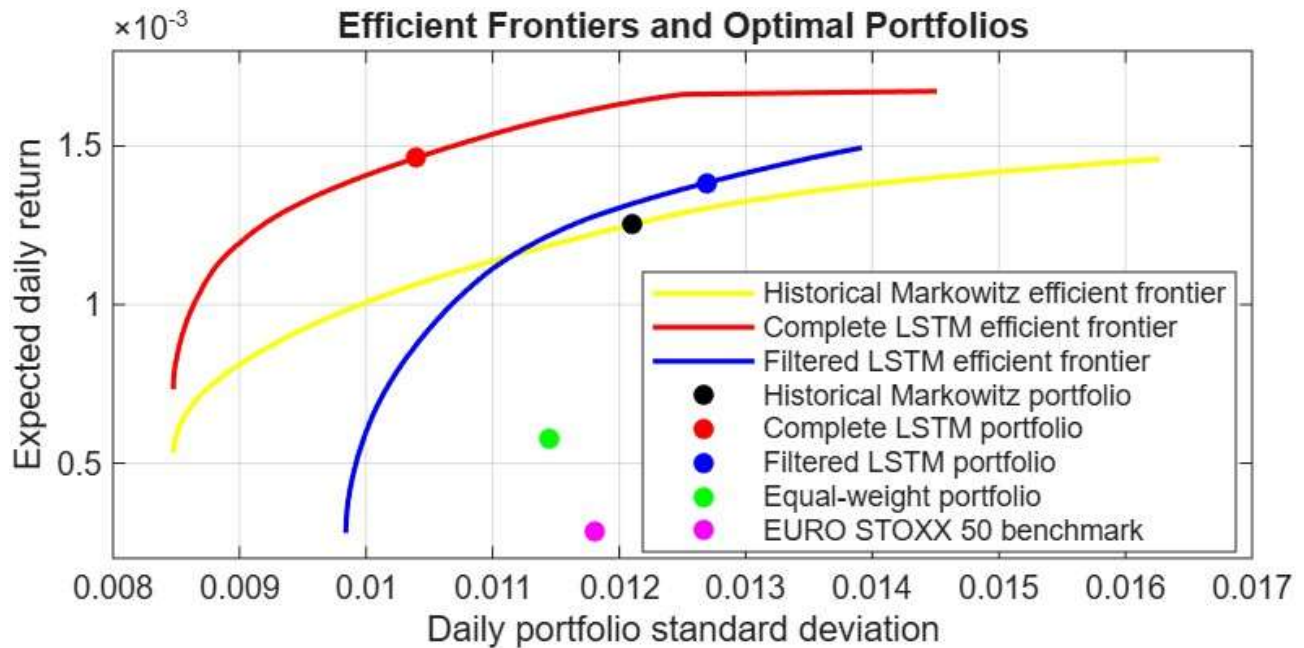


Figure 5: Different Efficient Frontiers for all the different kind of portfolios, with the corresponding optimal portfolios (those one that are maximizing the Sharpe ratio); compared with the benchmark.

5.1.2. Selection of the optimal portfolios

The final portfolio associated with each efficient frontier was selected by maximising the Sharpe ratio (fig. 5). The daily Sharpe ratio was calculated by using eq. (6).

An annual Euribor rate of 2.198% (Montemurro, 2026) was used as the risk-free rate and was converted into a daily equivalent return, with the following procedure of eq. (54).

$$r_{f,daily} = (1 + 0.02198)^{1/252} - 1 . \quad (54)$$

The daily measures were annualized (or the opposite) by using 252, which approximately represents the number of trading days in one year.

The portfolio with the highest Sharpe ratio on each efficient frontier was identified with the procedure explained in chapter one for the optimal or tangent portfolio (Sen, Mondal, & Mehtab, 2021).

The expected returns, standard deviations, and Sharpe ratios were annualized according to eq. (55), eq. (56) and eq. (57) respectively.

$$\mu_{p,annual} = 252 \times \mu_{p,daily} ; \quad (55)$$

$$\sigma_{p,annual} = \sqrt{252} \times \sigma_{p,daily} ; \quad (56)$$

$$SR_{p,annual} = \sqrt{252} \times SR_{p,daily} . \quad (57)$$

Expected returns (eq. 55) were multiplied by 252 because returns are assumed to accumulate linearly over time. Volatility (eq. 56) was multiplied by $\sqrt{252}$, since variance increases proportionally with time and standard deviation is the square root of variance. Consequently, the Sharpe ratio (eq. 57) was also multiplied by $\sqrt{252}$, because the annualized excess return increases by a factor of 252, while annualized volatility increases by a factor of $\sqrt{252}$.

These transformations assume that daily returns are identically distributed over time.

5.1.3. Portfolio optimization results

The results obtained from the portfolio optimization are summarised in table 4.

Portfolio	Daily Return	Annual Return	DailyS TD	Annual STD	Annual Sharpe Ratio
Complete LSTM portfolio	0,00146301	0,368678	0,0103961	0,165033	2,1022
Filtered LSTM portfolio	0,00138379	0,348715	0,0126878	0,201413	1,6234
Historical Markowitz portfolio	0,00125281	0,315709	0,0121048	0,192158	1,5298
Equal-weight portfolio	0,00057697	0,145397	0,0114457	0,181695	0,6806
EURO STOXX50 benchmark	0,00028446	0,071684	0,0118078	0,187443	0,2664

Table 4: It provides the results of all the different kind of portfolios built and the results of the benchmark.

The *complete LSTM portfolio* produced the most favourable ex-ante risk–return combination among the different strategies considered (fig. 5). It achieved an expected daily return of 0.1463%, corresponding to an annualized expected return of 36.87%. For the risk level, it reported a daily standard deviation of 1.0396%, equivalent to an annualized volatility of 16.50%. and the annualized Sharpe ratio was equal to 2.1022, the highest value among all the portfolios examined (table 4).

By comparing the complete LSTM portfolio with the historical Markowitz portfolio, the first one produced an expected annual return approximately 5.30 percentage points higher than the other one

(36.87% – 31.57%). And, at the same time, its annualized volatility was approximately 2.71 percentage points lower (16.50% – 19.22%), meaning that it had a higher expected returns but with a lower exposure to the risk.

The Sharpe ratio increased from 1.5298 for the historical Markowitz portfolio to 2.1022 for the complete LSTM portfolio. Therefore, the improvement associated with the LSTM forecasts did not result exclusively from a higher expected return, but for the most from the lower estimated level of risk.

The complete LSTM portfolio allocated positive weights to ten stocks, this result is shown in table 5. The largest allocations were assigned to Munich Re with approximately 16.91%, Pernod Ricard with 16.17%, Rheinmetall with 15.38%, Hermès with 11.27%, Vinci with 11.24%, and the remaining allocation was distributed across SAP, Argenx, Deutsche Börse, Ahold Delhaize, and Danone. None of the stocks included in the complete LSTM portfolio reached the maximum permitted weight of 25%, this indicated that the optimiser was able to construct a comparatively diversified allocation without relying excessively on a single stock.

The *filtered LSTM portfolio* achieved an expected daily return of 0.1384%, corresponding to an annualized expected return of 34.87%. Its daily standard deviation was 1.2688%, equivalent to an annualized volatility of 20.14%, while its annualized Sharpe ratio was 1.6234 (table 4). Like the complete one, the filtered portfolio performed better than the historical Markowitz portfolio in terms of both expected return and risk-adjusted performance. In fact, its annualized expected return was approximately 3.30 percentage points higher than that of the historical portfolio (34.87% – 31.57%). But differently from the complete portfolio, the annualized volatility was approximately 0.92 percentage points higher (20.14% – 19.22%), meaning that it had a higher expected returns with a slightly higher exposure to the risk than the historical portfolio. Despite the slightly higher risk, its Sharpe ratio was higher than the one of the historical portfolio, increasing from 1.5298 to 1.6234.

Although 24 stocks satisfied the filtering criteria (table 5), the optimal filtered portfolio allocated economically relevant weights to only five assets, where Munich Re and Vinci both reached the maximum weight of 25%, Safran received approximately 19.07%, Argenx 18.74%, and Iberdrola 12.19%. The resulting portfolio was therefore substantially more concentrated than the complete LSTM portfolio. The fact that two of the five stocks had reached the upper-bound constraint means that the optimiser would have assigned even larger weights to these securities in the absence of the 25% limit. More important, the application of the forecasting-performance filters did not improve the results relative to the complete LSTM portfolio. In fact, the complete portfolio achieved an expected annual return approximately 2.00 percentage points higher (36.87% – 34.87%), while presenting an annualized volatility approximately 3.64 percentage points lower (16.50% – 20.14%). Furthermore,

the annualized Sharpe ratio declined from 2.1022 of the complete LSTM portfolio to 1.6234 of the filtered one.

This result suggested that by selecting the securities exclusively according to individual forecasting metrics this did not necessarily improve portfolio-level efficiency. This happens probably because the Directional Accuracy and the Correlation evaluated the predictive performance of the LSTM model for each stock separately, and these metrics did not directly account for the covariance relationships among the securities. As a consequence, from the complete portfolio had been removed stocks with relatively weak individual forecasting metrics but with a positive contribution to the portfolio volatility by providing diversification benefits, so their exclusion could increase the total portfolio risk, even when the model had demonstrated limited predictive performance for that stock. The filtering procedure restricted the combinations available to the optimiser, by reducing the investment universe from 45 to 24 securities and produced a more concentrated and more volatile allocation.

The *historical Markowitz portfolio* achieved an expected daily return of 0.1253%, corresponding to an annualized expected return of 31.57%, and an annualized volatility of 19.22%. Its annualized Sharpe ratio was equal to 1.5298 (table 4).

As shown in table 5 the portfolio had provided an allocation to all the stocks, but with very small weights for most of the stocks. It was mainly concentrated in Argenx, Rheinmetall, Iberdrola, Ahold Delhaize, ASML, Deutsche Börse and Hermes. Argenx and Rheinmetall received weights close to the maximum permitted allocation, that were approximately 24.59% and 23.68% respectively, while Iberdrola received approximately 20.96%.

Despite the historical portfolio exhibited a weight for each stock, it has a greater degree of concentration than the complete LSTM portfolio. This is because the economically relevant weights inside the Markowitz portfolio were less than those of the complete LSTM portfolio (7 against 10 economically relevant allocations, respectively). Nevertheless, its allocation was less concentrated than the one of the filtered LSTM portfolio, since positive weights were distributed across a larger number of securities (7 against 5 economically relevant allocations respectively).

The *equally weighted portfolio* achieved: an expected daily return of 0.058% that annualized was 14.54%, an annualized volatility of 18.17%, and an annualized Sharpe ratio of 0.6806 (table 4).

Its expected return and Sharpe ratio were substantially lower than those of the three optimised portfolios. However, its volatility was lower than the one of both filtered LSTM and historical Markowitz portfolios, this result had illustrated the diversification benefit obtained by distributing capital uniformly across all 45 securities (each stock had a weight of 0.022 as shown in table 5). Nevertheless, equal weighting did not show benefits in expected returns, volatilities, or correlations

against the complete LSTM portfolio, in fact it had produced a less favourable expected risk–return trade-off.

The *EURO STOXX 50 benchmark* achieved an average daily return of 0.0284%, corresponding to an annualized return of 7.17%, its annualized volatility was 18.74%, while the annualized Sharpe ratio was equal to 0.2664 (table 4). As can be seen from table 4 and fig. 5, all four internally constructed portfolios achieved significantly higher expected annual returns, not always lower volatility, and higher Sharpe ratios than the benchmark. The complete LSTM portfolio, compared with the benchmark, had the biggest differences: between the expected annual returns (approximately 29.70% higher) and between the annualized volatility (approximately 2.24% lower).

The equal-weight portfolio also outperformed the benchmark. Despite its reported expected return was not significantly higher than the one of the benchmark, due to the lower volatility, it had reached a greater Sharpe ratio.

The graphical comparison of the efficient frontiers of fig. 5 confirmed the numerical results shown in table 4.

In fact, the complete LSTM Efficient Frontier (the red curve) was positioned above the historical Markowitz (the yellow line) and the filtered LSTM (the blue line). The filtered frontier started under the historical Markowitz frontier and ended over it.

The complete LSTM optimal portfolio (the red point) was positioned above and to the left of both the historical Markowitz (the black point) and filtered LSTM (the blue point) optimal portfolios, meaning that it provided a higher expected return with a lower estimated volatility.

The filtered LSTM frontier was, for lower risks, located below the historical frontier, meaning that for the same level of risk was providing a lower expected return. And for higher risk, it crossed the historical frontier and became superior and it started providing for the same level of risk higher expected return. This reflected the more restricted investment universe of the filtered portfolio, in fact the filtering procedure limited the possibility of constructing highly diversified low-risk portfolios.

The historical Markowitz portfolio offered a lower expected return than both kinds of LSTM optimal portfolios. The equally weighted portfolio (the green point) and the EURO STOXX 50 benchmark (the purple point) were positioned substantially below the efficient frontiers, indicating a less favourable expected return for their respective levels of risk.

Asset	Directional Accuracy	Correlation	Selected By Filter	LSTM Portfolio	Filtered LSTM	Historical Markowitz	Equal Weight
ASMLHoldingNV	54,4056	-0,023163	FALSE	0	0	0,10316	0,022222
AdidasAG	48,0282	-0,026164	FALSE	0	0	0,000004	0,022222
AirLiquideSA	54,2657	0,028991	TRUE	0	0	0,000002	0,022222
AirbusGroupSE	52,9371	0,003803	TRUE	0	0	0,000002	0,022222
Allianz	54,7887	0,071085	TRUE	0	0	0,000004	0,022222

AnheuserBuschInbevS ANV	48,9161	0,062357	FALSE	0	0	0,000001	0,022222
ArgenxSE	52,5874	0,018153	TRUE	0,063364	0,187373	0,245931	0,022222
AxaSA	57,9371	0,142927	TRUE	0	0	0,000003	0,022222
BASFSE	48,662	0,0016	FALSE	0	0	0,000001	0,022222
BAYERAG	54,0141	0,089751	TRUE	0	0	0,000001	0,022222
BNPParibasSA	54,3007	0,105074	TRUE	0	0	0,000004	0,022222
BancoBilbaoVizcayaAr gentaria	54,4755	0,101529	TRUE	0	0	0,000006	0,022222
BancoSantander	50,1399	0,01653	TRUE	0	0	0,000002	0,022222
BayerischeMotorenWer keAG	50,5282	0,028269	TRUE	0	0	0,000001	0,022222
DanoneSA	49,1958	0,0812	FALSE	0,038255	0	0,000002	0,022222
DeutscheBoerseAG	50,8803	-0,00212	FALSE	0,054003	0	0,071128	0,022222
DeutschePostAG	50,9155	-0,001934	FALSE	0	0	0,000003	0,022222
DeutscheTelekomAG	51,2324	0,041117	TRUE	0	0	0,000005	0,022222
Enel	55,2113	0,142071	TRUE	0	0	0,000004	0,022222
EniSpA	49,0845	0,106577	FALSE	0	0	0,000004	0,022222
EssilorLuxotticaSA	48,8112	-0,042383	FALSE	0	0	0,000001	0,022222
HermesInternationalSC A	46,2937	0,048467	FALSE	0,11267	0	0,017924	0,022222
INGGroepNV	54,4755	0,024262	TRUE	0	0	0,000005	0,022222
Iberdrola	55,2098	0,005641	TRUE	0	0,121886	0,209556	0,022222
IndustriaDeDisenoTexti ISA	54,0559	0,098675	TRUE	0	0	0,000002	0,022222
InfineonTechnologiesA G	54,4718	0,041168	TRUE	0	0	0,000008	0,022222
IntesaSanpaoloSpA	53,2042	-0,06068	FALSE	0	0	0,000028	0,022222
KeringSA	53,4266	0,053203	TRUE	0	0	0,000001	0,022222
KoneCorporation	49,3594	-0,018544	FALSE	0	0	0,000003	0,022222
KoninklijkeAholdDelha izeNV	48,042	-0,021332	FALSE	0,045688	0	0,115323	0,022222
KoninklijkePhilipsNV	53,4965	0,059681	TRUE	0	0	0,000001	0,022222
L_Or_alSA	50,8042	0,014745	TRUE	0	0	0,000004	0,022222
LVMHMoetHennesseyL ouisVuitto	48,7063	0,023272	FALSE	0	0	0,000003	0,022222
MercedesBenzGroupA G	48,5563	0,034485	FALSE	0	0	0,000001	0,022222
MunchenerRuckAG	51,0915	0,001068	TRUE	0,169147	0,25	0,000006	0,022222
PernodRicardSA	47,7972	0,072445	FALSE	0,161699	0	0,000001	0,022222
RheinmetallAG	48,9437	0,093557	FALSE	0,153847	0	0,236836	0,022222
SAPSE	47,0877	0,026352	FALSE	0,088974	0	0,000007	0,022222
SafranSA	52,5175	0,081026	TRUE	0	0,190741	0,000002	0,022222
SanofiSA	50,5944	-0,015691	FALSE	0	0	0,000002	0,022222
SchneiderElectricSE	57,1678	0,11621	TRUE	0	0	0,000005	0,022222
SiemensAGClassN	53,838	-0,005899	FALSE	0	0	0,000002	0,022222
StellantisNV	49,8592	0,022015	FALSE	0	0	0,000001	0,022222
TotalEnergiesSE	51,014	0,079626	TRUE	0	0	0,000007	0,022222
VinciSA	52,6923	0,021683	TRUE	0,112354	0,25	0,000003	0,022222

Table 5: For each single stock are presented: the Direction Accuracy, the Correlation coefficient, if the stock had been selected for the filtered LSTM portfolio (TRUE) or not (FALSE), the corresponding weights of each stock for the complete LSTM portfolio, the corresponding weights of each stock for the filtered LSTM portfolio, the corresponding weights of each stock for the historical Markowitz portfolio and the corresponding equal weights of all the stocks for the Equally Weighted portfolio .

5.1.4. Overall results and conclusion on the Portfolio Optimization

Overall, the portfolio optimisation results (table 4 and fig. 5) show that by replacing historical expected returns with the forecast returns produced by the LSTM model the Efficient Frontier generally shifts upward and, consequently, there is an improvement of the estimated risk–return trade-off.

Among the four construction strategies for the portfolio considered, the complete LSTM portfolio was the best, achieving: the highest annualized expected return, the lowest annualized volatility, the highest annualized Sharpe ratio, and a more diversified allocation than the filtered LSTM and the historical Markowitz portfolios.

The filtered LSTM portfolio also outperformed the historical Markowitz portfolio and the benchmark in terms of expected return and Sharpe ratio. But by filtering the stocks according to the Directional Accuracy and the Correlation coefficient there were a reduction of the diversification inside the final portfolio and a consequently higher portfolio volatility, reducing the annualized Sharpe ratio. So, the result is telling that the hypothesis of removing stocks with weaker individual forecasting metrics would necessarily improve the final portfolio is not correct, because by filtering the number of stocks available for the optimization procedure the diversification benefits were reduced.

The use of the portfolio optimization was only one of the methods used to understand if using the optimization input provided by the LSTM model was better than using the tradition historical data. These results, in fact, represents an ex-ante comparison based on estimated and predicted returns. In particular, the expected returns of the LSTM portfolios were model-generated forecasts rather than realised investment returns. Also, the mean–variance optimisation model is highly sensitive to expected-return estimation errors, meaning that relatively small variations in the forecasts can produce substantial changes in the optimal portfolio weights.

Consequently, the superior position (fig. 5) of the complete LSTM efficient frontier could not, by itself, demonstrate that the strategy would have generated superior realised performance.

To improve the analysis, it has been used the out-of-sample back-test procedure (that will be examined in the next section). In this approach the portfolio performance was measured by comparing predicted returns with the same time window realised returns, after accounting for periodic rebalancing and transaction costs.

5.2. Out-of-sample portfolio Back-Test

The out-of-sample back-test was conducted to evaluate whether the portfolios constructed by using the LSTM forecasts generated superior realised performance.

Unlike the efficient-frontier analysis, which compares portfolios using estimated expected returns and covariance matrices, the back-test measures the returns that would actually have been obtained over the subsequent investment periods. This means that at each rebalancing date, the portfolio weights are recalculated using only the information available at that time. For the complete and filtered LSTM portfolios, the expected return vector was obtained from the LSTM forecasts generated for the following investment period, while portfolio risk was estimated through the covariance matrix calculated from the most recent 252 historical daily returns (the number of trading days in one year). For the historical Markowitz portfolio, both the expected returns and the covariance matrix were estimated using the same rolling historical window. The resulting weights were then applied to the actual stock returns subsequently observed between the forecast date and the corresponding target date. In this way, the back-test evaluated the realised performance of the portfolios by comparing the allocations selected *ex ante* (following the expectation provided by the historical and predicted returns) with the returns that occurred *ex post* (the real returns observed after the time window had ended).

Five strategies were compared: the complete LSTM portfolio, the filtered LSTM portfolio, the historical Markowitz portfolio, the equally weighted portfolio, and the EURO STOXX 50 benchmark.

5.2.1. *Back-Test framework*

The LSTM model had been developed using a prediction horizon of three trading days. Therefore, each portfolio constructed at the forecast date t was held until the corresponding target date T , located three trading days ahead. To prevent the same daily return from being counted in more than one investment period, only non-overlapping forecast–target date pairs were retained.

The back-test had covered the period from 10 April 2025, the first forecast date, to 28 May 2026, the final target date, meaning that a total of 88 investment periods were included.

Consequently, the portfolio was rebalanced approximately every three trading days (the window could be larger if there were inside Saturday, Sunday or holiday). This approach ensured that each realised return represented a separate investment decision based exclusively on the predictions made at the beginning of the period.

At every forecast date (t), the portfolio optimisation was repeated using only the information available at that point in time. A rolling estimation window of 252 trading days was used to estimate historical expected returns and the covariance matrix. The choice of 252 observations approximately corresponded to one trading year and provided a balance between the use of sufficiently recent information and the need for a sufficiently large sample to estimate the covariance matrix.

For each forecast date t , the historical expected return of stock i was calculated by using eq. (58), that is:

$$\hat{\mu}_{i,t}^H = \frac{1}{252} \sum_{s=t-251}^t r_{i,s} . \quad (58)$$

Where $r_{i,s}$ represents the daily return of stock i on trading day s , and $\hat{\mu}_{i,t}^H$ is the rolling average return for the optimization of the historical Markowitz portfolio.

The rolling covariance matrix was similarly estimated using the daily returns observed over the same 252-day window ($\Sigma_t = \text{Cov}(R_{t-251:t})$).

A small regularisation term was added to the diagonal of the covariance matrix (eq. 59).

$$\tilde{\Sigma}_t = \Sigma_t + 10^{-8}I \quad (59)$$

where Σ_t is the original covariance matrix, 10^{-8} is the regularisation term, I is the diagonal matrix, and $\tilde{\Sigma}_t$ is the regularised matrix. This adjustment did not materially change the estimated risks, but it was used to improve the numerical stability of the quadratic optimisation procedure.

For the complete LSTM portfolio, the expected-return vector was obtained from the forecasts generated by the LSTM model for that specific forecast and target dates, and the entire stocks sample is used for the optimization.

The filtered LSTM portfolio used the same forecasted returns of the complete LSTM portfolio but with a restriction of the investment universe. It was filtered only to the 24 stocks (table 5), by using the same conditions set for the portfolio optimization.

5.2.2. *Back-Test functioning*

At each rebalancing date, the complete LSTM, filtered LSTM, and historical portfolios were reconstructed using the same mean–variance procedure adopted in the portfolio optimization phase with the same optimization constraints (portfolio fully invested, short selling not permitted and maximum allocation per stock limited to 25%).

For each target return, portfolio variance was minimised according to eq. (53), that is subjected to different constraints, given by eq. (50), eq. (51) and eq. (52). This procedure was repeated for each rebalancing date, but changing the predicted return used for both LSTM portfolios, the rolling expected return of the historical Markowitz portfolio, and by moving the rolling covariance matrix one step further.

This dynamic procedure differed from the previous Efficient Frontier analysis because portfolio weights were not held constant over the complete evaluation period. Instead, expected returns, the covariance matrix, and portfolio weights were recalculated at every forecast date.

After the weights had been determined, the realised return of each portfolio was calculated using the actual stock returns observed between the forecast date and the corresponding target date.

For portfolio p , the gross realised return over the three-day investment horizon was given by eq. (60), that is:

$$R_{p,t,T}^{\text{gross}} = w'_{p,t} \times R_{t,T}. \quad (60)$$

Where $R_{t,T}$ is the vector of realised simple stock returns between forecast date t and target date T , and $w'_{p,t}$ is the vector of the weights provided after the optimization for that time window. For all the four internally constructed portfolios was used the same realised stock-return vector to compute the gross return. Consequently, differences in performance were exclusively attributable to differences in portfolio weights given after the optimization.

The realised EURO STOXX 50 return was calculated directly from the index prices observed on the forecast and target dates, and the computation was made by using eq. (61). That is:

$$R_{t,T}^B = \frac{P_T^B}{P_t^B} - 1 \quad (61)$$

where P_T^B is the price at the target date and P_t^B is the price at the forecast date.

The benchmark was therefore evaluated over exactly the same investment periods as the portfolios to make the comparison of the results more coherent.

To have a correct analysis of the performance of the portfolios the transaction costs had been included, because the weights were adjusted every three trading days and the broker usually ask for a payment of a fee for this. The transaction costs were included for the complete LSTM, filtered LSTM, and historical portfolios, the ones that changed periodically the weights (Yashaswi, 2021).

A proportional transaction cost of four basis points was applied (AG), the cost ratio was given by eq. (62), that is:

$$c = \frac{4}{10,000} = 0.0004. \quad (62)$$

Where c is the average cost ratio paid to rebalance the portfolio.

The total cost for the portfolio changing weights was computed by using the portfolio turnover at rebalancing date t . The turnover measures how much the portfolio composition changes when it is rebalanced. The total cost was approximated with eq. (63), it is:

$$TO_{p,t} = \frac{1}{2} \sum_{i=1}^N |w_{i,t} - w_{i,t-1}|. \quad (63)$$

Where $w_{i,t}$ is the new weight assigned to asset i at date t , $w_{i,t-1}$ is the asset's weight immediately before rebalancing, and N is the number of assets of the sample from which the portfolio can be built. The factor of one half prevented simultaneous purchases and sales from being counted twice, for example, by moving 10% of the portfolio from one stock to another produces a total absolute weight change of 20%, but the actual portfolio turnover is 10%. For the first portfolio construction, turnover was set equal to one, corresponding to the initial investment of the entire available capital. A high turnover means that portfolio weights change substantially and that the trading activity is frequent,

leading to higher transaction costs. The transaction cost incurred in each period can be computed by using eq. (64), that is:

$$TC_{p,t} = TO_{p,t} \times 0.0004. \quad (64)$$

Where $TO_{p,t}$ is the turnover of the portfolio at date t , and $TC_{p,t}$ is the total cost to rebalance the portfolio at date t .

5.2.3. Portfolios back-test performance measures

For the performance of the portfolios had been used different measures.

The *net realised performance* of the portfolio was subsequently calculated by using eq. (65). It is:

$$R_{p,t,T}^{\text{net}} = R_{p,t,T}^{\text{gross}} - TC_{p,t} \quad (65)$$

where $R_{p,t,T}^{\text{gross}}$ is the gross performance of the portfolio inside the time window from t to T , and $TC_{p,t}$ is the total cost.

For all the strategies had been computed the *cumulative final wealth*. To all the strategies were assigned an initial wealth equal to one, and the cumulative wealth, after K investment periods, was calculated with eq. (66), that is:

$$W_{p,K} = \prod_{k=1}^K (1 + R_{p,k}^{\text{net}}). \quad (66)$$

The wealth obtained after K investment periods was calculated as the product of the net portfolio growth factors $(1 + R_{p,k}^{\text{net}})$ over all the periods. The product operator accounts for the compounding effect of returns, since the performance achieved in each period is applied to the wealth accumulated in the preceding periods. So, a final wealth greater than one indicates a positive cumulative return, on the other side a value below one indicates an overall loss. For example, a final wealth equal to 1.70 indicated that an initial investment of one unit had increased by approximately 70%.

The *annualized return* was calculated from the conversion of the complete sequence of realised three-day returns into an equivalent annual return. It was computed by using eq. (67), that is:

$$R_{p,\text{annual}} = [W_{p,K}]^{\frac{252/3}{K}} - 1 \quad (67)$$

where $W_{p,K}$ is the realised cumulative final wealth of portfolio p during all the three-day investment periods, K is the total number of three-day periods included in the back-test, 252 is the conventional number of trading days in one year, $252/3 = 84$ is the approximate number of three-day investment periods in one year, $\frac{252/3}{K}$ converts the cumulative growth obtained over K three-day periods into the equivalent growth over 84 three-day periods, corresponding to approximately one trading year, and finally subtracting one transforms the annualized growth factor back into an annualized return. When the back-test contains exactly $K = 84$ three-day periods, the exponent is equal to one, and the

cumulative return already corresponds to one year. On the other hand, when $K > 84$, the cumulative performance refers to more than one year and is scaled down to an annual rate, or when $K < 84$, it is projected to its one-year equivalent.

The *annualized volatility* was calculated by eq. (68), that is:

$$\sigma_{p,\text{annual}} = \sigma_{p,h} \sqrt{\frac{252}{3}} \quad (68)$$

where $\sigma_{p,h}$ was the standard deviation of the realised three-day portfolio returns.

The *annualized Sharpe ratio* was calculated with eq. (69). It is:

$$SR_{p,\text{annual}} = \frac{\bar{R}_{p,h} - r_{f,h}}{\sigma_{p,h}} \sqrt{\frac{252}{3}} \quad (69)$$

where the three-day risk-free rate was equal to $r_{f,h} = (1 + 0.02198)^{3/252} - 1$, $\bar{R}_{p,h}$ is the average of the realised three-day portfolio returns, and $\sigma_{p,h}$ is the three-day standard deviation.

The *maximum drawdown* was calculated as the largest percentage decline in cumulative wealth from a previous peak. It compared the cumulative wealth observed in each period with the highest wealth previously reached by the portfolio. The minimum value of this series identified the most severe peak-to-trough loss experienced during the back-testing period. Since drawdowns were expressed as negative values, a more negative maximum drawdown indicated greater downside risk, it was computed by using eq. (70). It is:

$$MDD_p = \min_k \left(\frac{W_{p,k}}{\max_{j \leq k} W_{p,j}} - 1 \right). \quad (70)$$

Where $W_{p,k}$ is the cumulative wealth of portfolio p at period k , $\max_{j \leq k} W_{p,j}$ is the highest cumulative wealth reached by the portfolio up to period k , the ratio compares current wealth with the previous peak and subtracting one expresses the decline as a percentage. When the current wealth is equal to its historical maximum, the drawdown is zero, when the current wealth is below the previous maximum, the drawdown is negative. For example, if the portfolio reached a peak wealth of 1.20, but subsequently declined to 1.02, the corresponding drawdown would be $-0.15 \left(\frac{1.02}{1.20} - 1 \right)$. This means that the portfolio had experienced a 15% decline from its previous peak, and eq. (70) extracts the most negative drawdown observed during the entire back-test.

5.2.4. Back-Test performance results

The results of the back-test had been compared between the performance of the portfolios considering the transaction costs, and the performance of the portfolios without transaction costs. This comparison had been made to see how much the transaction costs impacted on the final performance.

The results obtained without considering the transaction costs are shown in table 6 and fig. 6.

Portfolios	Annualized net-return	Annualized net-volatility	Annualized Sharpe	Maximum Drawdown	Final Wealth	Portfolio Turnover	Period cost	Period return
Complete LSTM	0,241295	0,15590	1,325388	- 0,157312	1,254137	0,391891	0	0,002718
Filtered LSTM	0,690396	0,160268	3,228959	- 0,115599	1,733186	0,398409	0	0,006419
Historical Markowitz	0,190942	0,146016	1,121143	- 0,087496	1,200893	0,111458	0	0,002207
Equal-weight	0,281751	0,136137	1,733931	- 0,09948	1,296991	0	0	0,003069
EURO STOXX 50	0,218071	0,146577	1,271814	- 0,106427	1,229567	0	0	0,002478

Table 6: Results for all the portfolios annualized with the assumption that in this market there are no transaction costs.

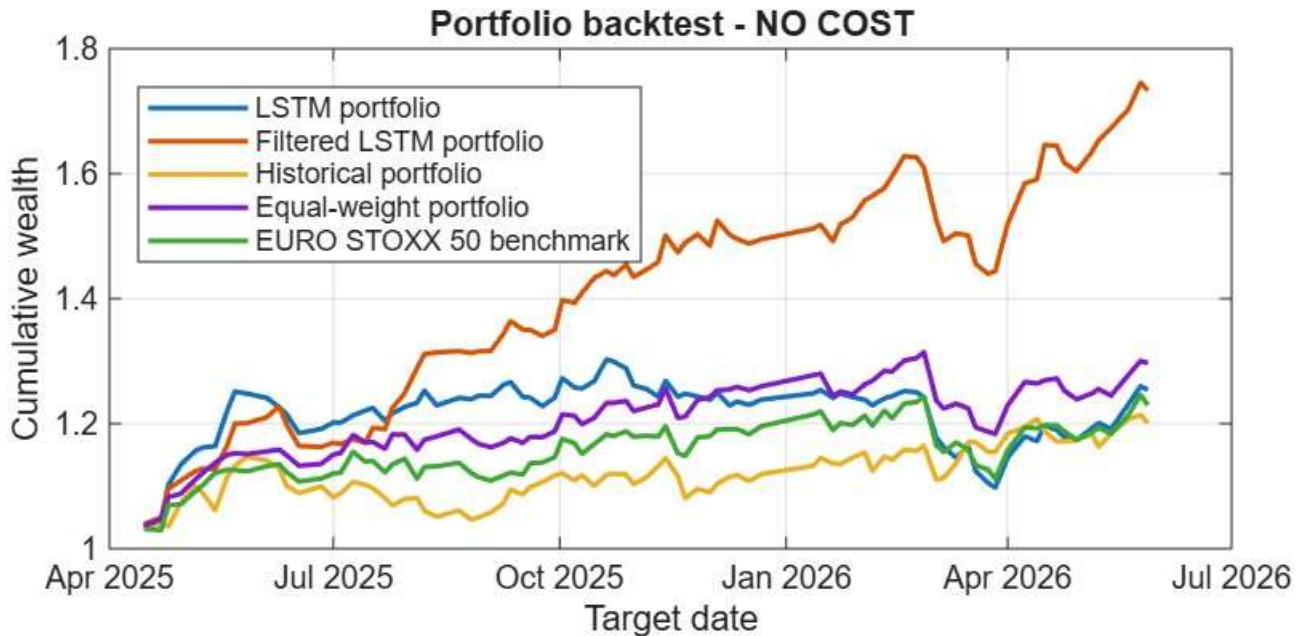


Figure 6: Cumulative wealth generated by the complete LSTM, filtered LSTM, historical Markowitz, equally weighted, and EURO STOXX 50 portfolios during the out-of-sample back-test. Portfolio weights were recalculated every three trading days. These measurements had been made with the assumption that in this market there are no transaction costs.

In table 6 the period cost is for all the portfolios equal to 0, because there was the assumption that the reallocation of the weights at each date t was free of transaction costs. On the other hand, the turnover column is showing a value different from 0, only for both LSTM portfolios and for the historical Markowitz portfolio. This because they are the only portfolios that were changing their weights at each period t , the equally weighted portfolio and the benchmark did not rebalance their weights periodically.

The annualized net returns and volatilities were computed by using the net period returns, that were computed by making the difference between the gross returns and the transaction costs (eq. (65)). For this reason, the results of table 6 and 7 were different, excluding the portfolio turnover, and because the results of table 7 were considering the impact of transaction costs they were lower than the one of table 6.

The results obtained taking in consideration the transaction costs are shown in table 7 and fig. 7.

Portfolios	Annualized net-return	Annualized net-volatility	Annualized Sharpe	Maximum Drawdown	Final Wealth	Portfolio Turnover	Period cost	Period return
Complete LSTM	0,225122	0,155758	1,242125	- 0,16151	1,23702	0,39189	0,000156	0,002562
Filtered LSTM	0,668032	0,160344	3,143945	- 0,11695	1,70917	0,39840	0,000159	0,006260
Historical Markowitz	0,186510	0,145947	1,096013	- 0,08819	1,19621	0,11145	0,000044	0,002163
Equal-weight	0,281751	0,136137	1,733931	- 0,09948	1,29699	0	0	0,003069
EURO STOXX 50	0,218071	0,146577	1,271814	- 0,106427	1,22956	0	0	0,002478

Table 7: Results for all the portfolios annualized with the assumption that in this market there are the transaction costs.

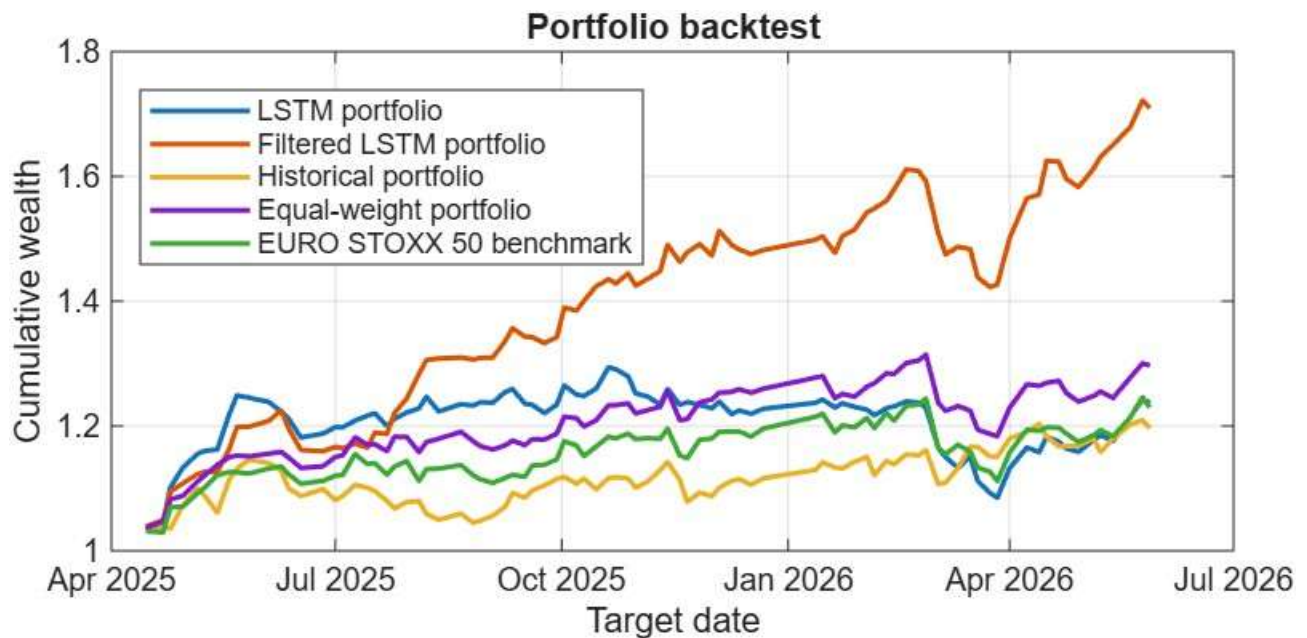


Figure 7: Cumulative wealth generated by the complete LSTM, filtered LSTM, historical Markowitz, equally weighted, and EURO STOXX 50 portfolios during the out-of-sample back-test. Portfolio weights were recalculated every three trading days. Proportional transaction costs of four basis points were applied to the complete LSTM, filtered LSTM, and historical portfolios.

Like can be seen from table 6 and 7 the turnovers are identical, this because the turnover was not influenced by the presence inside the market of the transaction costs. The complete and filtered LSTM portfolios exhibited considerably higher turnover than the historical Markowitz portfolio, their average turnover was approximately: 39.19%, 39.84% and 11.15% respectively.

This corresponded to an average transaction cost per rebalancing period (table 7) of approximately 1.57 basis points for the complete LSTM portfolio, 1.59 basis points for the filtered LSTM portfolio, and 0.45 basis points for the historical portfolio. The average transaction cost is directly related to the turnover of the weights inside the portfolio. In fact, the higher the average turnover was the higher the average transaction cost per period was, and both LSTM strategies reflected this. In fact, their average transaction cost was significantly higher than the one of the historical Markowitz portfolio. The first two portfolios were following the variability of the predicted expected returns and, consequently, they changed more frequent by their optimal weights, resulting in higher turnover and consequently in higher transaction costs.

In table 7 it is shown, like in table 6, that for both the equally weighted portfolio and the benchmark the turnover and the transaction costs are equal to 0, this because their weights were not reallocated periodically. In fact, their performances are the same in both tables.

The *filtered LSTM portfolio* generated the strongest realised performance among all the strategies for both market assumptions (fig. 6 and fig. 7).

Its final wealth, considering transaction costs, was equal to 1.7092 (table 7), corresponding to a cumulative return of approximately 70.92% over the complete back-testing period. This strategy achieved an annualized return of 66.80%, compared with 21.81% for the EURO STOXX 50 benchmark, being approximately 44.99 percentage points higher.

Despite its substantially higher return, it was really interesting to see that the filtered LSTM portfolio reported an annualized volatility of 16.03%, that was only 1.38 percentage points higher than the benchmark volatility of 14.66%. As a consequence, the filtered LSTM portfolio produced a significantly higher annualized Sharpe ratio, with a value of 3.1439 compared with 1.2718 for the benchmark.

The filtered portfolio also reported a maximum drawdown of -11.70%. Although this loss was slightly larger than the benchmark drawdown of -10.64%, due to the higher volatility, it remained substantially smaller than the maximum drawdown of the complete LSTM portfolio (-16,15%).

Fig. 7 showed that the filtered LSTM strategy began to diverge from the other portfolios during the second half of 2025, its wealth increased from approximately 1.20 in July 2025 to around 1.50 at the beginning of 2026. It subsequently reached a local peak of approximately 1.61 in February 2026, following an immediate decrease, probably due to the high volatility periods that the market faced for macroeconomic factors (war, political policy, custom duties), and this period affected all the

portfolios. This temporary decline experienced during March 2026, where the cumulative wealth fell to approximately 1.42, had been recovered rapidly by this strategy. The strategy concluded the back-test at its highest level, equal to approximately 1.71.

These results indicated that by restricting the investment universe to stocks for which the LSTM model had achieved both a Directional Accuracy above 50% and a positive Correlation coefficient considerably improved the realised portfolio performance during the analysed period, and this result was different than the one obtained in the portfolio optimization.

The *complete LSTM portfolio* achieved a final wealth of 1.2370, corresponding to a cumulative return of approximately 23.70%, with an annualized return of 22.51%, slightly higher than the benchmark return of 21.81% (table 7). This investment strategy reported an annualized volatility of 15.58%, compared with 14.66% for the benchmark, and a Sharpe ratio equal to 1.2421, slightly below the benchmark Sharpe ratio of 1.2718. This means that the slightly higher return achieved by the complete LSTM strategy was not sufficient to compensate for its higher level of risk relative to the benchmark of the market.

Also, the complete LSTM portfolio experienced the largest maximum drawdown among the strategies considered, equal to -16.15% . The largest decline happened during the same period of the filtered LSTM portfolio, but with larger negative effect bringing this strategy to the lowest cumulative wealth during that time window (fig. 7). Although the portfolio subsequently recovered and finished marginally above the benchmark, its performance did not demonstrate a clear risk-adjusted advantage.

The complete LSTM portfolio generated a higher final wealth than the EURO STOXX 50, but this result was achieved with higher volatility and a substantially larger drawdown.

Therefore, using all 45 stocks without considering the individual predictive quality of the model did not produce the strongest out-of-sample allocation.

The *historical Markowitz portfolio* achieved an annualized return of 18.65%, annualized volatility of 14.59%, a Sharpe ratio of 1.0960, and a final wealth equal to 1.1962, corresponding to a cumulative return of approximately 19.62%. This portfolio generated the lowest final wealth among the internally constructed strategies and underperformed both the EURO STOXX 50 and the two LSTM portfolios in terms of cumulative return. Despite this, it reported the smallest maximum drawdown, equal to -8.82% (table 7).

Its relatively limited downside risk reflected the greater stability of the rolling historical expected returns (lower standard deviation beaten only by that of the equally weighted portfolio) and the lower turnover of its portfolio weights.

The historical portfolio therefore provided a comparatively conservative allocation, but its lower risk was accompanied by lower realised performance.

Fig. 7 was showing a better adaptation of this strategy to the declining period experienced during the first half of 2026, recovering from the downturn faster than the benchmark and the complete LSTM portfolio. This strategy was partially convenient only when the market was facing a high volatility period, but there were better strategies like the filtered LSTM and the equally weighted portfolios.

The *equally weighted portfolio* produced a final wealth of 1.2970, representing a cumulative return of approximately 29.70%, with an annualized return of 28.18%, an annualized volatility of 13.61%, and a Sharpe ratio of 1.7339 (table 6 or table 7, the results are the same).

The equally weighted strategy outperformed the complete LSTM and the historical Markowitz portfolios and the benchmark in terms of both final wealth and risk-adjusted performance. It also reported the lowest annualized volatility among the five strategies, thanks to the fact that the composition of the portfolio remained the same and for the high diversification achieved, together with a maximum drawdown of -9.95% (the second-lowest, from table 7). This result confirmed the importance of diversification. By allocating the same weight to all 45 stocks, the portfolio avoided the estimation errors and concentration risks associated with mean–variance optimisation.

However, it remained substantially inferior to the filtered LSTM portfolio, in fact the filtered strategy generated an annualized return approximately 38.63 percentage points higher, while its volatility was only approximately 2.42 percentage points greater.

In fig. 7 it can be seen that the equally weighted portfolio was the second-best portfolio facing the high volatility market period, being under only the filtered LSTM portfolio.

The *EURO STOXX 50 benchmark* achieved final wealth of 1.2296, corresponding to a cumulative return of approximately 22.96%, and an annualized return equal to 21.81%, an annualized volatility of 14.66%, and an annualized Sharpe ratio of 1.2718 (table 6 or table 7, the results are the same).

The benchmark outperformed the historical portfolio but was slightly outperformed by the complete LSTM portfolio in terms of final wealth, but it was also substantially outperformed by the equally weighted and the filtered LSTM portfolios, as provided by fig. 7.

The benchmark experienced a maximum drawdown of -10.64% , mainly during the market decline observed between February and March 2026 (fig. 7).

To understand the *impact of the transaction costs* it is necessary to compare the results between table 6 (no transaction costs) and table 7 (with transaction costs).

The transaction costs had reduced the annualized return of the complete LSTM portfolio by approximately 1.62 percentage points and its final wealth by approximately 0.0171. The largest reduction had been experienced by the filtered LSTM portfolio, its annualized return declined by approximately 2.24 percentage points, while the final wealth decreased from 1.7332 to 1.7092.

The historical Markowitz portfolio was affected less significantly because its weights changed more gradually (lower turnover), with a reduction in the annualized return of approximately 0.44 percentage points, and on the final wealth of approximately 0.0047.

The results therefore showed that the transaction costs were economically relevant especially for both LSTM strategies because of their higher turnover. Nevertheless, the filtered LSTM portfolio remained the strongest strategy even after costs had been included.

By comparing fig. 6 and fig. 7 the paths of the strategies are the same for all the strategies, but in fig. 7 the lines of both LSTM portfolios and of the historical Markowitz portfolio are shifted downwards due to the transaction costs, while the benchmark and the equally weighted portfolio had not been affected.

5.2.5. Overall results and conclusion on the out-of-sample back-test

The out-of-sample back-test produced a different conclusion than the results presented in the previous ex-ante Efficient-Frontier analysis.

In the static optimisation, the complete LSTM portfolio appeared to offer the most favourable expected risk–return combination, however once the portfolios were repeatedly re-estimated and evaluated using realised returns (table 7), the filtered LSTM portfolio generated by far the strongest performance outperforming all the other strategies from all the performance measures point of view. The complete LSTM portfolio only marginally outperformed the EURO STOXX 50 benchmark in terms of final wealth and did not outperform it according to the Sharpe ratio or maximum drawdown. The historical Markowitz portfolio provided the lowest drawdown but also the weakest realised return.

The equally weighted strategy represented a strong benchmark, outperforming the complete LSTM and the historical Markowitz portfolios while maintaining relatively low volatility.

The main result was reached by the filtered LSTM portfolio, the restriction used to the stocks sample of Directional Accuracy above 50% and positive prediction–realisation Correlation coefficient appeared to remove securities for which the model forecasts were less informative. During the analysed period, this filtering strategy substantially improved both cumulative and risk-adjusted performance.

Nevertheless, the filtered strategy experienced a meaningful drawdown in early 2026, but recovering it very quickly (fig. 7), improving the cumulative wealth. Its performance was therefore not free of risk, but the risk was completely repaid. The inclusion of transaction costs remained necessary to make a correct evaluation of the strategy for its practical applicability with real market condition, and the results had been confirmed, filtering was the best optimization strategy.

CONCLUSION

The objective of this thesis was to investigate whether LSTM forecasts could improve portfolio optimization by providing forward-looking estimates of expected stock returns within a traditional Markowitz framework.

The predictive results showed that forecasting short-term stock returns remains difficult. The LSTM achieved an average Directional Accuracy slightly above 50% and generally positive, but relatively low correlations between predicted and actual returns. Meaning that, the model did not provide highly accurate forecasts of the exact magnitude of future returns, but the results suggested that the predictions contained some economically useful information.

The portfolio analysis confirmed this distinction. In the ex-ante optimization, the complete LSTM portfolio achieved the best estimated risk–return trade-off, with an annualized expected return of approximately 36.87% and a Sharpe ratio of 2.10.

However, the most important result emerged from the out-of-sample back-test. After transaction costs, the filtered LSTM portfolio achieved an annualized return of approximately 66.80% and a Sharpe ratio of 3.14, substantially outperforming the complete LSTM portfolio, the historical Markowitz portfolio, the equally weighted strategy, and the EURO STOXX 50 benchmark.

This result suggests that the usefulness of the LSTM forecasts does not necessarily depend on extremely high standalone predictive accuracy, but instead their economic value appears to increase when the forecasts are combined with an appropriate stock-selection criterion and with the diversification and risk-estimation principles of the traditional Markowitz framework.

Overall, the findings suggest that LSTM networks should not necessarily be considered as a replacement for traditional portfolio theory. Rather, they can represent a complementary forecasting tool capable of improving portfolio allocation when their predictions are appropriately selected and combined with traditional risk estimation.

BIBLIOGRAPHY

- Abdellilah, N., Abdellah, Y., & Abdellah, E. (2023). Equity-Market-Neutral Strategy Portfolio Construction Using LSTM-Based Stock Prediction and Selection: An Application to S&P500 Consumer Staples Stocks. (D. K. Nguyen, Ed.) *International Journal of Financial*, 11, 4 - 40. doi:<https://doi.org/10.3390/ijfs11020057>
- AG, D. B. (n.d.). *Brokerage commission*. Retrieved August 4, 2026, from Deutsche Borse Live: <https://live.deutsche-boerse.com/en/know-how/glossary/brokerage-commission>
- Ahmed, S., Bin Alam, M., Hassan, M., Rozbu, M., Ishtiak, T., Rafa, N., . . . Gandomi, A. (2023, April 17). Deep learning modelling techniques: current progress, applications, advantages, and challenges. *Artificial Intelligence Review*, 56, 13521–13617. Retrieved from <https://link.springer.com/article/10.1007/s10462-023-10466-8>
- Andrew, A. M. (1999). *REINFORCEMENT LEARNING: AN INTRODUCTION* by Richard S. Sutton and Andrew G. Barto, *Adaptive Computation and Machine Learning series*, MIT Press (Bradford Book) (Second ed.). Cambridge: C. R. Gallistel.
- Benhamou, E., Saltiel, D., & Abh, S. (2020). Bridging the gap between Markowitz planning and deep reinforcement learning. *Association for the Advancement of Artificial*, 1 - 10.
- Bodie, Z., Kane, A., & Marcus, A. J. (2013). *Investments*, 10th edition. McGraw-Hill Education.
- Boyd, S., Johansson, K., Kahn, R., Schiele, P., & Schmelzer, T. (2024, January 5). Markowitz Portfolio Construction at Seventy. 1 - 54. Retrieved from <https://arxiv.org/pdf/2401.05080>
- Campbell, S., Harvey, R., & Akhtar. (2020). TIME-VARYING CONDITIONAL SKEWNESS AND THE MARKET RISK PREMIUM. *Research in Banking and Finance*, JAI/Elsevier Inc, 2 - 30.
- CFI-Team. (2020, May 7). *Adjusted Closing Price*. Retrieved from CFI: <https://corporatefinanceinstitute.com/resources/equities/adjusted-closing-price/>
- Chen, J.-F., Chen, W.-L., Huang, C.-P., Huang, S.-H., & Chen, A.-P. (2016). Financial Time-series Data Analysis using Deep Convolutional Neural Networks. *International Conference on Cloud Computing and Big Data*, 87 - 92. doi:10.1109/CCBD.2016.027
- Chen, K., Zhou, Y., & Dai, F. (2015). A LSTM-based method for stock returns prediction : A case study of China stock market. *IEEE International Conference on Big Data (Big Data)*, 2823 -

- Fadlalla, A., & Lin, C.-H. (2001, Aug 1). An Analysis of the Applications of Neural Networks in Finance. *31*(4), 112 - 122. doi:<https://doi.org/10.1287/inte.31.4.112.9662>
- Fischer, T., & Krauss, C. (2017). Deep learning with long short-term memory networks for financial market predictions. *FAU Discussion Papers in Economics*, 654 - 669. doi:<https://doi.org/10.1016/j.ejor.2017.11.054>
- Gao, P., Zhang, R., & Yang, X. (2020). The Application of Stock Index Price Prediction with Neural Network. *Mathematical and Computational Applications*, 53.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Cambridge, MA, USA: The MIT Press.
- Gu, S., Kelly, B., & Xiu, D. (2020, May). Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, Vol. 33, No. 5, 2223 - 2273.
- Habib, Z., & Meryem-Nadjat, N. (2025, June). Portfolio Optimization Based on MPT-LSTM Neural Networks: A case study of Cryptocurrency Markets. *Institute for Economics and Politics University of National and World Economy Bulgaria*. doi:10.37075/FABA.2025.1.07
- Hawkins, & Douglas, M. (2003, October 30). The Problem of Overfitting. *Journal of Chemical Information and Computer Sciences*, Vol. 44, No. 1 - 12, 1 – 12.
- J.P.Morgan/Reuters. (1996). *RiskMetricsTM — Technical Document* (Fourth ed.). New York. Retrieved from <https://www.msci.com/documents/10199/5915b101-4206-4ba0-ae2-3449d5c7e95a?>
- Kufel, J. ..., Bargieł-Lączek, K., Kocot, S., Koźlik, M., Bartnikowska, W., Janik, M., . . . Gruszczyńska, K. (2023, August 3). What is Machine Learning, Artificial Neural Networks and Deep Learning? — Examples of Practical Applications in Medicine. (Q. A. Dechang Chen, Ed.) *Diagnostics*, 13 - 2582. Retrieved from <https://doi.org/10.3390/diagnostics13152582>
- Livingstone, D., Manallack, D., & Tetko, I. (1997). Data modelling with neural networks: Advantages and limitations. *Journal of Computer-Aided Molecular Design*, 135 - 142. doi:<https://doi.org/10.1023/a:1008074223811>
- Ma, Y., Han, R., & Wang, W. (2020). Prediction-Based Portfolio Optimization Models. *School of Economics and Management, Southeast University, Nanjing 211189, China*, 115393 -

- Markowitz. (1989). *Mean—Variance Analysis*. London: Palgrave Macmillan.
doi:https://doi.org/10.1007/978-1-349-20213-3_21
- Markowitz, H. (1952, March). Portfolio Selection. *The Journal of Finance*, Vol. 7 No. 1., 77 - 91.
- Martínez-Barbero, X., Cervelló-Royo, R., & Ribal, J. (2025). Portfolio Optimization with Prediction-Based Return Using Long Short-Term Memory Neural Networks: Testing on Upward and Downward European Markets. *Computational Economics*, 65, 1479 – 1504.
doi:<https://doi.org/10.1007/s10614-024-10604-6>
- MathWorks. (n.d.). *Reinforcement Learning*. Retrieved from The MathWorks, Inc:
<https://www.mathworks.com/discovery/reinforcement-learning.html>
- Michuad, & O, R. (1989). The Markowitz Optimization Enigma: Is 'Optimized' Optimal? *Financial Analysts Journal*, 31 – 42. Retrieved from <https://doi.org/10.2469/faj.v45.n1.31>
- Montemurro, L. (2026, 02 20). *Aggiornamento tassi Eurirs ed Euribor - 19 febbraio 2026*. Retrieved from Facile.it: <https://www.facile.it/mutui/news/aggiornamento-tassi-eurirs-euribor-19-febbraio-2026.html>
- Pokhrel, A. (2021/2022). Directional movement prediction of stock returns using LSTM and Tree-Based models. 1 - 42.
- Priya, S., Manoj, J., Mohamed, S., Mohamed, A.-m., & Thippa reddy, G. (2023, Sep 27). Harnessing a Hybrid CNN-LSTM Model for Portfolio Performance: A Case Study on Stock Selection and Optimization. *IEEE Access*, 11, 104000-104015. doi:10.1109/ACCESS.2023.3317953
- Rather, A. (2021). LSTM-based Deep Learning Model for Stock Prediction and Predictive Optimization Model. *EURO Journal on Decision Processes*, 100001.
- Sen, J., Dutta, A., & Mehtab, S. (2021). Stock Portfolio Optimization Using a Deep Learning LSTM Model. *IEEE Mysore Sub Section International Conference (MysuruCon)*, 263 - 271.
- Sen, J., Mondal, S., & Mehtab, S. (2021). Portfolio Optimization on NIFTY Thematic Sector Stocks Using an LSTM Model. *International Conference on Data Analytics for Business and Industry (ICDABI)*, 364 - 369.
- Siami Namin, S., & Siami Namin, A. (2018). FORECASTING ECONOMIC AND FINANCIAL

- Sood, S., Papasotiriou, K., Vaiciulis, M., & Balch, T. (2023). Deep Reinforcement Learning for Optimal Portfolio Allocation: A Comparative. *J.P. Morgan AI Research*, Papers 2602.17098.
- Srinivasan, A. (2023). Reinforcement Learning: Advancements, Limitations, and Real-world Applications. *International Journal of Scientific Research in Engineering and Management (IJSREM)*. doi:10.55041/IJSREM25118
- Tu, & V., J. (1996). Advantages and Disadvantages of Using Artificial Neural Networks versus Logistic Regression for Predicting Medical Outcomes. *J Clin Epidemiol*, vol. 49, no. 11, 1225 - 1231.
- Van-Dai, T., Chuan-Ming, L., & Direselign, A. T. (2020, January 7). Portfolio Optimization-Based Stock Prediction Using Long-Short Term Memory Network in Quantitative Trading. *applied sciences*, 437.
- Yashaswi, K. (2021). Deep Reinforcement Learning for Portfolio Optimization using Latent Feature State Space (LFSS) Module. Papers 2102.06233.
- Zhang, X., & et al. (2019). AT-LSTM: An Attention-based LSTM Model for Financial Time Series Prediction. *IOP Conference Series: Materials Science and Engineering*, 052037. Retrieved from <https://iopscience.iop.org/article/10.1088/1757-899X/569/5/052037>
- Zhang, Y., Su, Y., Liu, W., & Yang, X. (2022, September 10). Portfolio optimization with LSTM-based return and risk information. *School of Management, Guangdong University of Technology, Guangzhou, Guangdong 510520, China*, 39. doi:<https://dx.doi.org/10.2139/ssrn.4215299>
- Zhong, Z. (2024). Intelligent Financial Modeling Data Analysis Based on MATLAB. *Second International Conference on Networks, Multimedia and Information Technology (NMITCON)*, 1 - 5.
- Zou, J., Huss, M., Abid, A., & Moha, P. (2024). A primer on deep learning in genomics. *Nat Genet*, 12 - 18.
- Zrara, L. (2021, January). Portfolio optimization using deep learning for the moroccan market. *School of science & engineering – AL AKHAWAYN UNIVERSITY*, 45.

MATLAB SCRIPT AND DATA-SET

LSTM model



LSTM_model.pdf

Portfolio Backtest



Portfolio_Backtest.pdf

Portfolio optimization



Portfolio_optimization.pdf

splitBuiltInTimeSeries normalizeBuiltInSequences



splitBuiltInTimeSeries.pdf



normalizeBuiltInSequences.pdf

Benchmark



EURO STOXX50.xlsx

Data Set



data set.xlsx