Contents

Introduction

Institutional investors, financial advisors and portfolio managers build assets portfolio for their clients using different portfolio optimization techniques. Over the last few decades, many strategies have been developed to help these professionals (e.g., Markowitz model, Black-Litterman model and Monte Carlo simulation). Perold and Sharpe (1988) and Zhang et al. (2020) report portfolio optimization solution in a dynamic environment. They empirically prov that under certain circumstances it could be more convenient to adjust portfolio weights over time. These models update the way to solve portfolio optimization problem thanks to more data and computational resource available. Regarding the last point, is it possible to exploit this volume of data through machine learning models to provide even more reliable and efficient solutions?

Beltratti (2022) mentions two main approaches to portfolio management: tactical allocation and strategic allocation. The first one is defined over a short horizon of time (i.e., between 1 to 3 months). The second one runs over a time horizon of at least 2 or 3 years and it is solely based on historical time series. Due to market volatility and uncertainty, investors goals and technological advancements, institution started providing dynamic portfolio allocation solutions. Tsai and Wang (2020), Liu et al. (2021) and Awad et al. (2023) develop and suggest new predictive modeling algorithms which aim to build portfolios by constantly purchasing and selling assets over a specific period of time. Most of these models are only focused on providing a buy-neutral-sell schema without actually returning portfolio weights. As an example, Almahdi and Yang (2017) provides a model exclusively focused on obtaining buy, sell or neutral signals. The time frame within which decisions are made is generally lower than one day. This could lead to notably high transaction costs. Moreover, it is often assumed that short selling is possible, an activity that is generally accessible only by large financial institutions. Most of the evidence provided are primarily based on datasets composed by technical indicators. Macroeconomics factors and news analysis through model of sentiment analysis are often set apart and treated individually. The assets on which

the model training and evaluation are provided follows a discretional pattern, on most occasions, they are cryptocurrencies.

Through this thesis work, I develop a model that parametrize a policy function with the objective to maximize the Sharpe ratio over a specific period of time. In accordance with changes in the financial market, the model adjusts the portfolio weights every 15 days. Portfolio weights vectors are strictly positive, i.e., short selling is not allowed. I train and evaluate the models on dataset composed by statistical information (e.g., standard deviation on different period of times and VIX Index mean return over the last 30 days). I perform several experiments in a combination of technical information and sentiment analysis on financial macroeconomics and industry-specific news. Both for the modeling and evaluation phases, I used as reference dataset the 12 industry portfolio returns dataset composed by Kenneth R. French and Eugene Fama. In this way, the dataset is able to capture a wide range of economic industries and to provide a well-diversified representation of the market.

My main findings are as follows. First, in some cases, the daily mean returns produced through the machine learning models outperform the ones of the tangency portfolio obtained through the Markowitz model. Second, although the returns obtained through the machine learning model are higher, the Sharpe ratio is lower due to their more volatile returns. Third, machine learning model architectures based on convolutional layers perform better than the ones based on recurrent layers. Fourth, models trained through datasets containing sentiment analysis features underperform on some occasions with respect to model trained without these features.

My thesis makes three main contributions to the literature. First, it provides a new approach to portfolio optimization techniques based on deep reinforcement learning. The dynamic portfolio optimization problem involves adjusting portfolio weights on a defined period of time. Our objective is to find an optimal decision policy function that is capable of performing properly across different scenarios according to one or more specific variables (e.g., Sharpe ratio, Sortino ratio and return). Cheng et al. (2018),

Fisher et al. (2018) and Hoseinzade et al. (2019) are concerned on developing models based on a supervised learning approach (e.g., random forest, support vector machine and neural networks). Moreover, these models are implemented to solve other type of financial-related problems (e.g., buy-sell stock decision, return prediction and market condition classification).

Second, the thesis combines aspects coming from the unsupervised learning paradigm, in particular, the sentiment analysis, together with a reinforcement learning approach. Rouf et al. (2021) and Fazlija et al. (2022) prov that macroeconomics information and industry specific market news provide useful information regarding the general market sentiment. Some of the performed experiments include datasets composed by sentiment analysis on financial news through a large language model (LLM). Hedayati Moghaddam et al. (2016) and Chong et al. (2017) provide models that analyze the stock market only through statistical information on specific assets in a short range of time (e.g., relative strength index and moving average convergence/divergence). The thesis offers a more general framework which aims to model a policy function able to build portfolio exploiting both technical and fundamental information.

Third, the models built include both convolutional and recurrent layers based neural networks. Cheng et al. (2018), Fischer et al. (2018) and Moghar et al. (2020) provide time series analysis on financial assets implementing recurrent layers based neural networks. Leaving the implementation of convolutional layers based neural networks for other machine learning fields (e.g., computer vision and speech recognition). Recent works showed that also convolutional neural networks is capable of finding useful patterns inside the time series. In particular, these types of neural networks are good in finding local pattern and modelling short-term dependencies. Differently from the traditional recurrent neural networks that are capable of finding patterns and dependencies over long-term time series. At the same time, it is important to note that

the recurrent neural networks are generally built implementing layers that slow down the training process due to their considerable high level of complexity.

Data and Variables of Interest

I build the dataset using: (1) 12 industry portfolios daily returns dataset prepared by Eugene F. Fama and Kenneth R. French; (2) financial news on Google News; and (3) macroeconomics features from Federal Reserve Economic Data (FRED). The datasets associated with each data source have different periods of availability. Data preprocessing was individually performed for each dataset. Once the processes of data collection, data cleaning and data transformation were completed, the datasets were aggregated. In this way, I obtain greater flexibility during the modeling phase.

A. 12 Industry Portfolio Daily Returns

I base both the training and the evaluation phase of the model on this dataset. The dataset categorizes firms into twelve different industries (e.g., consumer durables, consumer nondurables and manufacturing). The selection of the companies for each of the industries is based on the four-digit SIC codes. The dataset has been chosen for computational reasons. Its low matrix dimensionality permits less computational time expense with respect to other datasets composed by many more industries (e.g., 38 or 49 industry portfolio).

In accordance with the CFA Institute, the dataset is not able to properly consider dynamics and market changes constantly characterizing industries. Some of the SIC codes definitions could include firms that present different financial and economical characteristic. For this thesis work, the Fama-French dataset is an optimal solution for its capability to represents market dynamics for each of the most important industries composing the U.S. Economy. It is a good trade-off balance between the representative capacity of U.S. financial market and computational costs reduction.

The dataset is freely available online in two versions: Average Value Weighted Returns and Average Equal Weighted Returns. For this work, I chose the Average Equal Weighted Returns version and I added features considered impactful according

to the literature: momentum, individual industry risk measure and CBOE Volatility Index. Fama and French (1996) recognizes the existence of momentum effect, defined as an anomaly characterizing financial assets. Jegadeesh and Titman (1993) perform the same analysis and noticed that the strategy of buying stocks that showed high returns performances and selling stocks that performed badly in the past could lead to significant returns in the next 3- to 12-months period. For these reasons, the momentum has been computed for each industry for different periods in particular: 15, 40 and 70 days. Respectively to obtain numbers that could represent: short, medium and long term momentum effect on industries market.

The standard deviation for each industry return represents another feature characterizing the dataset. More specifically, I was concerned to add a feature that could represent a risk of measure for each of the industry of the financial market. The use of standard deviation as a measure of risk is largely attributed to Markowitz. Moreover, also Brock et al. (1992) contribute to this statement, providing empirical support. Thanks to its ease of computation, I computed the periodic standard deviation with respect to each single industry. The periods chosen are: 15, 40 and 70 days, as already done for the momentum feature.

As another measure of risk, I added to the dataset the CBOE Volatility Index. With respect to the twelve industries portfolio returns dataset, the VIX adds more general and aggregated information regarding the market, differently from the periodic single values of standard deviation computed for each industry. This tool is a measure of the stock market's expectation of volatility based on the S&P500 index options. I incorporated this index as a proxy for: market stability and investor uncertainty. Bekaert, Hoerova and Lo Duca (2010) support the assertion of the predictive power of VIX for stock market returns. The financial information on the index has been fetched through the Yahoo Finance API. In particular, on the close prices obtained, I computed: the returns and the standard deviation. Both these two measures are computed over three different periods of time: 5, 20 and 40 days. The data relative to this index are

only available from the 1st January 1990. Differently, the data of the 12 industries dataset are available from the 1st July 1926.

B. Sentiment Analysis

Sentiment Analysis is defined as a tool able to identify, measure and quantify subjective information. More specifically, for my purposes, an algorithm of sentiment analysis must be able to properly encode the article of a specific financial news into a distribution of three possible states. The state space is identified through the following set: $\{0, 1, 2\}$, respectively representing positivity, negativity and neutrality of the news sentiment. Performing a sentiment analysis on a news article means obtaining the associated probabilities for each of the three possible states. As an example, given an article whose text mentions the outstanding points increase the Dow Jones had on a specific day, the sentiment analysis should be able to return a high probability value for the positive sentiment and respectively, low probability values for both the negative and the neutral sentiment.

In this work, I perform the sentiment analysis through a large language model (LLM). It is a class of machine learning models whose objective is to understand the text and, subsequently, performing activities such as: text completion, translation and question answering. I implement a model named Bidirectional Encoder Representations from Transformers (BERT) and developed by Google AI Language Team in the 2018. It is important to note that this model can easily finetuned, making it one of the most common models used for: text classification, text summarization and question answering. The process of finetuning allows the model to better performance at tasks it has been designed for. For this thesis, I implement a finetuned version of BERT called FinBERT. It is so called because the model has been further trained (finetuned) on a dataset composed by financial news coming from the most popular English and American publishers. Each news is labeled with one of the three states mentioned above (positive, negative and neutral sentiment). More precisely, the dataset is freely available online and it is titled Financial PhraseBank. According to

Araci (2019), when the article presents a complex semantic structure, FinBERT is not sometimes able to properly distinguish neutral sentiment news from positive or negative sentiment news. This incapacity is offset by certain precautions in the composition of the final dataset.

For my thesis, I perform sentiment analysis with respect to each of the 12 industries composing the dataset. I collected through the Google News API, all the financial and macroeconomics news related to each industry. Google News offers a wide range of financial-related journal websites (e.g., Yahoo Finance, CNBC and Bloomberg). For this reason, it is important to properly formulate questions able to avoid any possible misleading and non-institutionally accredited newspapers.

The dataset has been composed as follows. Starting from the 1st January 2010, every ten days, I collected, through the Google News API, all the financial news for each of the industries. I performed the sentiment analysis through the FinBERT model for each of the article and for each industry. Computed the mean positive sentiment probability for the articles for each industry. I discarded all the articles whose neutral sentiment was particularly high ($s_n > 0.5$), since they could have potentially decreased the mean positive sentiment value. In this way, I increased the probability the sentiment analysis was performed in a more accurate way. The news has collected starting from 1st January 2010, due to imposed limits on the number of requests. The preparation of the dataset took a particular high amount of time. For each of the news article available, I stored the text and performed sentiment analysis over it. The dataset is available from the 1st January 2010 up to the 31st December 2020.

Shi and Fan (2023) propose an innovative approach to perform sentiment analysis. They built a probabilistic model that exploits Bayesian deep neural network to compute sentiment levels on tweets. The authors design a model able to constantly update the probabilities associated to the three main sentiment states anytime new information is released. However, the implementation of this type of model is not properly suited for news article whose text is longer and more complex compared to tweets.

C. Federal Reserve Economic Data

Other more technical indicator composes the final dataset. In particular, the U.S. daily inflation has been included in the analysis. Rouf et al. (2021) highlights the importance of some macroeconomics variables for stock prediction (e.g., GDP, inflation and government policies). The U.S. daily inflation rate is an important measure that could impact the market dynamics and investor behavior. This variable has been downloaded through the database of FRED and added to the main dataset.

Model

In this work, I develop a deep reinforcement learning model to build and parametrize a policy function able to dynamically adjust the portfolio weights over a specific period of time. More specifically, two different variants of the deep deterministic policy gradient (DDPG) algorithm are implemented. The first one provides a neural network composed by convolutional layers. The second one provides a recurrent neural network built using long short-term memory layers.

A. Reinforcement Learning Paradigm

The machine learning field is composed by three main different type of paradigms: supervised learning, unsupervised learning and reinforcement learning. The first one involves the model to train through input-output pairs. Activities generally performed through supervised learning models are: image classification, fraud detection and medical diagnosis. The objective is to parametrize a function able to correctly map inputs to outputs. Both inputs and inputs are present in the dataset. The second one, involves the model to train only through a dataset composed by data without any label. Among the most common tasks executed: dimensionality reduction, clustering and anomaly detection. The objective is to find hidden patterns within collection of unlabeled data. The third one, involves the training of a model through the interaction of an agent with an environment. The paradigm aims to parametrize a function (known as policy function) able to achieve optimal solution both in the short and the long-term run, given a stochastic environment, as the financial market is. In more simple terms,

the agent (parametrized function) learns through a trial-and-error approach by interacting with the environment. There are different areas of application: robotics, involving training robots to perform human-like activities; autonomous driving, involving the development of car-system to drive and act autonomously; game playing, involving training of model to play games (e.g., chess and poker).

Although the significant impact the reinforcement learning had in the machine learning field, there are various and important related challenges: exploration-exploitation trade-off, reward function design, policy function approximation. Regarding the first problem, the agent should theoretically prefer actions that in the past gave to him a high reward (exploitation). On the other hand, the agent should try and discover new actions that could potentially lead him to even higher reward (exploration). With reference to this work, given an already experienced financial scenario, the agent should choose the same portfolio weights that gave to him a high reward in the past. At the same time, the agent could also try to set new portfolio weights that might bring him to a higher reward, although this is a risk.

Regarding the second problem, the agent aims to maximize the reward both in the short and in the long-term. For an autonomous driving problem, the car-system agent continuously receives numerical feedback regarding the way it is driving. As an example, the feedback (reward) could be defined as a linear combination of variables such as: time spent driving, obeying the law and lane keeping. For each of the variable composing the reward function, there is a coefficient representing the level of importance of the associated variable. The challenge relates to the ability to properly design a feedback system that contains significant variables and give the proper weights for each of them. Otherwise, the agent will not be able to properly train with respect to the objective we designed the model for.

Regarding the third problem, the agent receives, at each time step, new information before deciding the action to take. The information is generally represented by a vector. The challenge arises when dealing with a tremendous amount of data. High-

dimensional data generally require the policy function approximation (agent) to be parametrized through a neural network. This approach could lead to unstable and overfitting results, particularly, considering the non-stationarity of financial time series. To solve this problem, there are two important fronts to work on: implementation of regularization techniques and design of optimal neural network architecture. Through the first one, overfitting and training stability problems are generally solved. Regarding the second one, it is important to follow good practices on neural network design, as specified by literature.

The algorithm of reinforcement learning implemented for this thesis work is named: Deep Deterministic Policy Gradient (DDPG). It has been developed by Google DeepMind in 2016. The research group aimed to build an algorithm able to solve complex activities whose data inputs were in a high-dimensional continuous real or complex valued space ($\mathbb{R}^n$ or $\mathbb{C}^n$). Bao and Liu (2019) and Liu et al. (2021) developed models to solve financial stock trading problems through reinforcement learning in discrete action space. The decision relative to portfolio management were treated as a vector whose components could only be 0, 1 or 2. The first number represented the neural position, the second number the long position and the third number the short position. Through this algorithm I am able to parametrize a policy function whose output belong to $\mathbb{R}^{\mathrm{n}}$, with n representing the number of assets composing the portfolio, more specifically, the vector of portfolio weights.

Before concretely moving into the analysis of my algorithm, it is important to define the way the reinforcement learning problem are designed. The Markov Decision Process is the mathematical formal framework that defines the way agent and environment interacts. There are three essential elements that define the problem to solve: state space, action space and the reward function. With respect to the dynamic portfolio optimization problem, the state space is represented as a matrix whose rows represent the days and the columns represent the features. The action space represents the set of all possible actions that an agent could potentially perform in the

environment. In this particular case, the action space is represented by A^{n}, with A being the interval $[0, 1] \subset \mathbb{R}$ and n representing the number of assets by which the portfolio is composed. It is important to note that for each action vector $\vec{\mathrm{a}} \in \mathrm{A}^{\mathrm{n}}$ taken by the agent, the sum of its components must equal 1 (i.e., $\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{a}_{\mathrm{i}} = 1$). The reward function is a fundamental component for our agent to know whether its action is performing well with respect to our environment, more precisely, our objective function. In the dynamic portfolio optimization problem, an example of reward function could be a linear combination of variables such as: Pearson linear correlation among assets, portfolio returns and standard deviation on returns. In this work, the reward function is defined as the difference of the portfolio Sharpe ratio obtained through the weights of the model and the portfolio Sharpe ratio generated by the tangency portfolio in the Markowitz optimization model.

It is important to note that over the last few years, many new approaches have been developed. Many of these belong to the class of generative AI models, among them: Generative Adversarial Network (GAN) and Variational Autoencoders (VAE). Others are updated version of what I implemented, as an example: attention layers embedded with LSTM layers. The latest has been proposed by Cheng et al. (2018) and Zhang et al. (2019). Through this approach the analysis on non-stationary sequences and non-linear relationships can be captured more efficiently. At the same time, computational cost required by the architecture of this type of model impose the use of high-performance computer.

Figure 1 – Simplified version of the algorithm

The figure below describes synthetically the training process relative to the DDPG algorithms implemented. I define the $\mathbf{X}$ as the features dataset, $\mathbf{R}$ as the dataset of returns (i.e., 12 industries Fama-French dataset), $\mathbf{G}$ as the vector of the risk-free rates. Then I define the policy function $\pi(\cdot)$, the function that takes the portfolio decision in accordance with the inputs received. It returns the vector containing the new weights of the portfolio. I define the function $m(\cdot)$ that returns the weights of the tangency portfolio obtained through the Markowitz optimization process. The while loop contains all the operations applied to compute the Sharpe ratio and to update the parameters of the model. The training process could take long time in accordance with the complexity of the datasets and the models implemented.

Algorithm 1: Simplified Functioning of the Algorithm

Data: $\mathbf{X} \in \mathbb{R}^{m \times n}; \mathbf{R} \in \mathbb{R}^{m \times k}; \mathbf{G} \in \mathbb{R}^{m}; \pi(\cdot), m(\cdot) \colon \mathbb{R}^{d \times n} \to \mathbb{R}^{k};\ m, n, k, d \in \mathbb{N}^{+}, t \in \mathbb{N}$

$m \leftarrow 252;$
$n \leftarrow 64;$
$k \leftarrow 12;$
$t \leftarrow 0;$
$y \leftarrow [\,0,\ 0, \ldots,\ 0\,];$
while $t \neq m$ **do**
  $\mathbf{x} \leftarrow \mathbf{X}[t \colon t + d - 1, :\,],\ \mathbf{x} \in \mathbb{R}^{d \times n};$
  $\vec{y} \leftarrow \pi(\mathbf{x});$
  $\vec{h} \leftarrow m(\mathbf{x});$
  $\mathbf{E} \leftarrow \mathbf{R}[t + d \colon t + 2d, :\,],\ \mathbf{E} \in \mathbb{R}^{d \times k};$
  $\vec{r}_{\pi} = \mathbf{E}\vec{y},\ \vec{r}_{\pi} \in \mathbb{R}^{d};$
  $\vec{r}_{m} = \mathbf{E}\vec{h},\ \vec{r}_{m} \in \mathbb{R}^{d};$
  $\mu_{\pi} = \frac{\vec{r}_{\pi}^{T}\mathbf{1}}{d};\ \mu_{m} = \frac{\vec{r}_{m}^{T}\mathbf{1}}{d};$
  $\vec{D}_{\pi} = \vec{r}_{\pi} - \mu_{\pi}\mathbf{1};\ \vec{D}_{m} = \vec{r}_{m} - \mu_{m}\mathbf{1};$
  $\sigma_{\pi}^{2} = \frac{1}{d}\vec{D}_{\pi}^{T}\vec{D}_{\pi};\ \sigma_{m}^{2} = \frac{1}{d}\vec{D}_{m}^{T}\vec{D}_{m};$
  $\sigma_{\pi} = \sqrt{\sigma_{\pi}^{2}};\ \sigma_{m} = \sqrt{\sigma_{m}^{2}};$
  $\vec{g} = \mathbf{G}[t + d \colon t + 2d];$
  $\mu_{g} = \frac{\vec{g}^{T}\mathbf{1}}{d};$
  $s_{\pi} = \frac{\mu_{\pi} - \mu_{g}}{\sigma_{\pi}};\ s_{g} = \frac{\mu_{m} - \mu_{g}}{\sigma_{\pi}};$
  $b = s_{m} - s_{\pi};$
  if *Training Condition* **then**
    $\pi \leftarrow \mathrm{UpdatePolicy}(\pi, \mathrm{data});$
  end
  $t \leftarrow t + d;$
end

B. Convolutional Neural Network

To parametrize the policy function of the agent, I proposed two different approaches: the convolutional neural network and the recurrent neural network. LeCun et al. (1998) proposed a new way to solve the problem of handwritten digit recognition through the use of neural networks. They studied and invented a new type of layer that is based on the mathematical concept of convolution. The original real-world field of application concerned about images, thereafter, Hoseinzade and Haratizadeh (2019) proposed the implementation of deep learning techniques (i.e., neural networks model) based on convolutional layers in the financial time series prediction activity. The results showed a significant performance superiority with respect to other machine learning algorithms, such as: principal component analysis and artificial neural networks, support vector machine and naïve bayes.

Table 1 – Architecture of the convolutional neural networks

The figures represent the architecture composing the convolutional neural network. For each layer, it is defined the type of layer, the shape of the layer and the eventual transformation applied to the output of the layer (i.e., activation function). With respect to the input layer, the variable delta_days represents the number of days chosen before adjusting the portfolio weights. The variable n_features represents the number of features composing the dataset (e.g., standard deviation and positive sentiment). A sequence of mathematical transformations is applied to the initial inputs. Convolutional (Conv1D) and batch normalization layers are the most important ones. These layers aim to identify hidden patterns over the input data. Through the latest layer, the output coming from the previous layers are standardized through the SoftMax activation function: the output corresponds to a vector whose dimension equals to the number of assets of the portfolio and components are positive and sum up to 1. The variable n_actions represents the number of assets composing the portfolio.

Layer Type	Layer Shape	Activation Function
Input	(delta_days, n_features)	-
Conv1D	512	-
BatchNormalization	-	-
Dense	-	ReLU
Conv1D	256	-
BatchNormalization	-	-
Dense	-	ReLU
Conv1D	64	-
BatchNormalization	-	-
Dense	-	ReLU
GlobalAveragePooling1D	-	-
Flatten	-	-
Dense	n_actions	SoftMax

The main reason behind the use of convolutional layers stands for the ability to capture short term trend and seasonality in time series. This feature allows the neural network to significantly enhance prediction accuracy, according to the literature. Within the structure of the neural network, the model built present layers of Batch Normalization whose task is to accelerate and stabilizing training. Ioffe and Szegedy (2015) proposed for the first time these types of layers, allowing researchers to achieve quicker the training convergence and adding a regularization effect. Regarding the “GlobalAveragePooling1D” layer: Lin, Chen and Yan (2014) proposed this new type of layer. Differently from the Batch Normalization layer, the Global Average Pooling has the only purpose of reducing the probability of overfitting during the training process. These two types of layers have been both implemented to improve training efficiency and model performance.

The final activation function relative to the last layer is represented by the SoftMax function: $SoftMax(\vec{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$. It is defined as function whose output belongs to $\mathrm{A}^{\mathrm{n}} \subset \mathbb{R}^{\mathrm{n}}$, with $\mathrm{A} = [0, 1]$ and n representing the number of assets composing the portfolio. As expected, the components of the vector output of the SoftMax function sum up to 1.

C. Recurrent Neural Network

Rumelhart, Hinton and Williams (1986) introduced the recurrent neural networks (RNN) that are fundamental for activities such as: time series analysis, text translation and speech recognition. Generally, there are three main classes of layers implemented in RNN: standard RNN layer, LSTM layer and gated recurrent unit (GRU) layer. I built the architecture of the neural network using long short-term memory (LSTM) layers. Hochreiter and Schmidhuber (1997) developed and suggested the LSTM architecture for the first time. The main challenges solved through the implementation of this type of layer are: vanishing gradient problem and inability to capture long term dependencies. The first problem appeared in an incipient version of my RNN model. In particular, given the complexity of the initial neural network architecture, the model was no able to properly train: the cost function minimization process was slowing down and no effective results were obtained. The second problem is a main consequence of the first one. As proposed by Joyeux (2001), Ubal et al. (2023) and Maitra et al. (2023), financial time series present long-term dependencies. Given the mathematical structure of the LSTM layers, they are capable of capturing much better these dependencies compared to the standard RNN layers.

The use of a decreasing number of parameters for each of the three LSTM layers (256, 128 and 64 parameters) is justified by the intention of capturing respectively short, intermediate and long-term dependencies with respect to each feature in the dataset. This type of structure is suggested by LeCun, Bengio and Hinton (2015) and it is known as hierarchical feature extraction. The approach to add even more layers one after the other is decisive to obtain even more information and potentially discover

complex patterns behind the features. As an example, Fischer and Krauss (2018) implemented a LSTM based neural networks and noticed that most of the anomalies characterizing the stock market were correctly identified. Among the anomalies: short-term reversal, stocks prices that significantly fall for a certain period, tend to rise again soon after and vice versa; weak momentum, the 10 top and flop stocks listed at NYSE and AMEX (as Jegadeesh and Titman reported in 1993) shows lower momentum effect compared to the others.

Differently from the convolutional neural network architecture, I do not apply any activation function. Hamilton (1994) and Tsay (2010) mention the importance to capture dependencies and patterns within the financial time series without applying non-linear transformation over the intermediate outputs generated.

Table 2 – Architecture of the recurrent neural networks

The figures represent the architecture composing the recurrent neural network. For each layer, it is defined the type of layer, the shape of the layer and the eventual transformation applied to the output of the layer (i.e., activation function). With respect to the input layer, the variable delta_days represents the number of days chosen before adjusting the portfolio weights. The variable n_features represents the number of features composing the dataset (e.g., standard deviation and positive sentiment). A sequence of transformations is applied to the initial inputs. The LSTM layers aim to identify hidden patterns over the input data by performing a long list of mathematical operations over the inputs (Hochreiter and Schmidhuber, 1997). Through the latest layer, the output coming from the previous layers are standardized through the SoftMax activation function: the output corresponds to a vector whose dimension correspond to the number of assets of the portfolio and components are positive and sum up to 1. The variable 'n_actions' represents the number of assets composing the portfolio.

Layer Type	Layer Shape	Activation Function
Input	(delta_days, n_features)	-
LSTM 1	256	-
LSTM 2	128	-
LSTM 3	64	-
Dense	n_actions	SoftMax

Results and evaluations

In this section, I am going to present the results of the experiments performed, together with an evaluation for each of them. It is important to highlight that each of the experiment is classified in accordance with the type of neural network layers composing the model and the features included into the dataset. The first distinguishes the experiment where convolutional neural networks or recurrent neural networks are

implemented. The second distinguishes the experiment whose dataset contains or not the level of positive sentiment for each of the industry of the Fama-French dataset. In this section, I present four version of the model. I trained each of them with data going from the 1st January 2010 up to 31st December 2015. There are two reasons for this small range of dates: the computational cost associated to the training process and the complex structure of the neural networks. The first problem relates to the limits imposed by the computational resources available. This restricts the access to a quicker training process. The second problem refers to the inability to construct more complex neural networks which would have allowed a more proper features extraction approach. This problem restrains the learning ability of the model.

I evaluate my model over four different metrics: period return, annualized daily mean returns, annualized standard deviation and annualized Sharpe ratio. The performance metric obtained through the model are compared with respect to two other types of portfolios. The first type is defined through a traditional mean-variance optimization technique applied to a dataset of 1-year of daily returns. I compute the portfolio weights that maximize the Sharpe ratio over this 1-year period of time (i.e., tangency portfolio of the efficient frontier). I estimate the aforementioned metrics over the subsequent period of time. The second type of portfolio follows the same initial structure defined for the first type of portfolio. In this version, the portfolio weights are adjusted every 15 days. The data used to adjust the portfolio weights are the 60-daily returns preceding the day scheduled for the adjustment. I named respectively the first and the second versions of the portfolio: ‘Static Markowitz’ and ‘Dynamic Portfolio’. For each experiment performed, I set the beginning date of the evaluation process, the number of periods over to evaluate the models and the length of each period.

Table 3 and 4 show the number and the type of features included in the two versions of the datasets. The first version includes only statistical information. It includes the returns of the Fama-French dataset, the momentum and the standard deviation computed over the returns of the same dataset. More specifically, both the

momentum and standard deviation are elaborated over the three different period of times in order to obtain information on a short, medium and a long-term perspective (15, 40 and 70 days of periods). Moreover, information on the CBOE Volatility Index is included: daily close prices, returns and standard deviation of the daily close prices. Both returns and standard deviation are elaborated over three different period of time: 5, 20 and 40 days. The second version of the dataset follows the same structure described for the first version, but it also includes the percentage of the positive sentiment of the financial news.

Table 3 – Features of the dataset without sentiment analysis

The table shows all the features composing the version of the dataset without sentiment analysis. The first column represents the type of source of data. The second column represents all the types of data or calculations performed over a specific source of data. The third column represents the period of days relative to each feature (e.g., standard deviation with 15 period days represents the standard deviation of the returns over the last 15 days). The fourth column represents the number of features associated to each type of data. The dataset contains exclusively statistical information going from the 1st January 2010 up to the 31st December 2015. The number of features amounts to 92.

12 Industries Portfolio Fama-French	Returns	1	12
	Standard Deviation	15, 40, 70	36
	Momentum	15, 40, 70	36
Volatility Index	Close Price	1	1
	Returns	5, 20, 40	3
	Standard Deviation	5, 20, 40	3
Inflation Rate	Rate	1	1

To properly compare the metric performances of my model with the other two portfolios, I performed various statistical tests. In particular, for each of the three elements of comparison, I defined three indicator functions and I estimate the parameters associated to each of the function in the following equation:

$$V_{1,2,3} = \alpha_1 I_1 + \alpha_2 I_2 + \alpha_3 I_3 \quad (1)$$

where $V_{1,2,3}$ represents the vector concatenating the metric performance (e.g., period return, annualized mean or annualized Sharpe ratio) for each of the three types of portfolios, I_i, $i = 1,2,3$ represents the indicator functions, and α_i, $i = 1,2,3$ are the parameters to estimate. Then two statistical tests are performed. The first one compares the performances of the portfolios respectively obtained through my model and the traditional mean-variance optimization. The second one compares the

performances of the portfolios respectively obtained through my model and the 'Dynamic Markowitz'.

Table 4 - Features of the dataset with sentiment analysis

The table shows all the features composing the version of the dataset without sentiment analysis performed on financial news. The first column represents the type of source of data. The second column represents all the types of data or calculations performed over a specific source of data. The third column represents the period of days relative to each feature (e.g., standard deviation with 15 period days represents the standard deviation of the returns over the last 15 days). The fourth column represents the number of features associated to each type of data. Both the statistical and the news information refer to the period going from the 1st January 2010 up to the 31st December 2015. The number of features amounts to 104.

12 Industries Portfolio Fama-French	Returns	1	12
	Standard Deviation	15, 40, 70	36
	Momentum	15, 40, 70	36
Volatility Index	Close Price	1	1
	Returns	5, 20, 40	3
	Standard Deviation	5, 20, 40	3
Inflation Rate	Rate	1	1
Sentiment Analysis	Positive Sentiment Percentage	-	12

I perform a total number of eight experiments. The first four experiments start at the 1st January 2016. Each of them is composed by 12 periods of evaluation and each of the twelve periods lasts 6 months. The last four experiments keep the same number of periods of evaluation and the length of 6 months for each of them. Instead, the experiments are performed from the 1st January 2017. Each of the two groups of experiments are distinguished in four different versions of the experiment due to: type of neural networks implemented (i.e., convolutional neural networks and recurrent neural networks) and the features included in the dataset (i.e., dataset with sentiment analysis and dataset without).

A. First group of experiments (From 1st January 2016 to 1st June 2017)

Table 5 presents the metrics performances of the two types of portfolios (e.g., static and dynamic Markowitz) along with the results of the four versions of the models. The abbreviation 'CNN' represents the model implementing the convolutional layers. The abbreviation 'RNN' represents the model implementing the LSTM layers of the

recurrent neural networks. Moreover, for each of the last four columns of the table it is specified the version of the dataset with which the model was trained.

Table 5 – Metrics performances on portfolios

The table compare the performances of all the portfolios involved in my analysis going from the 1st January 2016 to 1st June 2017. I defined the evaluating process as follows. I define 12 time periods of 6 months. The first one is defined from the 1st January 2016 to the 1st July 2016, the second is defined from the 1st February 2016 to the 1st August 2016 and so on. The last time period is defined from the 1st December 2016 to the 1st June 2017. For each of these 12 time periods, I applied 6 different portfolio techniques. The first two columns represent the portfolios with whom I compared the performances of my model that are represented by the last 4 columns of the table. The 'Static Markowitz' portfolio is defined as a traditional tangency portfolio obtained solving a Markowitz optimization problem over the mean vector and the covariance matrix returns over the 1-year daily returns dataset preceding the period considered. To clarify, consider the first time period defined from the 1st January 2016 to 1st July 2016. I compute the tangency portfolio weights considering the data on returns from the 1st January 2015 to the 31st December 2015. I store the performance of the portfolio from the 1st January 2016 to 1st July 2016 and I apply the same procedure with incoming periods. The tangency portfolio weights over the second period (defined from the 1st February 2016 to the 1st August 2016) are obtained through the mean vector and covariance matrix of the returns of the period going from the 1st February 2015 to 31st January 2016. The Dynamic Markowitz portfolio follows the same structure of the Static Markowitz portfolio. Initially the portfolio weights are defined in the same way as for the Static Markowitz. Successively, every 15 days the portfolio weights are adjusted. The adjustments are performed using the data returns of the 60 days preceding the day of the adjustment. To clarify, considering the first period going from the 1st January 2016 to the 1st July 2016, the portfolio weights over the first 15 days of the period are computed over the returns going from the 1st January 2015 to the 31st December 2015. At the 15th January, the portfolio weights are adjusted solving the Markowitz optimization problem over the previous 60-days daily returns dataset (from the 15th November 2015 to 14th January 2016). The others four models follow the same schema described with the Dynamic Markowitz. The model inputs are the datasets composed my numerous features (e.g., standard deviation, positive sentiment on financial news and momentum) and not the mere mean vector and covariance matric of the returns used to solve the Markowitz optimization problem. Once all the 6 different portfolio management techniques have been applied over the 12 periods of time. I compute the mean relative to all the metrics performances measured: period return, annualized daily mean returns, annualized standard deviation and the annualized Sharpe ratio.

	Static Markowitz	**Dynamic Markowitz**	**CNN No Sentiment Analysis**	**RNN No Sentiment Analysis**	**CNN Sentiment Analysis**	**RNN Sentiment Analysis**
Period Return	11.43%	12.51%	15.96%	14.77%	22.24%	9.21%
Annualized Daily Mean Returns	22.15%	24.31%	30.56%	28.28%	41.08%	17.97%
Annualized Standard Deviation	14.02%	13.75%	14.89%	16.61%	15.82%	12.29%
Annualized Sharpe Ratio	1.63	1.75	2.10	1.73	2.70	1.45

The metric performances of the portfolios obtained over the 12 periods through the convolutional neural networks are averagely higher compared to the portfolios of

comparison expect from the annualized standard deviation. Moreover, it is possible to note that the metric performances of the convolutional neural networks trained over the dataset with the sentiment analysis is visibly higher with respect to the other. The higher level of annualized standard deviation can be the result of a general higher level of daily returns. It is important to recall that the objective function behind the optimization process of my model, was to maximize the Sharpe ratio. Table 6 shows the statistical tests that compares the mean values of the annualized Sharpe ratio and the period return of the two CNN models respectively compared to the two portfolios of comparison.

Table 6 – Statistical Tests over Period Return and Annualized Sharpe Ratio

The table shows the statistical tests performed over two metrics: the period return and the annualized Sharpe ratio. The values in the cells are the p-values while, the values in parenthesis are the t-statistics. I compare the result obtained through the model defined through the convolutional layers and trained over the two different versions of the dataset. The first version of the datasets contains only statistical information (e.g., standard deviation, momentum and returns). The second version of the dataset contains the statistical information defined in the first version of the dataset and the sentiment analysis performed for each of the 12 industries. I performed one-sided statistical tests. I test if the metrics performances obtained through my models are higher compared to the ones of the 'Static Markowitz' and the 'Dynamic Markowitz'. The CNN model trained over the dataset containing sentiment analysis is able to perform better with respect to both the 'Static Markowitz' and the 'Dynamic Markowitz' portfolios in terms of the two metrics performances. The p-values are lower than 0.05. Differently, the CNN model trained over the dataset without the sentiment analysis is not able to perform better than the 'Static Markowitz' and 'Dynamic Markowitz' with statistical significance. All the p-values are greater than 0.05.

	CNN No Sentiment Analysis		**CNN Sentiment Analysis**	
	Static Markowitz	**Dynamic Markowitz**	**Static Markowitz**	**Dynamic Markowitz**
Period Return	0.051 (1.770)	0.101 (1.349)	0.001 (3.920)	0.002 (3.530)
Annualized Sharpe Ratio	0.098 (1.371)	0.164 (1.019)	0.005 (3.010)	0.010 (2.669)

The statistical tests confirm the considerations relative to the higher performance obtained through the CNN model trained through the dataset with the sentiment analysis. The values defined in parenthesis are the t-statistics. It manages to succeed both the metrics performances with respect to the period return and the annualized Sharpe ratio. With respect to the CNN model version trained through the dataset without the sentiment analysis, it is possible to note that values tend to be quite low but not lower than 0.05.

Moreover, the metric performances obtained through the RNN models are not satisfactory compared to the other portfolios. Only the RNN model trained with the dataset without the sentiment analysis performs slightly better with respect to the period return and annualized mean return of the two portfolios of comparison. I perform the statistical tests that compares the period return and the annualized Sharpe ratio with the other two portfolios. The p-values are higher than 0.22 in all the cases and so, not statistically significant. Our objective is not satisfied due to a high amount of annualized standard deviation. The annualized Sharpe ratio is almost similar to the ones obtained over the static and dynamic comparison portfolios. This problem mainly relates to the training process adopted for the recurrent neural networks. Bengio, Simard and Frasconi (1994) reported the difficulty of training in a short period of time this type of neural networks. There are associated problem of training and for the same reason, the result obtained with the RNN model are extensively unstable.

Among the other problems with the limited capacity of this type of models I recognize: the large dimension of the datasets requires longer period of training time and more performant computational resources. Secondly, the portfolio weights adjustment is scheduled every 15 days and the inputs I give to the model are very recent. For this type of activity, the convolutional neural networks are more performing and generally adapt better. The recurrent neural networks are generally implemented when dealing with long-term dependencies over complex financial time-series.

Singh et al. (2021) report the performance obtained using the layers I implemented in my model (i.e., long short-term memory layers) and another category of recurrent layers: gated recurrent unit (GRU). In the paper, they show that the performances obtained through the second type of layers with respect to the time required for the training process is better compared to the long short-term memory. The reason refers essentially to the simplified structure of the GRU layers with respect to the LSTM.

B. Second group of experiments (From 1st January 2017 to 1st June 2018)

Table 7 follows the same structure provided by the Table 5. It presents the metrics performances of six portfolio: the first columns represent the metrics obtained through the portfolio of comparison and the last four columns represents four different versions of my model. Differently, from the first group of experiment, the only difference relates to the date of the beginning for this evaluation: 1st January 2017.

At a first glance, it is possible to confirm again the superiority performances of the CNN models with respect to the static and dynamic versions of the portfolios of comparison and also the RNN models. Differently from the analysis of the first group of experiments, the metrics performances obtained through CNN model trained over the dataset with the sentiment analysis underperforms the CNN model trained over the dataset without the sentiment analysis.

Table 7 – Metrics performances on portfolios

The table compare the performances of all the portfolios involved in my analysis going from the 1st January 2017 to 1st June 2018. I defined the evaluating process as follows. I define 12 time periods of 6 months. The first one is defined from the 1st January 2017 to the 1st July 2017, the second is defined from the 1st February 2017 to the 1st August 2017 and so on. The last time period is defined from the 1st December 2017 to the 1st June 2018. For each of these 12 time periods, I applied 6 different portfolio techniques. The first two columns represent the portfolios with whom I compared the performances of my model that are represented by the last 4 columns of the table. The 'Static Markowitz' portfolio is defined as a traditional tangency portfolio obtained solving a Markowitz optimization problem over the mean vector and the covariance matrix returns over the 1-year daily returns dataset preceding the period considered. To clarify, consider the first time period defined from the 1st January 2017 to 1st July 2017. I compute the tangency portfolio weights considering the data on returns from the 1st January 2016 to the 31st December 2016. I store the performance of the portfolio from the 1st January 2017 to 1st July 2017 and I apply the same procedure with incoming periods. The tangency portfolio weights over the second period (defined from the 1st February 2017 to the 1st August 2017) are obtained through the mean vector and covariance matrix of the returns of the period going from the 1st February 2016 to 31st January 2017. The Dynamic Markowitz portfolio follows the same structure of the Static Markowitz portfolio. Initially the portfolio weights are defined in the same way as for the Static Markowitz. Successively, every 15 days the portfolio weights are adjusted. The adjustments are performed using the data returns of the 60 days preceding the day of the adjustment. To clarify, considering the first period going from the 1st January 2017 to the 1st July 2017, the portfolio weights over the first 15 days of the period are computed over the returns going from the 1st January 2016 to the 31st December 2016. At the 15th January, the portfolio weights are adjusted solving the Markowitz optimization problem over the previous 60-days daily returns dataset (from the 15th November 2016 to 14th January 2017). The others four models follow the same schema described with the Dynamic Markowitz. The model inputs are the datasets composed my numerous features (e.g., standard deviation, positive sentiment on financial news and momentum) and not the mere mean vector and covariance matric of the returns used to solve the Markowitz optimization problem. Once all the 6 different portfolio management techniques

have been applied over the 12 periods of time. I compute the mean relative to all the metrics performances measured: period return, annualized daily mean returns, annualized standard deviation and the annualized Sharpe ratio.

	Static Markowitz	Dynamic Markowitz	CNN No Sentiment Analysis	RNN No Sentiment Analysis	CNN Sentiment Analysis	RNN Sentiment Analysis
Period Return	3.64%	1.33%	9.53%	4.58%	7.91%	5.58%
Annualized Daily Mean Returns	7.48%	3.46%	18.97%	9.38%	15.76%	11.27%
Annualized Standard Deviation	13.33%	14.82%	12.96%	13.44%	12.69%	10.79%
Annualized Sharpe Ratio	0.80	0.22	1.50	0.79	1.33	1.04

The RNN models manage to slightly overperform the two portfolios of comparison. I performed statistical tests to check if the mean values over the 12 periods considered of the period return and the annualized Sharpe ratio metrics of the CNN models are higher with statistical significance with respect to the static and dynamic portfolio comparison. Table 8 shows a different situation compared to the first group of experiments, in particular: with respect the dynamic portfolio of comparison, both the version of the CNN models successfully overperform it. The statistical evidence of these greater capacity is more evident in the CNN model trained with dataset not including the sentiment analysis. The CNN model trained with the dataset including the sentiment analysis present p-values lower than 0.05 which confirm a better perform with respect to the portfolios of comparison.

Table 8 – Statistical Tests over Period Return and Annualized Sharpe Ratio

The table shows the statistical tests performed over two metrics: the period return and the annualized Sharpe ratio. The values in the cells are the p-values while, the values in parenthesis are the t-statistics. I compare the result obtained through the model defined through the convolutional layers and trained over the two different versions of the dataset. The first version of the datasets contains only statistical information (e.g., standard deviation, momentum and returns). The second version of the dataset contains the statistical information defined in the first version of the dataset and the sentiment analysis performed for each of the 12 industries. I performed one-sided statistical tests. I test if the metrics performances obtained through my models are higher compared to the ones of the 'Static Markowitz' and the 'Dynamic Markowitz'. The two versions of the CNN model are able to perform better with respect to the 'Dynamic Markowitz' portfolio in terms of the two metrics performances. The p-values are lower than 0.05. Differently, with respect to the 'Static Markowitz' portfolio, the CNN models are not able to significantly show better results. Expect for the period return metric where the p-value Is lower than 0.05 (0.025).

	CNN No Sentiment Analysis		CNN Sentiment Analysis	
	Static Markowitz	**Dynamic Markowitz**	**Static Markowitz**	**Dynamic Markowitz**
Period Return	0.025 (2.177)	0.005 (3.032)	0.081 (1.494)	0.020 (2.302)
Annualized Sharpe Ratio	0.058 (1.699)	0.005 (3.097)	0.122 (1.224)	0.013 (2.357)

In this case, with respect to the static portfolio of comparison, the results slightly differ in metrics performances. It is important to note that the dynamic version over the periods considered underperformed with respect to all the other portfolios. Adjusting portfolio weights every 15 days through the weights of the tangency portfolio elaborated with a Markowitz optimization over the 60 days preceding the day of adjustment seems not be the performant choice (i.e., dynamic Markowitz portfolio). The CNN model is able to offer more performant solutions with respect to the period return and annualized Sharpe ratio metrics through a dynamic environment.

The annualized Sharpe ratio computed through the CNN model trained over the dataset without sentiment analysis show an unsatisfactory result. Although, as the Table 5 shows, the value obtained is higher compared to the traditional tangency portfolio obtained through the Markowitz optimization using the preceding 1-year returns data (i.e., static Markowitz portfolio), the p-value demonstrates a lack of statistical significance: 0.058 ($p > 0.05$).

It is important to note that the period of time considered it is characterized by high volatility, differently from the results obtained over the period of time defined of the first group of experiments. The metrics performances of the models are highly dependent on the uncertainty of the model. The training dataset used to train the different models contains data going from the 1st January 2010 up to the 31st December 2015. Statistical properties changes over the time. To perform better results, the training dataset could contain information, in order to train the model with more consistency. This would have required too high computational cost resources.

Further consideration

There are five main issues affecting the implementation of this model: ability to properly encode the market state, reward function design, criteria for portfolio weights adjustment, real-world financial market considerations and neural network layer composition. The first problem relates to the obstacle of identifying features that describes the financial markets status. Ideally, the model must be able to properly encode all the information available up to a certain point in time to perform the best possible action. In my thesis work, the action corresponds to the portfolio weights choice. If the information is, in certain circumstances, unnecessary or is characterized by specific statistical properties could negatively affect the model performances. Cont (2001) presented different statistical properties of financial assets (e.g., distributional properties, conditional heavy tails and volume/volatility correlation). Yashaswi (2021) reports that two important elements are the main driver of this issue: volatility, non-stationarity and noise affecting the financial time series. In his paper proposes three different approaches to minimize the quantity of noise, implementing techniques defined in the information theory field. Solovyeva (2018) proposed a technique based on recurrent neural networks to filter out noise from distorted images. This suggestion could be likewise applied to financial time series to obtain more cleaned signal. The mentioned techniques should improve decision-making process of the model.

The second problem relates the difficulty in defining a proper reward function. In this thesis work, I aimed to maximize the Sharpe ratio over a specific period of time. The associated reward function can be defined in different ways. The choice cannot be unique and it is highly dependent from the type of features included in the training dataset. Linear combinations of the different metrics such as: standard deviation, maximum dropdown and return rate, could potentially bring to more performant results due to the complexity of the problem definition.

The third problem refers to the choice of criteria for portfolio weights adjustment. In this thesis work, I set a basis of 15 days for portfolio weights adjustments. In future

works, the choice could be dependent over other metrics such as: achievement of a specific rate of return, specific pattern followed by returns in the financial asset or sentiment on financial news. The implementation of these approaches could lead further difficulty due to the complexity of the problem. In this way, it is possible to avoid possible substantial loss of money and consequently maximizing the objective function in a more appropriate way. The choice to define a specific number of days between the portfolio weights adjustment is recognized as a strong constrained.

The fourth problem relates to consideration of aspects such as: transaction costs, asset allocation and continuous-time environment. The transactions costs are not considered in this work. Most of the papers related on portfolio management implementing machine learning techniques do not consider transaction costs due to their complex nature. There exist linear and non-linear, explicit and implicit transactions costs that are difficult to include in the experiments, but they could negatively affect the performances (e.g., bid-ask spread, market impact cost and broker fees). The environment designed for this work define portfolio weights whose components belong to the multi-dimensional real number space. In the real-world, only some brokers offer the possibility to buy fractional parts of shares but these are limited up to certain decimal points (e.g., Robinhood and Trading212). Lastly, we assume it is possible buy and sell stocks without any eventual liquidity problem.

The fifth problem relates to the incapacity of the recurrent neural networks (RNN) to provide performant result. As reported above, this issue essentially relates to the nature of data given to the model. LeCun et al. (1998) report the capability of convolutional neural networks (CNN) to discover pattern and local dependencies on a given input. With reference to this work, the CNN models are able to analyze short-term dependencies over data. Differently, the RNN models offer a framework specialized in long-term dependencies over the time series. This specific property requires a significant amount of training time. Future research works could focus on analyzing the metrics performances obtained with the same type of recurrent neural

network but by setting 30 or more days as the requirement between a portfolio weights adjustment and another. Differently from this thesis work were there were only 15 days of difference.

Conclusion

I built and evaluated different versions of a deep reinforcement learning model. Various academics over the last few decades offered different solutions to bring dynamicity to the portfolio management activity. The financial markets evolved and nowadays, many financial agents operate in the financial market with the scope of exploiting inefficiencies and eventual market crisis. I suggested different approaches that could lead to valuable portfolio performances in a dynamic environment. It is important to highlight that there are still some issues relative to the techniques I showed. From a real-word perspective, the implementation of these type of models could lead to unintended results. For the same reason, the structure of the models and the dataset must be trained and evaluated over other period of times and possibly allowing more time for the training phase.

It is also important to mention the existence of many new types of layers and model architecture. These could potentially address the dynamic portfolio optimization problem in more consistent and efficient ways. Among the new available tools: Transformers layers, Vaswani et al. (2017) introduced a new layer architecture to strongly outperform the recurrent layers; Large Language Models, thanks to the computational resources, it is now possible to train model composed by billions of parameters (e.g., ChatGPT 4).

References

Almahdi Saud, and Steve Y. Yang, "An Adaptive Portfolio Trading System: A Risk-Return Portfolio Optimization using Recurrent Reinforcement Learning with Expected Maximum Drawdown." *Expert Systems with Applications* 87 (2017): 267-279.

Araci Dogu, "FinBERT: Financial Sentiment Analysis with Pre-Trained Language Models." Working paper, University of Amsterdam (2019).

Awad Alamir Labib, Saleh Mesbah Elkaffas, and Mohammed Waleed Fakhr, "Stock Market Prediction using Deep Reinforcement Learning." *Applied System Innovation* 6, no. 6 (2023): 106.

Bao Wenhang, and Liu Xiao-yang, "Multi-Agent Deep Reinforcement Learning for Liquidation Strategy Analysis." Working paper, Columbia University (2019).

Bekaert, Geert, Marie Hoerova, and Marco Lo Duca, *"Risk, Uncertainty and Monetary Policy."* NBER Working Paper, no. 16397 (2010).

Beltratti Andrea, *Investimenti Finanziari*. Milano: EGEA, 2022.

Bengio Y., P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult." *IEEE Transactions on Neural Networks* 5, no. 2 (1994): 157-166.

Brock William, Josef Lakinishok, and Blake LeBaron, "Simple Technical Trading Rules and the Stochastic Properties of Stock Returns." *Journal of Finance (New York)* 47, no. 5 (1992): 1731-1764.

Cheng Li-Chen, Yu-Hsiang Huang, and Mu-En Wu, "Applied Attention-Based LSTM Neural Networks in Stock Prediction." IEEE, 2018.

Chong Eunsuk, Chulwoo Han, and Frank C. Park, "Deep Learning Networks for Stock Market Analysis and Prediction: Methodology, Data Representations, and Case Studies." *Expert Systems with Applications* 83, (2017): 187-205.

Cont R., "Empirical Properties of Asset Returns: Stylized Facts and Statistical Issues." *Quantitative Finance* 1, no. 2 (2001): 223-236.

Rumelhart David E., Geoffrey E. Hinton, and Ronald J. Williams, "Learning Representations by Back-Propagating Errors." 323, no. 6088 (1986): 533-536.

Fama Eugene F., and Kenneth R. French, "Multifactor Explanations of Asset Pricing Anomalies." *Journal of Finance (New York)* 51, no. 1 (1996): 55-84.

Fazlija Bledar, and Pedro Harder, "Using Financial News Sentiment for Stock Price Direction Prediction." *Mathematics (Basel)* 10, no. 13 (2022): 2156.

Fischer Thomas, and Christopher Krauss, "Deep Learning with Long Short-Term Memory Networks for Financial Market Predictions." *European Journal of Operational Research* 270, no. 2 (2018): 654-669.

Hamilton James D., *Time Series Analysis / James D. Hamilton*. Princeton: Princeton University Press, 1994.

Hedayati Moghaddam Amin, Moein Hedayati Moghaddam, and Morteza Esfandyari, "Stock Market Index Prediction using Artificial Neural Network." *Journal of Economics, Finance and Administrative Science* 21, no. 41 (2016): 89-93.

Hochreiter Sepp, and Jürgen Schmidhuber, "Long Short-Term Memory." 9, no. 8 (1997): 1735-1780.

Hoseinzade Ehsan, and Saman Haratizadeh, "CNNpred: CNN-Based Stock Market Prediction using a Diverse Set of Variables." *Expert Systems with Applications* 129, (2019): 273-285.

Ioffe Sergey, and Christian Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." *Working paper, Google Inc.* (2015).

Jegadeesh Narasimhan, and Sheridan Titman, "Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency." *Journal of Finance (New York)* 48, no. 1 (1993): 65-91.

Joyeux Roselyne, "An Introduction to Long-Memory Time Series Models and Fractional Differencing." Cambridge University Press (2001): 321-337.

LeCun Yann, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition." *Proceedings of the IEEE* 86, no. 11 (1998): 2278-2324.

LeCun Yann, Yoshua Bengio, and Geoffrey Hinton, "Deep Learning." *Nature (London)* 521, no. 7553 (2015): 436-444.

Lin Min, Qiang Chen, and Shuicheng Yan, "Network in Network." *Working paper, National University of Singapore* (2014).

Maitra Sarit, Vivek Mishra, Srashti Dwivedi, Sukanya Kundu, and Goutam Kumar Kundu, "Time-Series Forecasting: Unleashing Long-Term Dependencies with Fractionally Differenced Data." *Working paper, Alliance University* (2023).

Moghar Adil, and Mhamed Hamiche, "Stock Market Prediction using LSTM Recurrent Neural Network." *Procedia Computer Science* 170, (2020): 1168-1173.

Perold André F., and William F. Sharpe, "Dynamic Strategies for Asset Allocation." *Financial Analysts Journal* 44, no. 1 (1988): 16-27.

Rouf Nusrat, Majid Bashir Malik, Tasleem Arif, Sparsh Sharma, Saurabh Singh, Satyabrata Aich, and Hee Cheol Kim, "Stock Market Prediction using Machine Learning Techniques: A Decade Survey on Methodologies, Recent Developments, and Future Directions." *Electronics (Basel)* 10, no. 21 (2021): 2717.

Shi Zhan, and Chongjun Fan, "Short Text Sentiment Classification using Bayesian and Deep Neural Networks." *Electronics (Basel)* 12, no. 7 (2023): 1589.

Singh Mayank, Vipin Tyagi, P. K. Gupta, Jan Flusser, Tuncer Ören, and V. R. Sonawane, "Prediction of Stock Price for Indian Stock Market: A Comparative Study using LSTM and GRU." In . Vol. 1441, 292-302. Switzerland: Springer International Publishing AG, 2021.

Solovyeva, E., "Recurrent Neural Networks as Approximators of Non-Linear Filters Operators." *Journal of Physics: Conference Series* 1141, no. 1 (2018): 12115.

Tsai Yun-Cheng, Chun-Chieh Wang, Fu-Min Szu, and Kuan-Jen Wang, "Deep Reinforcement Learning for Foreign Exchange Trading." In . Vol. 12144, 387-392. Cham: Springer International Publishing, 2020.

Tsay Ruey S., *Analysis of Financial Time Series, Third Edition*. 3rd ed. Hoboken, NJ: Wiley, 2010.

Ubal Cristian, Gustavo Di-Giorgi, Javier Contreras-Reyes, and Rodrigo Salas, "Predicting the Long-Term Dependencies in Time Series using Recurrent Artificial Neural Networks." *Machine Learning and Knowledge Extraction* 5, no. 4 (2023): 1340-1358.

Vaswani Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention Is All You Need." *Working paper, Google Brain* (2017).

Xiao-Yang Liu, Hongyang Yang, Jiechao Gao, and Christina Dan Wang, "FinRL: Deep Reinforcement Learning Framework to Automate Trading in Quantitative Finance." *Working paper, Columbia University* (2021).

Yashaswi Kumar, "Deep Reinforcement Learning for Portfolio Optimization using Latent Feature State Space (LFSS) Module." *Working paper, Indian Institute of Technology* (2021).

Zhang Xuan, Xun Liang, Aakas Zhiyuli, Shusen Zhang, Rui Xu, and Bo Wu, "AT-LSTM: An Attention-Based LSTM Model for Financial Time Series Prediction." *IOP Conference Series: Materials Science and Engineering* 569, no. 5 (2019): 52037.

Zhang Zihao, Stefan Zohren, and Stephen Roberts, "Deep Learning for Portfolio Optimization." *Journal of Financial Data Science* 2, no. 4 (2020): 8-20.