NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

Machine Learning Methods for Markowitz Portfolio Optimization

Nikitas N. Sakkas

SUPERVISOR: **Yannis Panagakis,** Associate Professor

ATHENS

NOVEMBER 2021

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Μέθοδοι Μηχανικής Μάθησης για την Βελτιστοποίηση Χαρτοφυλακίου Markowitz

Νικήτας Ν. Σακκάς

ΕΠΙΒΛΕΠΩΝ: **Γιάννης Παναγάκης,** Αναπληρωτής Καθηγητής

ΑΘΗΝΑ

ΝΟΕΜΒΡΙΟΣ 2021

BSc THESIS

Machine Learning Methods for Markowitz Portfolio Optimization

Nikitas N. Sakkas
S.N.: 1115201400176

SUPERVISOR: **Yannis Panagakis,** Associate Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Μέθοδοι Μηχανικής Μάθησης για την Βελτιστοποίηση Χαρτοφυλακίου Markowitz

Νικήτας Ν. Σακκάς
Α.Μ.: 1115201400176

ΕΠΙΒΛΕΠΩΝ: **Γιάννης Παναγάκης,** Αναπληρωτής Καθηγητής

ABSTRACT

In this thesis, we consider the problem of Markowitz Portfolio Optimization. It is defined as attempting to minimize the variance of a diversified investment's returns. We use several conventional Machine Learning techniques to solve it, namely CVXpy, CVXpy-layers, Proximal and Projected Gradient Descent. We also propose a Deep Learning approach, which uses an LSTM unit. As investment units to train our models, we use the historic returns of 48 industry sector portfolios from 2019 to 2021(FF48 daily returns [6]). Four of our models including our Deep Learning approach manage to surpass the performance of the equally weighted portfolio which is considered a tough benchmark in this problem. Finally, we propose modifications for further improvements.

SUBJECT AREA: Machine Learning

KEYWORDS: markowitz portfolio, optimization, cvxpy, projected gradient descent, lstm

ΠΕΡΙΛΗΨΗ

Στην παρακάτω πτυχιακή εργασία, εξετάζουμε το πρόβλημα της βελτιστοποίησης Χαρτοφυλακίου Markowitz, το οποίο ορίζεται ως η απόπειρα ελαχιστοποιήσης της διακύμανσης των επιστροφών μιας διαφοροποιημένης επένδυσης. Για να το επιλύσουμε, χρησιμοποιούμε πολλές συμβατικές τεχνικές Μηχανικής Μάθησης, συγκεκριμένα CVXpy, CVXpy-layers, Proximal και Projected Gradient Descent. Επίσης, προτείνουμε και μια προσέγγιση Βαθιάς Μάθησης (Deep Learning), η οποία βασίζεται σε Δικτύο Μακράς Βραχύχρονης Μνήμης (LSTM). Ως επενδυτικές μονάδες για την εκπαίδευση των μοντέλων μας, χρησιμοποιούμε τις ιστορικές επενδυτικές αποδόσεις 48 χαρτοφυλακίων διαφορετικών βιομηχανικών κλάδων για τα έτη 2019-2021(FF48 daily returns [6]). Τέσσερα από τα μοντέλα μας, συμπεριλαμβανομένης και της υλοποίησης Deep Learning, καταφέρνουν να ξεπεράσουν την απόδοση του εξίσου σταθμισμένου χαρτοφυλακίου, κάτι που θεωρείται ιδιαίτερα δύσκολο σε αυτό το πρόβλημα. Τέλος, προτείνουμε τροποποιήσεις για περαιτέρω βελτιώσεις.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: χαρτοφυλάκιο markowitz, βελτιστοποίηση, cvxpy, projected gradient descent, lstm

To my family.

ACKNOWLEDGMENTS

I would like to thank my supervisor Dr Yannis Panagakis for his valuable guidance and assistance during this thesis.

CONTENTS

LIST OF FIGURES

PREFACE

This document was completed in Athens, 2021. It has been written to fulfill the graduation requirements of the Department of Informatics and Telecommunications undergraduate program. I was engaged in researching and writing it from December 2019 to November 2021. It is mainly addressed to Machine Learning and Stock Investing enthusiasts. I hope you will find its contents interesting and useful.

Nikitas Sakkas

1. INTRODUCTION

1.1 Historical Background

Markowitz's portfolio theory is one of the most important theoretical developments in finance. In 1952, Henry Markowitz applied mathematics to the analysis of the stock market. During his thesis, he realized that the current methods only took into account the expected return and not the risk of an investment. This insight led to the development of his seminal theory of portfolio allocation under uncertainty, later published by the Journal of Finance.

" The investor does (or should) consider expected return a desirable thing and variance (risk) of return an undesirable thing."

1.2 Risk and Diversification

In economics, risk implies future uncertainty about deviation from expected earnings. Risk measures the uncertainty that an investor is willing to take to realize a gain from an investment [3]. There are two types of risk, systematic and unsystematic [4].

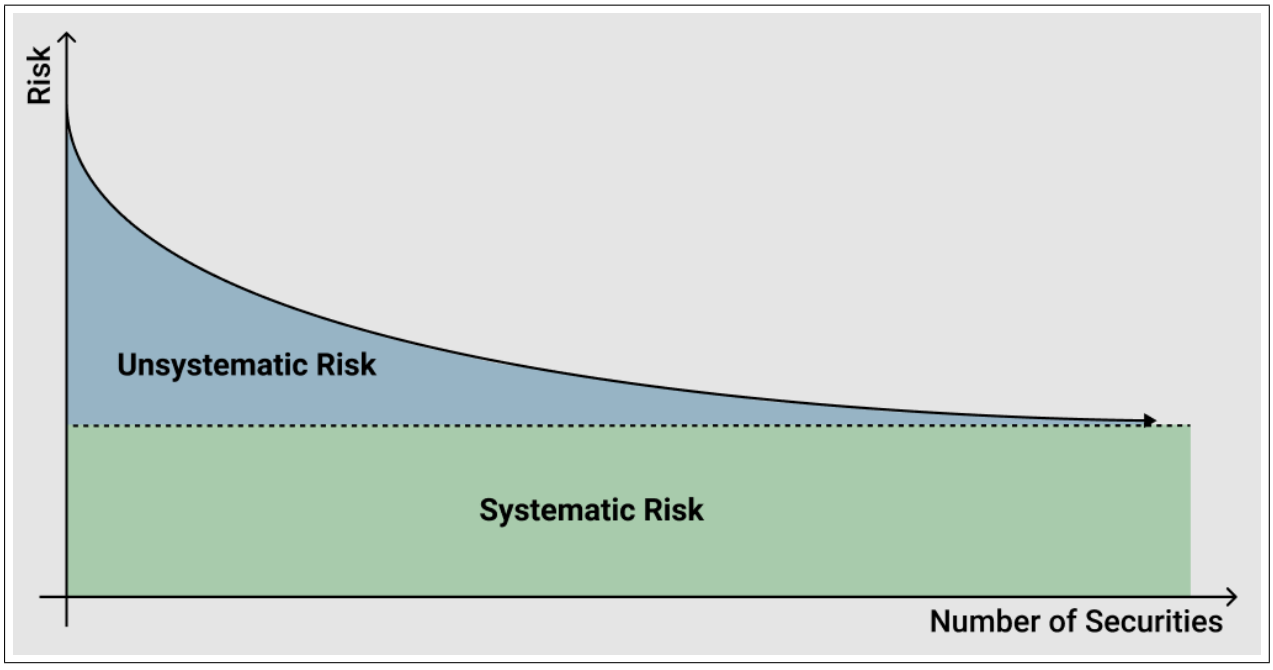


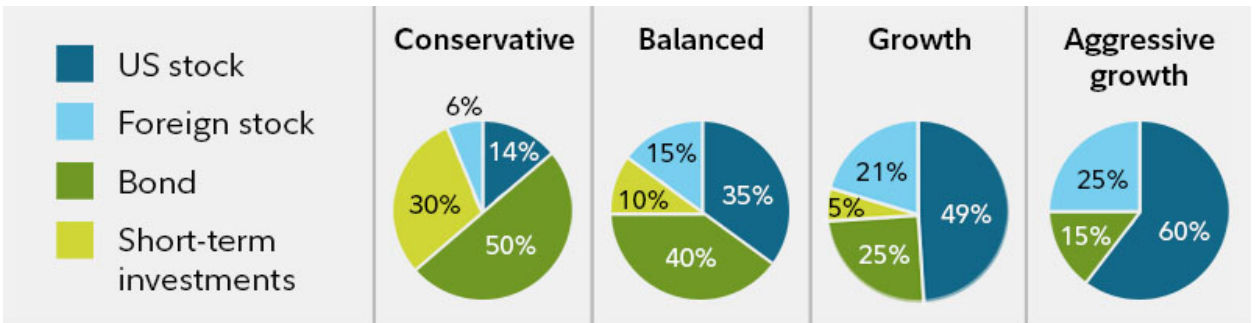
Figure 1: Diversifying an investment minimizes unsystematic risk

1. **Systematic Risk** does not have a specific definition but is an inherent risk existing in the stock market. These risks are applicable to all the sectors but can be controlled.
2. **Unsystematic Risk** is an industry or firm-specific threat in each kind of investment.

The unsystematic risk can be minimized using diversification. The more the uncorrelated assets in an investment portfolio, the more the unsystematic risk shrinks. However, minimizing the systematic risk is a significantly tougher obstacle.

1.3 Background

Hence, over the last decades multiple investment models have been created [1] [16] [15] [28] [5] . Their goal is to pay attention on the systematic risk of various portfolios and not just focus solely on their returns. However, to our knowledge, when it comes to Markowitz Portfolio Optimization, most of them use conventional Machine Learning and optimization techniques even though Deep Learning has surprisingly improved accuracy in various other domains [30]. Specifically, Recurrent Neural Networks [21] are becoming more and more popular when it comes to time series data such as speech recognition, music composition, or in our case predictions in the stock market.



	Annual return %			
Average annual return	5.96%	7.98%	9.00%	9.70%
Worst 12-month return	–17.67%	–40.64%	–52.92%	–60.78%
Best 12-month return	31.06%	76.57%	109.55%	136.07%
Worst 20-year return (annualized)	2.92%	3.43%	3.10%	2.66%
Best 20-year return (annualized)	10.98%	13.84%	15.34%	16.49%
Historical volatility	4.49%	9.56%	13.07%	15.76%

Figure 2: Different Portfolios based on the same Assets. From left to right, more attention is paid to returns, rather than systematic risk. [36]

1.4 Motivation

For this reason, we attempted to build an LSTM model to predict the returns of various industries, and then use an optimizer to create optimal Markowitz Portfolios. We also modified multiple conventional models to solve this problem for better understanding and comparisons.

1.5 Related Work

Over the last decades, multiple Machine Learning models have been developed in finance and banking, including units used for stock prediction and portfolio construction. In the following subsections we review implementations related to our problem. We split them into two categories, those who use conventional Machine Learning and those who apply

Deep Learning.

1.5.1 Conventional Machine Learning

Conventional Machine Learning techniques have been applied on this problem for decades. We selected the following work as guidelines of our conventional methodologies:

- In 2007, Victor DeMiguel, Lorenzo Garlappi, Francisco J. Nogales and Raman Uppal provided a general framework for creating norm-constrained portfolios that perform well out-of-sample even in the presence of estimation error [15]. They achieved this by solving the traditional minimum-variance problem (see Chapter 2), but subject to the additional constraint that the p-norm of the portfolio-weight vector be smaller than a given threshold.
- In 2008, Brodie, Daubechies, De Mol, Giannone and Loris followed up on this work [12]. However, their goal wasn't only regularization through an added norm, but stability and sparsity of identified portfolios, which was achieved by l_1 penalization. This was consistent with the observation made by Jagannathan and Ma (2003) that a restriction to non-negative-weights-only can have a regularizing effect on Markowitz's portfolio construction [16]. They then used the **L**east **A**ngle **R**egression algorithm (**LARS**) to calculate the portfolios, which yielded great results in terms of stability, sparsity and returns. In one of our approaches, we used the same objective function but solved it using a different algorithm, namely **Proximal Gradient Descent** (see 3.3.2).
- It is important to note that the equally weighted portfolio is considered a high benchmark in this problem. It is comprised by the same amount of investment in each asset, and it achieves low variance due to its high diversification. For this reason, Daniel Mckenzie (2017) proposed to create a sparse portfolio that replicates the performance of the equally weighted portfolio [5]. He based his methodology on Anastasios Kyrillidis, Stephen Becker, Volkan Cevher and Christoph Koch, [28] using **sparse projections onto the simplex** [17]. We included this approach in our models, since the equally weighted portfolio is very effective in this problem, and this methodology is quite different from the rest (see 3.3.3). We also solved his minimization problem using CVXpy-layers (see 3.3.4).

1.5.2 Deep Learning

In recent years, Deep Learning has been gaining ground when it comes to predictions and optimization in finance. For our problem, we examined the following approaches:

- Zhang Z., Zohren S. and Roberts S. (2005) adopted Deep Learning models to directly optimize the portfolio Sharpe Ratio [20]. Instead of selecting individual assets, they traded **Exchange-Traded Funds** (ETFs) [25] of market indices to form a port-

folio. Indices of different asset classes show robust correlations and trading them substantially reduces the spectrum of available assets to choose from. Our own asset selection was based on this approach along with [1], which tackled correlation by selecting different Industry Portfolios as assets [6]. Regarding their Deep Learning model, they based their work on [19] and [29].

- Hieu K. Cao, Han K. Cao and Binh T. Nguyen (2020) used Deep Learning, namely **Recurrent** and **Convolutional Neural Networks** to tackle portfolio optimization that outperformed popular indexes [24]. However, in their approach they attempted to maximize the Sharpe Ratio of the portfolio (same as [20]), which differs from the Markowitz portfolio problem. In our Deep Learning approach, we also use a Recurrent Neural Network (**LSTM**), but to create a Markowitz portfolio instead (see 3.4).
- There have also been many implementations of Stock Predictions using Deep Learning [33] [32] [27]. We base the first part of our LSTM models on their methodology, and then use their predictions to form a Markowitz Portfolio.

1.6 Objective

The objective of this thesis is split in two parts. The first goal is to attempt to modify known optimization methodologies when it comes to Markowitz Portfolios. We propose several conventional Machine Learning methods and models to achieve that goal. Secondly, we consider optimal Markowitz Portfolio creation using Deep Learning, something that (to our knowledge) hasn't been attempted yet.

Briefly, to build our conventional models we used CVXpy, LASSO regression, equally weighted portfolio replication and CVXpy layers. Our Deep Learning Model consists of an LSTM unit which serves as a predictor, and an optimizer which uses these predictions to calculate optimal Markowitz Portfolios.

To implement our methods, we used a set of portfolios created by Fama and French [6] as our assets, namely 48 Industry Portfolios (FF48) daily returns. The data refer to a time period which is approximately from 07/2019 to 06/2021.

1.7 Outline

The outline of our thesis is as follows:

In the second Chapter, we go over notation, define the Markowitz portfolio problem and then form the objective function that quantifies it.

In Chapter 3 we firstly go over the data we will use in our implementation. Then, we describe various conventional Machine Learning algorithms that utilize that data to form optimal Markowitz Portfolios. Finally, we analyze our Deep Learning unit which predicts the behavior of our assets and then applies an optimizer on the predictions to create a Markowitz Portfolio.

In Chapter 4, we firstly evaluate the accuracy of our Deep Learning's unit predictions and

then compare the performance of all our models.

In Chapter 5, our conclusions are drawn and we suggest modifications to further new research.

2. MARKOWITZ PORTFOLIO: PROBLEM DEFINITION

In this Chapter we go over notation regarding investment portfolios, and then define and quantify the Markowitz Portfolio optimization problem.

2.1 Notation

2.1.1 Investment Portfolio Basics

We'll begin with basic notation. Consider there are $\boldsymbol{N}$ assets available to an investor. The price of the i-th assets at given time $\boldsymbol{t}$ is $\boldsymbol{p_i(t)}$. The return of an investment in a stock is $\boldsymbol{r_i = \frac{p_i(t) - p_i(t-1)}{p_i(t-1)}}$. The vector of returns at given time t is $\boldsymbol{r_t = (r_{1t}, r_{2t}, r_{3t}, ..., r_{Nt})}$.

We assume that the returns are stationary [1] and define the vector of expected returns $\boldsymbol{\mu = E[r_t]}$. A portfolio is a vector of $\boldsymbol{N}$ weights of investment $\boldsymbol{w = (w_0, w_2, ..., w_N)^\top}$. For simplicity, we assume that we have 1 unit of investment that must be fully used and therefore:

$$\sum_{i=0}^{N} w_i = 1$$

The **return** of a portfolio w at time t is given by:

$$\sum_{i=0}^{N} w_i r_{it} = w^\top r_t$$

The **expected return** of a portfolio w at time t is defined as:

$$w^\top \mu$$

2.1.2 Calculating the Variance

If $\bar{x} = \mathbb{E}[x_t] = w^\top \mu$, then the **variance** of a portfolio w at time t is:

$$\begin{aligned}
\mathbb{E}[(x_t - \bar{x})^2] &= \mathbb{E}[(w^\top r_t - w^\top \mu)^2] \\
&= \mathbb{E}[(w^\top (r_t - \mu))^2] \\
&= \mathbb{E}[(w^\top (r_t - \mu))(w^\top (r_t - \mu))] \\
&= \mathbb{E}[w^\top (r_t - \mu)(r_t - \mu)^\top w] \\
&= w^\top \, \mathbb{E}[(r_t - \mu)(r_t - \mu)^\top] w \\
&= \boldsymbol{w^\top S w}
\end{aligned} \tag{1}$$

$\boldsymbol{S}$ is the **Covariance Matrix** $(S = E[r_t r_t^\top] - \mu\mu^\top)$.

$$\mathrm{K_{XX}} = \begin{bmatrix} \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_n - \mathrm{E}[X_n])] \\ \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_n - \mathrm{E}[X_n])] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n - \mathrm{E}[X_n])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_n - \mathrm{E}[X_n])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_n - \mathrm{E}[X_n])(X_n - \mathrm{E}[X_n])] \end{bmatrix}$$

Figure 3: Covariance Matrix [37]

2.2 Focusing on the Markowitz Portfolio

Generally, there are 3 ways to optimize a portfolio [31]:

1. **Maximize the expected return**:

   $w_{opt} = \arg\max \quad w^\top \mu$

2. **Minimize the variance**:

   $w_{opt} = \arg\min \quad w^\top S w$

3. **Trade-off (both of the above)** [2]:

   $w_{opt} = \arg\min \quad w^\top S w - \lambda w^\top \mu$, for $\lambda > 0$

In a Markowitz (1952) portfolio optimization by Harry M. Markowitz, the goal is to minimize the variance of a portfolio, given an expected return $\boldsymbol{\rho}$. This way, we are effectively minimizing the risk of an investment.

2.3 Forming the Objective Function

Since we are trying to minimize the variance of the portfolio, using (1) the objective function is given by:

$$\begin{aligned} w = \arg\min_{w} \quad & w^\top S w \\ \text{s.t.} \quad & w^\top \mu = \rho \\ & \sum_{i=1}^{N} w_i = 1 \end{aligned} \tag{2}$$

Given that $S = E[r_t r_t^\top] - \mu\mu^\top$, the objective function becomes:

$$\begin{aligned} w = \arg\min_{w} \quad & E[|w^\top r_t - \rho|^2] \\ \text{s.t.} \quad & w^\top \mu = \rho \\ & \sum_{i=1}^{N} w_i = 1 \end{aligned} \tag{3}$$

We can use historic data in order to approximate the expected return $\mu \approx \hat{\mu} = \frac{1}{T}\sum_{t=0}^{T} r_t$. If $\boldsymbol{R}$ is the $\boldsymbol{T} \times \boldsymbol{N}$ matrix (e.g., R_{23} refers to stock 3 at time t_2), then the objective function becomes:

$$\begin{aligned} \hat{w} = \arg\min_{w} \quad & \frac{1}{T} E[|Rw - \rho \mathbf{1}_T|_2^2] \\ \text{s.t.} \quad & w^\top \mu = \rho \\ & \sum_{i=1}^{N} w_i = 1 \end{aligned} \tag{4}$$

which is a **least squares** minimization problem.

In Chapter 3, our models attempt to solve this problem or modifications of it.

3. DATA AND MODELS

In this Chapter, we firstly present the data we used to train and test our models. Secondly, we present the series of optimizations that our models will execute in order to evaluate them. Then, we analyze the Conventional Machine Learning Models we developed and the optimization problem variations they solve. Finally, we demonstrate our final model which uses Deep Learning to predict investment returns and then applies an optimizer to create a Markowitz Portfolio.

3.1 Data

For our data set, we used daily data from the Fama-French 48 set [6]. This data contains stock price changes of 48 industries between the years 1926-2021. However, due to the great size of the data, only the latest years of it are used in our models. Also, the latest years do not have any missing data, so there is no need for replacements or interpolation.

Agric	Food	Soda	Beer	Smoke	Toys	Fun	Books	Hshld	...
0.00	0.62	0.43	0.54	0.18	0.78	0.84	-0.01	0.58	...
0.76	1.11	0.27	1.29	0.16	2.05	1.43	1.59	1.32	...
0.31	1.23	0.50	2.02	1.59	1.40	0.83	0.79	1.03	...
-0.04	-0.59	-0.41	0.11	-0.02	-0.43	-1.51	-0.04	-0.53	...
0.46	-0.89	-0.79	-1.19	-1.09	-1.66	-1.68	-0.55	-1.46	...
...	...	...	...	...	...	...	...	...	...

Figure 4: Fama-French 48 data set sample

3.2 Series of Optimizations

Rather than creating a single portfolio with each optimization model and then comparing the results, we implemented a series of optimizations than our models will execute. In total, we used the final 600 days of our dataset. Starting from the oldest day, the models will use 60 ”past” days to optimize the portfolio which describes the investment for the following 12 ”future” days. We repeat this process 44 times and then calculate the sum of the cumulative returns each model had. This process is visualized in Figure 5 and analyzed in Algorithm 1.

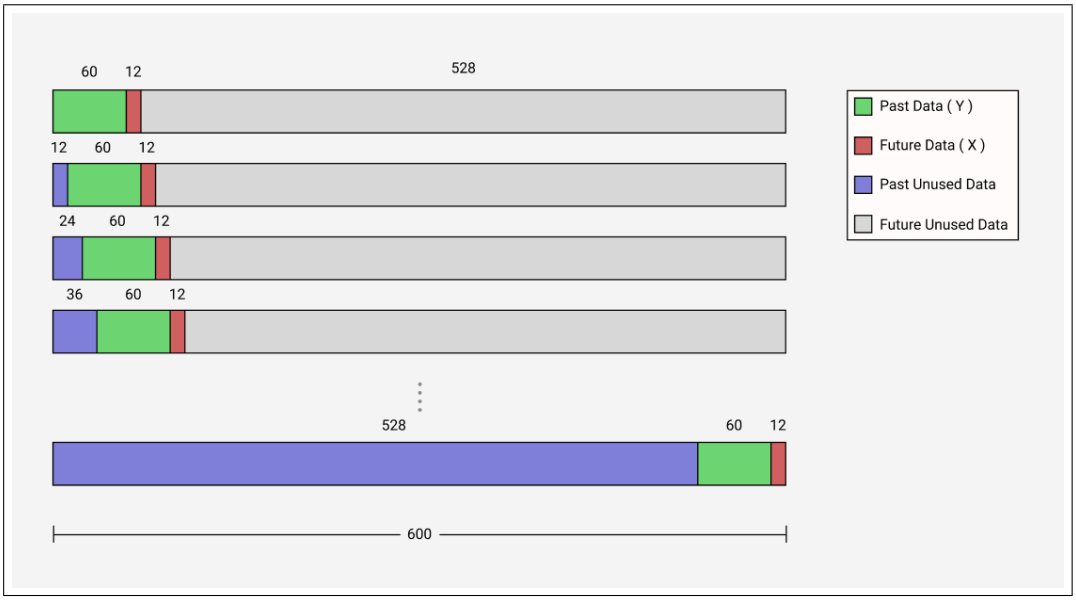


Figure 5: The green data form the *R* we defined in (2.3). This optimization is implemented 45 times in total creating 45 portfolios for each model. Each time, the portfolio created is used to calculate the model returns of the next 12 days (red data).

Algorithm 1 Multiple Executions of each Optimization Model

Input
(600 $\times$ 48) Matrix ***R*** of FF48 Dataset
Output
(45 Cumulative Returns of the 45 Calculated Portfolios)

$i \leftarrow 60$
$OverallReturns = [\,]$
while $i < 600$ **do**
  $PastData \leftarrow R[i-60 : i]$ ▷ Green Data in Figure 5
  $FutureData \leftarrow R[i : i+12]$ ▷ Red Data in Figure 5
  $Portfolio \leftarrow$ **CalculatePortfolio**$(PastData)$ ▷ Optimization Model execution
  $Returns[0:12] \leftarrow$ **CalculateReturns**$(Portfolio, FutureData)$
  $OverallReturns.append(Returns)$
  $i \leftarrow i+12$
end while
$CumulativeReturns =$ **CalculateCumulativeReturns**$(OverallReturns)$

3.3 Conventional Optimization Models

3.3.1 CVXpy

CVXpy is a Python-embedded modeling language for convex optimization problems [8][7]. We chose this as a first approach since its use is pretty straightforward and generally yields good results for convex problems [9].

As the CVXpy solver, we selected the ECOS solver, since it's optimal for second-order cone programs (SOCPs) [11]. The objective function wasn't changed by this model, CVXpy solved the least squares optimization problem (4).

3.3.2 LASSO regression using Proximal Gradient Descent

L_1 regularization

Another frequent approach in convex optimizations is to add a regularizer to the objective function. In our implementation, we chose the $\boldsymbol{l_1}$ penalty:

$$\lambda||w||_1$$

This penalty is comprised of two parts:

- $||w||_1 = \sum_{i=0}^{N} |w_i|$
- λ is a parameter that allows us to adjust the relative importance of the l_1 penalization in our optimization

By adding that penalty, the objective function **(4)** becomes:

$$\begin{aligned} \hat{w} = \arg\min_{w} \quad & \frac{1}{T} E[|Rw - \rho 1_T|_2^2] + \boldsymbol{\lambda||w||_1} \\ \text{s.t.} \quad & w^\top \mu = \rho \\ & \sum_{i=0}^{N} w_i = 1 \end{aligned} \tag{5}$$

This modification converts the problem into a **LASSO Regression Problem** [12].

l_1 Benefits

Adding an $\boldsymbol{l1}$ penalty to an optimization problem ”encourages” the model to shrink the less important feature’s coefficients to zero (in our case it suggests to only invest in a few of the available industries). This poses several advantages [1]:

1. **It yields sparse results, which is something investors generally prefer**. This is because it is simpler and easier to monitor and liquidate fewer securities.
2. **It minimizes transaction costs**. Generally, transaction costs can be split into two categories. Fixed costs independent of the size of the investment (fixed salaries, fixed fees, etc.) and variable costs that are relative to the size of the investment (commissions, taxes, etc.) [13]. The latter cost is the same regardless of the number of different stocks in the portfolio (since $\sum_{i=0}^{N} w_i = 1$). However, the fixed costs are minimized if our portfolio is sparse.
3. **It promotes stability**. Adding an $\boldsymbol{l_1}$ penalty protects the model from possible collinearities between the securities [14].

Proximal Gradient Descent

Performing normal Gradient Descent to optimize the objective function (5) is not possible, since the l_1 penalty is non-differentiable. To address this, we solve the LASSO

Regression Problem using Proximal Gradient Descent [17]. It is a generalized form of projection used to solve non-differentiable convex optimization problems.

The proximal mapping for a lasso objective is calculated by:

$$\begin{aligned} prox_t(\beta) &= \arg\min_z \frac{1}{2t}||y - \boldsymbol{X}\beta|| + \lambda||z||_1 \\ &\equiv \arg\min_z \frac{1}{2}||y - \boldsymbol{X}\beta|| + \lambda t||z||_1 \\ &= \boldsymbol{S}_{\lambda t}(\beta) \end{aligned} \tag{6}$$

$\boldsymbol{S}_{\lambda t}$ is called the **Soft-Thresholding operator**.

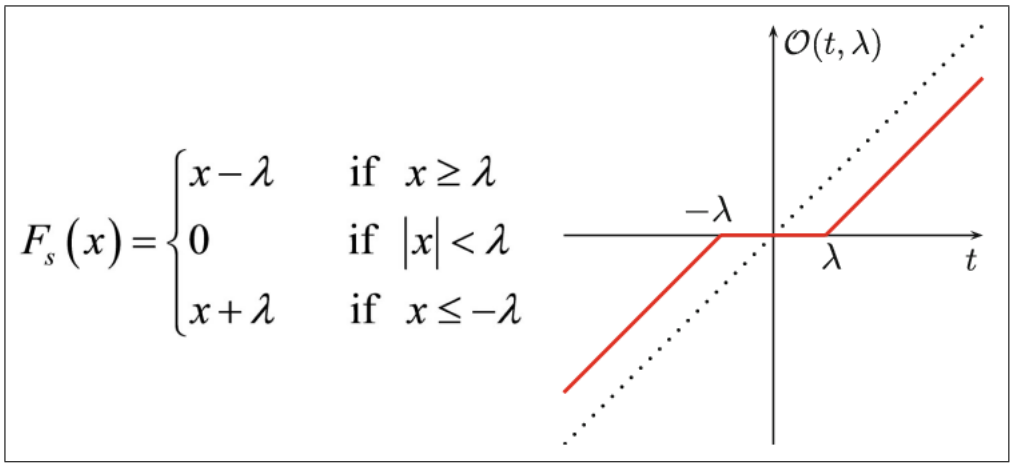


Figure 6: Soft-Thresholding operator. (Sources: [35] [38])

This means that ultimately, the problem is solved by implementing gradient descent on the objective function (4) and soft-thresholding the result of every iteration by $\lambda\tau$:

$$prox_{L1}(\beta) = S_{\lambda\tau}(\beta)$$

β is the result of a gradient descent iteration of the least squares equation **(4)**.

Algorithm 2 describes the steps of this process.

Algorithm 2 Proximal Gradient Descent

Input
(60 $\times$ 48) Matrix **X** (Green Data in Figure 5)
Output
Optimal Portfolio w
Our Functions
Objective Function: $\boldsymbol{F}(X, w)$
Objective Function's Gradient: $\boldsymbol{G}(X, w)$
Function That Determines Learning Rate Multiplier: $\boldsymbol{DLR}(start\_lr, X, F, G, w)$
Soft Threshold Function $\boldsymbol{S}(\tau, w)$

$max\_iter \leftarrow 100$ ▷ max number of gradient descent iterations
$\varepsilon \leftarrow 0.001$ ▷ criterion to end gradient descent
$iter \leftarrow 0$
$\tau \leftarrow 0.2$ ▷ importance of the l1 norm
$start\_lr \leftarrow 0.1$ ▷ starting learning rate

$w \leftarrow [\frac{1}{48}, \frac{1}{48}, \frac{1}{48}, ..., \frac{1}{48}]^\top$ ▷ begin with equally weighted portfolio

while $(norm_2(grad) > \varepsilon$ **and** $iter < max\_iter)$ **do**
  $f \leftarrow \boldsymbol{F}(X, w)$
  $grad \leftarrow \boldsymbol{G}(X, w)$
  $lr \leftarrow \boldsymbol{DLR}(start\_lr, X, f, grad, w)$
  $w \leftarrow w - lr * grad$
  $w \leftarrow \boldsymbol{S}(\tau, w)$
  $w \leftarrow \boldsymbol{Normalize}(w)$ ▷ normalize result to maintain $\sum_{i=0}^{48} w_i = 1$ constraint

  $iter = iter + 1$
end while
return w

3.3.3 Sparse Equally-Weighted portfolio replication

Equally Weighted Portfolio

As mentioned before, the **equally weighted portfolio w_{ew}** is a tough benchmark when it comes to Markowitz portfolio formulation. The reason is that this portfolio is as diversified as possible, and therefore it has relatively low variance.

$$w_{ew(i)} = \frac{1}{N}$$

Replication of w_{ew} Problem

For this reason, we attempt to create a k-sparse portfolio that replicates the variance of the equally weighted portfolio [5]. Hopefully, this will result in performance similar to the w_{ew}, along with the advantages of sparsity (see 3.3.2).

This approach amounts to solving:

$$\begin{aligned} \hat{w} = \arg\min_{w} \quad & ||Rw - Rw_{ew}||_2^2 \\ \text{s.t.} \quad & \sum_{i=0}^{N} w_i = 1 \\ & w \in \boldsymbol{\Sigma_k} \end{aligned} \tag{7}$$

$\boldsymbol{\Sigma_k}$ is the set of k-sparse vectors (***N*** sized vectors with ***k*** non-zero values).

Solving the Problem

This problem is solved using **Projected Gradient Descent** [17], using the **G**reedy **S**elector and **S**implex **P**rojection (**GSSP**) algorithm from Kyrillidis (et al. 2013)[28]. This algorithm does the following in every Gradient Descent iteration's result w^+:

1. **Greedy Selector**: Keeps the k-largest entries of w^+ and sets the rest to zero.
2. **Simplex Projection**: Projects the result into the simplex in order to maintain the $\sum_{i=0}^{48} w_i = 1$ constraint.

   The projection applied is the **Euclidean Projection** $\mathcal{P}_{\boldsymbol{\lambda}}$:

   $$(\mathcal{P}_{\boldsymbol{\lambda}}(w_i^*)) = w_i^* - t, \text{ where } t = \tfrac{1}{N}(\textstyle\sum_{i=1}^{N} w_i^* - \lambda)$$

   Where:

   - w^* is the result of step's 1 Greedy Selector.
   - N is the number of dimensions of w, in our case 48.
   - $\lambda = \sum_{i=1}^{N} w_i$, in our case 1.

We created 2 models using this algorithm, with $k = 5$ and $k = 10$.

3.3.4 CVXpy-layers

CVXpy-layers is a Python library for constructing differentiable convex optimization layers in PyTorch, JAX, and TensorFlow using CVXpy. A convex optimization layer solves a parametrized convex optimization problem in the forward pass to produce a solution. It computes the derivative of the solution with respect to the parameters in the backward pass.

Effectively, it should yield similar results to CVXpy, so we selected it to check the validity of the CVXpy implementation. We also used it to attempt another equally weighted portfolio replication (see 3.3.3), this time with no sparsity constraints.

3.4 Creating a Markowitz Portfolio using LSTM's Predicted Returns

In our last approach, we decided to use a Neural Network to predict the returns of the days in the future window (red data in Figure 5) and then optimize based on the predicted returns.

3.4.1 Neural Network

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks, which have their roots in artificial intelligence, are swiftly gaining popularity in the development of trading systems [18]. In recent years neural networks have been widely used for forecasting financial information. For our final model, we used a newly formed class of neural networks, the **Recurrent Neural Network**.

3.4.2 Recurrent Neural Network

A **Recurrent Neural Network** (**RNN**) is a type of artificial neural network which has "memory" as it takes information from prior outputs to influence the current input. RNNs are excellent for training with time series data [21]. However, it can be difficult to train standard RNNs to solve problems that require learning long-term temporal dependencies (such as ours). This is because the gradient of the loss function decays exponentially with time (called the vanishing gradient problem). For this reason, we selected to use a **L**ong **S**hort-**T**erm **M**emory unit (**LSTM**) [24][27].

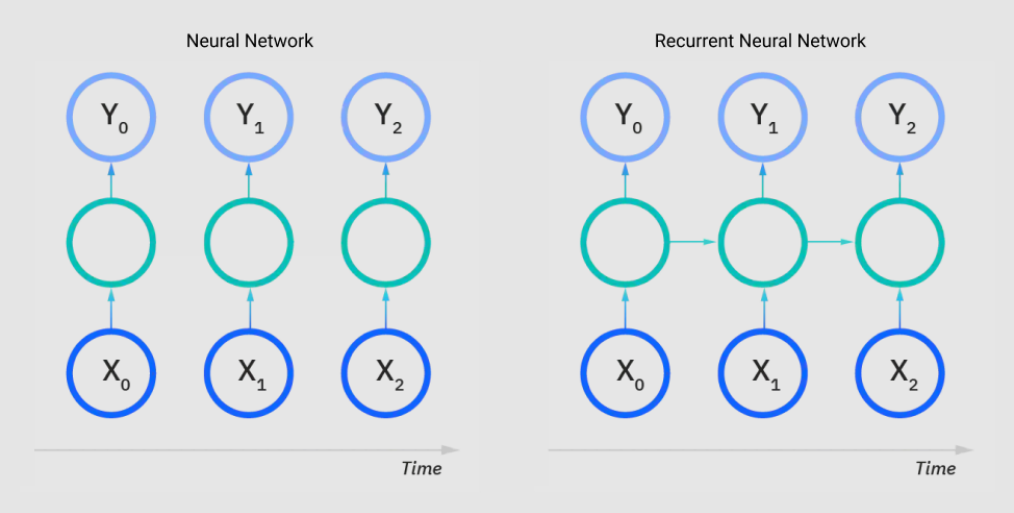


Figure 7: Recurrent Neural Networks use prior outputs as inputs, making them superior for time series predictions. (Source: [21])

3.4.3 Long Short-Term Memory Unit

Long **S**hort-**T**erm **M**emory units (**LSTMs**) are designed to detect long-term dependencies. They achieve this by adding these gates to the hidden layer:

1. Input gate
2. Output gate
3. Forget gate

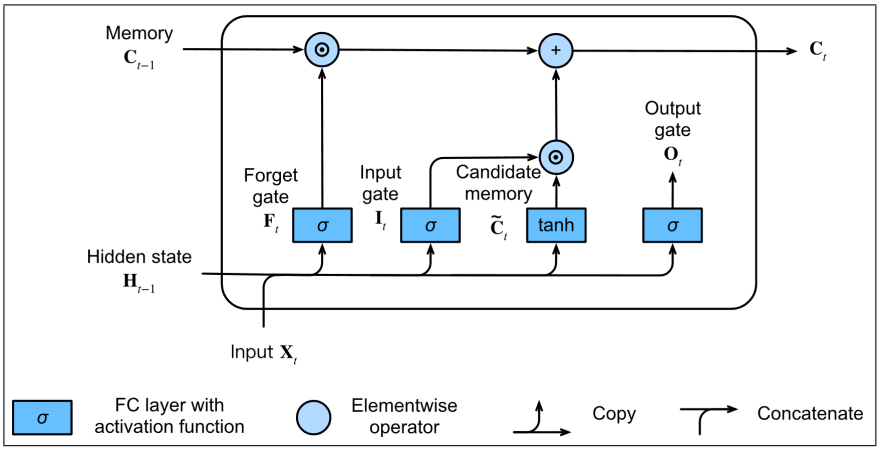


Figure 8: LSTM Cell. (Source: [10])

These three gates have independent weights and biases that and are used decide how much of the current input to keep, how much information is no longer required and how much of the internal hidden state to send to the output [23].
As a network optimizer we chose the Adam optimizer. That algorithm uses a decaying learning rate, which means that the steps become smaller as the optimizer reaches the objective, which is not only good for execution time but also for accuracy [22]. Figure 9 displays the Adam optimizer's performance compared to other optimizers.

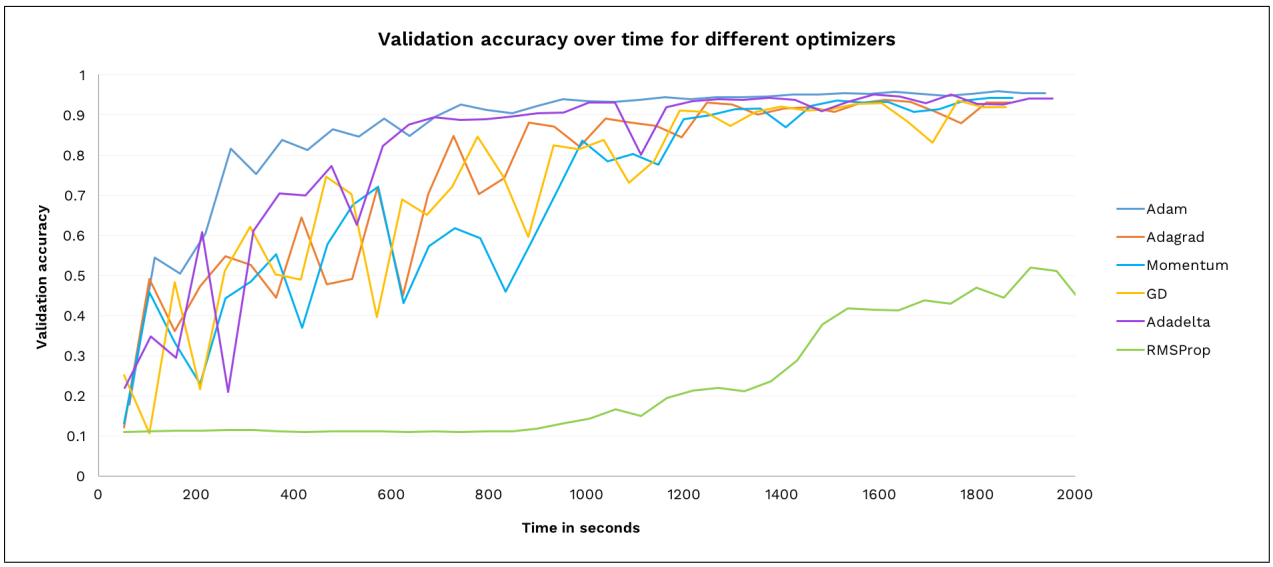


Figure 9: Keras Optimizers Validation Accuracy (Source: [34])

3.4.4 Our Deep Learning Model's implementation steps

1. Train an LSTM unit to receive 60 time series data (red data size in Figure 5) of an industry's returns and predict the return of the following day. Repeat for each industry (48 times, training 48 models).
2. Use each model to predict on a data set of 600 time-series data (same data the other models use), with a 12-day prediction depth. To achieve that depth, we use earlier predictions as input for the following ones.
3. Concatenate the results of the 48 units, creating an array $\boldsymbol{R^*}$ of size $\mathbf{540} \times \mathbf{48}$.
4. Run an optimization model on the $\boldsymbol{R^*}$ array. The only difference with the previous optimizations is that in each iteration the optimization model wills use an array of size $\mathbf{12} \times \mathbf{48}$ (predicted red data) instead of $\mathbf{60} \times \mathbf{48}$ (green data). Simply put, this time we will run our optimizers on ”predicted future windows” instead of ”past windows”.

3.4.5 Data Preprocessing

Data

To train our model, we used a training set containing daily returns for every industry over 2100 days.

Scaling

Feature scaling is essential for machine learning algorithms that calculate distances between data. To scale our data, we used sclearn's **MinMaxScaler**. This scaler translates each feature individually such that it is in the given range on the training set, in our case between zero and one. The new training set values (x_{scaled}) are calculated by:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

3.4.6 Network Setup

There are many ways of setting up a neural network. However, due to the nature of the data, it is sometimes hard to decide which direction to take. In our case, we tried different tuning until we achieved a very high **Correlation Coefficient** between predicted and expected prices.

Our model is comprised of 3 LSTM layers with 50 neurons and 1 Dense layer. Between each LSTM layer we also added a 20% **Dropout Rate** to prevent overfitting [26]. The network was trained for 50 epochs with a batch size of 3. We implemented it using the Keras library [27]. To train it, we used 2040 vectors containing 60 ”past” days returns (**X**) and each following day as the output (**Y**).

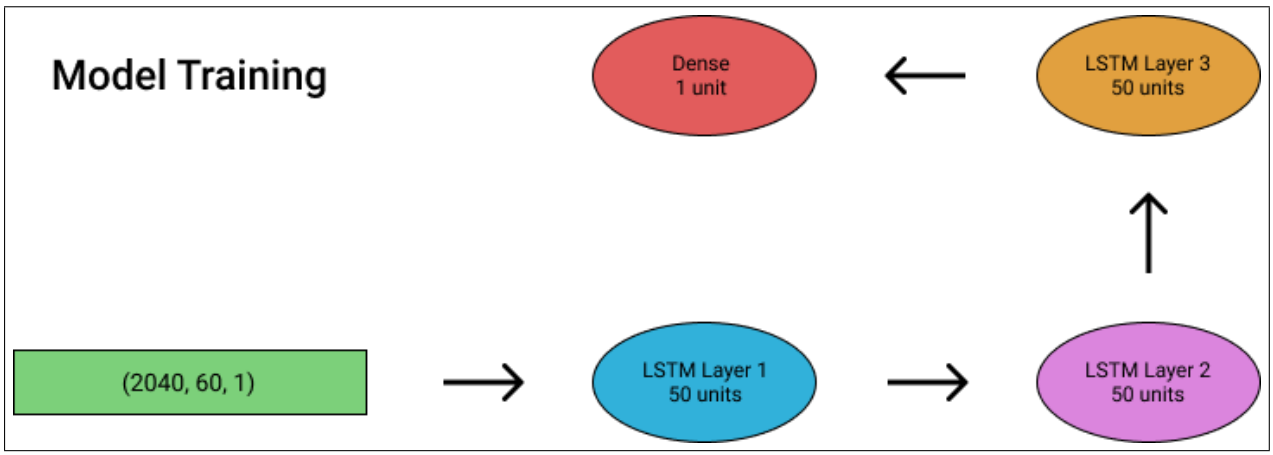


Figure 10: Model Training Overview

Making Predictions

Our Predictions are made using the same data set the conventional optimizers used. Since we wanted to predict 12 days ahead, we decided to use our own predictions as input for the predictions further in. Figure 11 describes this process.

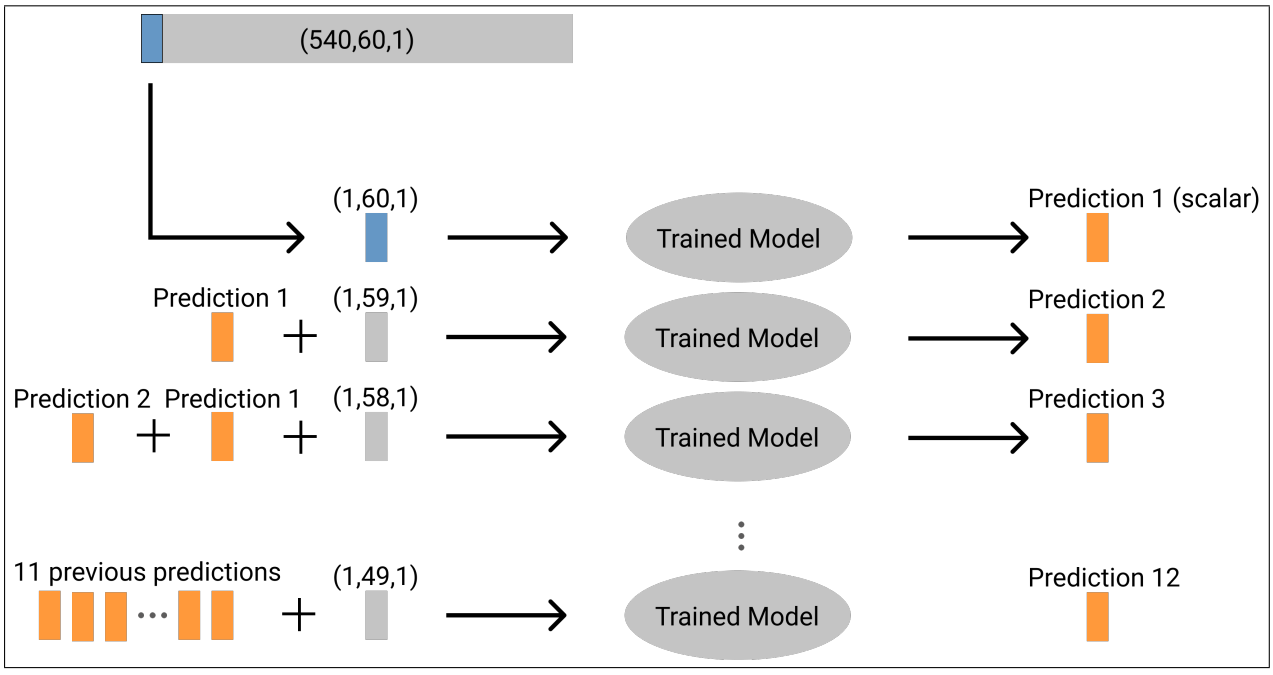


Figure 11: Making Return Predictions regarding one Industry's returns. To achieve prediction depth, earlier predictions are used for the following ones. Reset every 12 days (e.g.: predicted returns on days 61, 73 and 85 should be the most accurate, while days 72, 84 and 96 should be the hardest to make predictions on).

Optimizing based on Predicted Returns

Finally, using the predicted returns we calculated, we needed to run an optimizer to create our proposed portfolios. We chose the CVXpy optimizer, since it showed the most promising returns in our previous optimizations, as our experimental evaluation will indicate.

4. EXPERIMENTAL EVALUATION

In this Chapter we evaluate the effectiveness of our models. Firstly, we go over the accuracy of our LSTM unit regarding it's predictions on an asset's prices over a forecasting window of 12 days. Secondly, we examine and compare our optimization models, when it comes to creating Markowitz portfolios.

4.1 LSTM Predictions

4.1.1 Results

As an example and for visualization purposes, Figure 12 displays our predictions for the value of an investment in the agricultural industry, compared with the actual value during this period of 540 days.

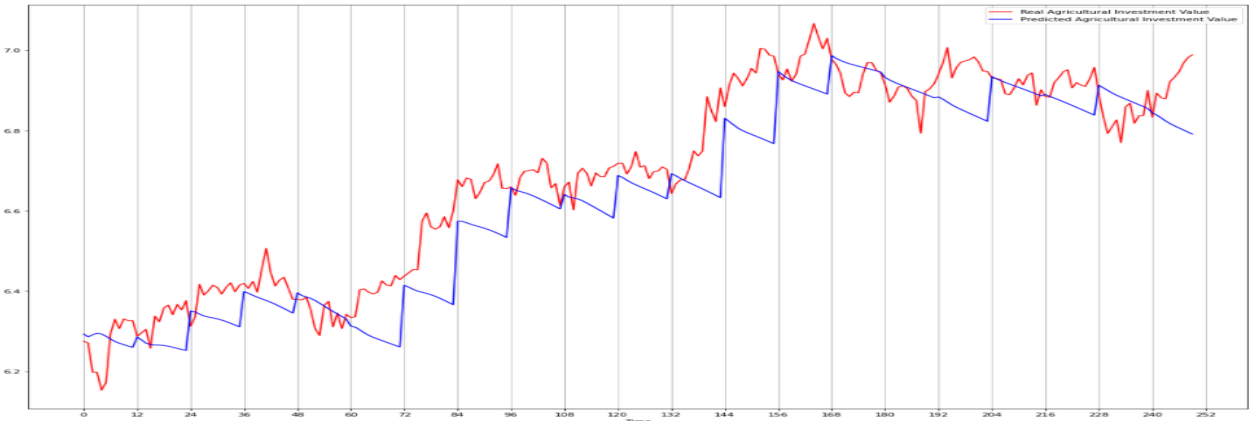


Figure 12: Real Agricultural Industry Investment value (red) vs predicted through our model (blue).

4.1.2 Prediction depth and accuracy

As stated in the previous Chapter, to make 12 predictions in each step, we decided to use our earlier predictions as input for the latter ones. This approach would of course mean that the bigger our prediction depth got, the harder it would be to make more accurate predictions (Figure 13)). In spite of this problem though, our model performed relatively well. We quantified its accuracy by calculating the **Correlation Coefficient** between the predicted value of an investment and the real one.

$$cc = \frac{\sum_{i=0}^{T}((p_i - \bar{p})(r_i - \bar{r}))}{\sqrt{(\sum_{i=0}^{T}(p_i - \bar{p}))(\sum_{i=0}^{T}(r_i - \bar{r}))}}$$

Where:

1. cc is the Correlation Coefficient.
2. p_i, r_i are the predicted and real value of the investment in an Industry Portfolio (one of our 48 assets) respectively.
3. $\bar{p}, \bar{r}$ are the mean of the predicted and real values of the investment in an Industry Portfolio respectively.

4. T is the size of the data, in our case 540 days.

The relation between the prediction depth and Correlation Coefficient is shown in Figure (14).

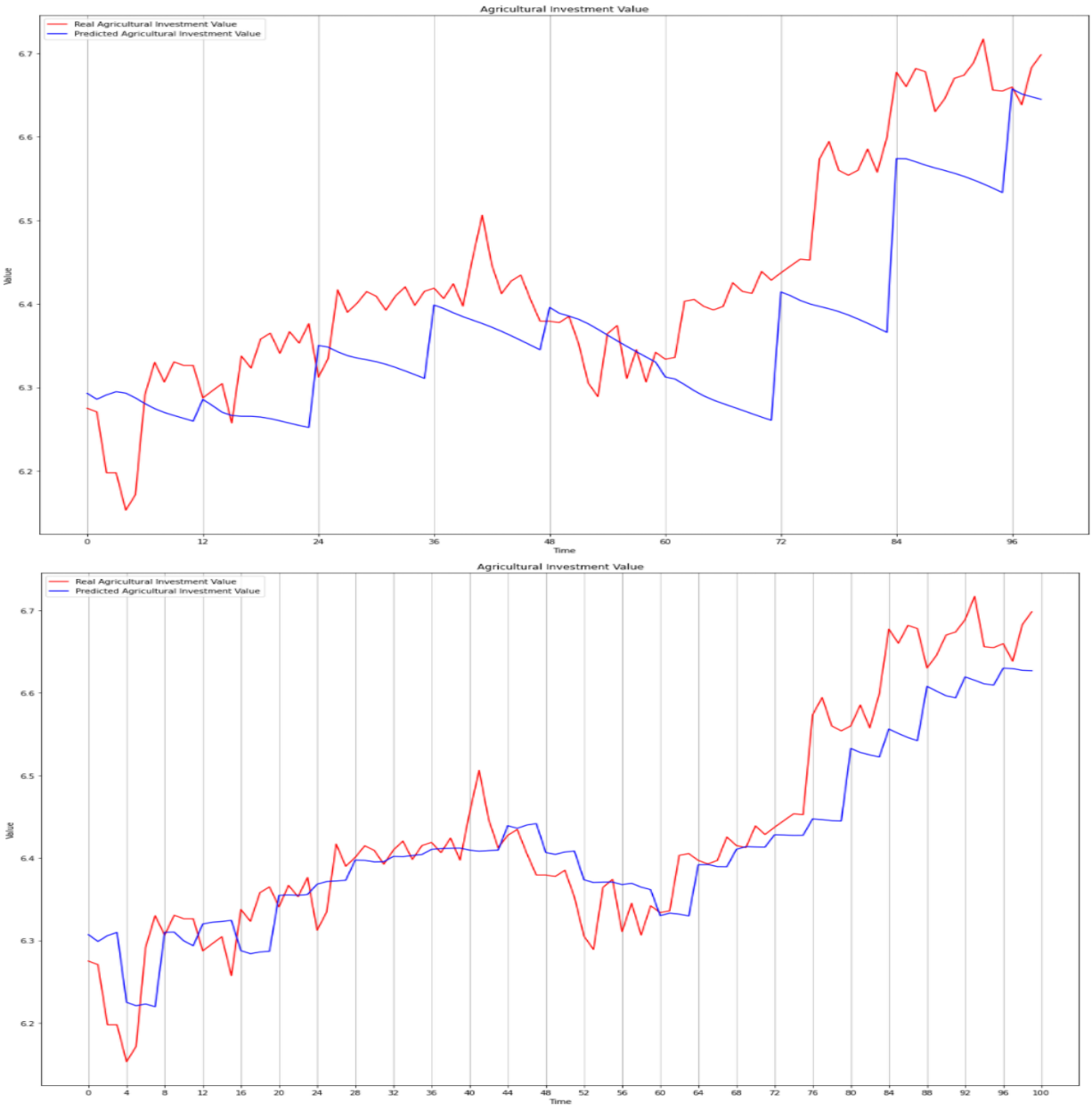


Figure 13: Model predictions with different prediction depth. The first graph has a depth of 12, while the second only has a depth of 4. For more clarity, the plots are referring to predictions of the first 100 days.

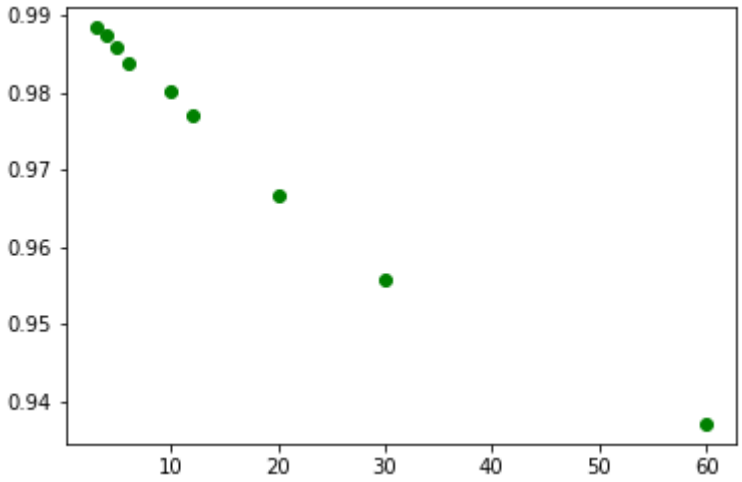


Figure 14: Correlation Coefficient in relation to Prediction Depth (number of predicted days in each step)

Note that the graphs show the value of an investment in the agricultural industry and not its daily returns. Therefore, we expect the Correlation Coefficient to be high, since the value of an investment in a FF48 industry is unlikely to be volatile over 100 days (the daily returns however can most definitely vary from day to day).

4.2 Comparing our models

To compare all our models, we calculated the **sum of cumulative returns** of all the portfolios each model created. These are given by:

$$Creturn_{(i)} = \frac{Value_{(i)} - Value_{(i-1)}}{Value_{(i-1)}}$$
$$Csum = \sum_{i=1}^{45} Creturn_{(i)} \tag{8}$$

Figure 15 displays the cumulative returns sum of each model over the time period of 540 days. Figure 16 displays the final cumulative returns each model yielded.

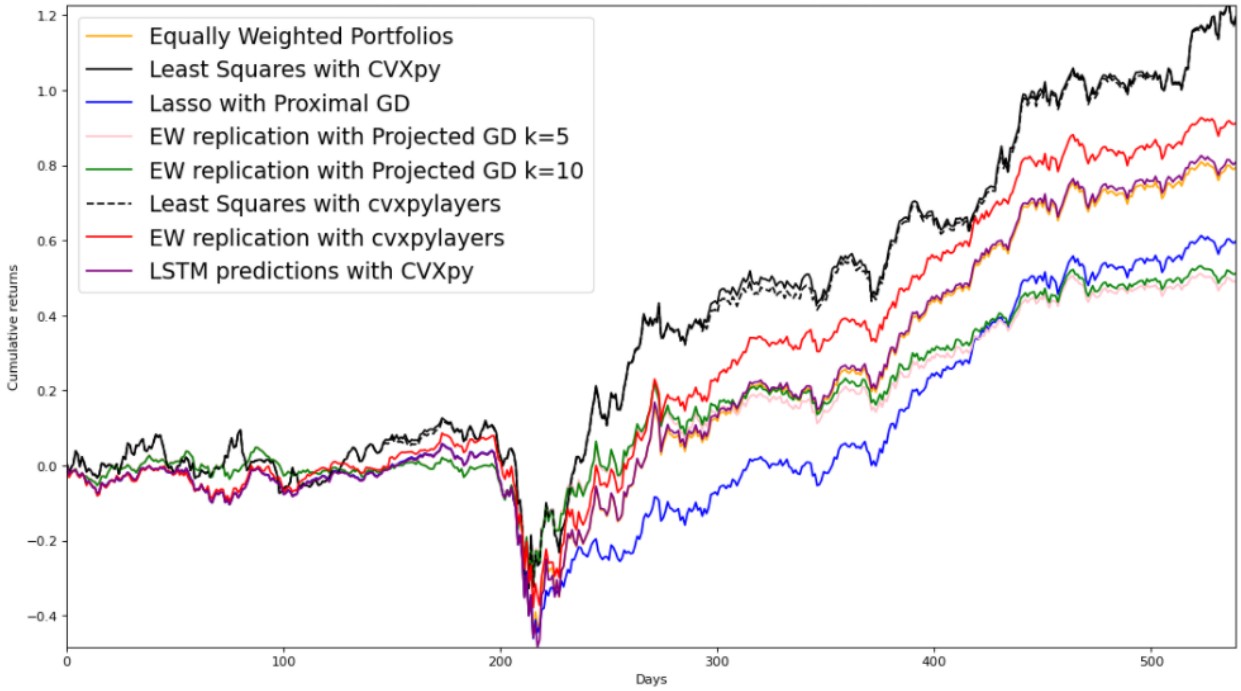


Figure 15: Sum of Cumulative Returns of each model over 540 days

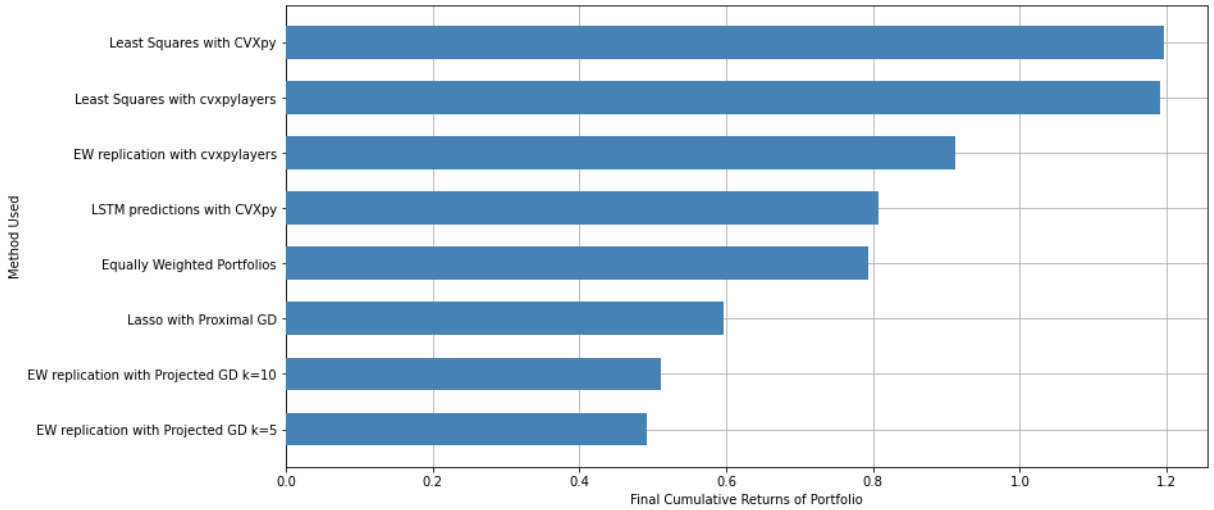


Figure 16: Final Returns of each model. For example, the value of investment regarding the model which uses CVXpy to solve the Least Squares Objective Function will be: $\approx 1.2 + 1 = 2.2$ at the end of the optimization series.

5. CONCLUSIONS

In this Chapter, we draw our conclusions based on the evaluations. We also propose additional modifications that could further improve our models.

5.1 Analysis

The results indicate that CVXpy produces the most successful portfolios. However, it appears our Deep Learning approach had relatively good results too, since it surpassed most of the conventional methods. It also proved itself slightly superior to the naive (equally weighted) portfolio approach, which is a hard benchmark for most models to significantly or consistently outperform. It should also be noted that our portfolios derived from Proximal and Projected Gradient descent could also be of value since what they lose in performance they somewhat gain from the benefits of sparsity (see 3.3.2).

5.2 Further Modifications

Modifications and variants that could improve our models:

1. The LSTM model would most likely yield better results if the training set length and the timestamp used to make predictions were larger (they were 2000 days and 60 days respectively). However, this would fall beyond the scope of our paper, since our goal was to propose this new approach of optimizing Markowitz portfolios, and increasing these sizes would significantly increase the training time of our unit.
2. Another modification is to train an LSTM model to predict the returns of all 48 industries instead of training 48 models for each industry. This would of course add a lot of complexity to our problem, but this way the model would possibly also find patterns between correlated industries, and thus yield better results.
3. Another optimizer besides CVXpy could be used to created portfolios based on LSTM's predictions. Even if they performed worse, they could have other advantages like sparsity or low volatility.

Finally, these models could be used to solve not only other portfolio optimization problems besides the Markowitz portfolio (such as the trade-off problem), but also other optimization problems in general.

REFERENCES

[1] Brodie, J., Daubechies, I., De Mol, C., Giannone, D. & Loris, I. Sparse and Stable Markowitz Portfolios. *PNAS*. (2008)

[2] Li, Q. & Yanqin Bai Optimal trade-off portfolio selection between total risk and maximum relative marginal risk. *Taylor & Francis*. (2015)

[3] Times, E. Definition of Risk. https://economictimes.indiatimes.com/definition/risk

[4] Vaidya, D. Systematic Risk vs Unsystematic Risk. https://www.wallstreetmojo.com/systematic-risk-vs-unsystematic-risk

[5] McKenzie, D. Optimal Sparse Markowitz Portfolios. (2017)

[6] R. French, K. U.S. Research Returns Data. https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html#BookEquity

[7] Diamond, S. & Boyd, S. CVXPY: A Python-embedded modeling language for convex optimization. *Journal Of Machine Learning Research*. **17**, 1-5 (2016)

[8] Agrawal, A., Verschueren, R., Diamond, S. & Boyd, S. A rewriting system for convex optimization problems. *Journal Of Control And Decision*. **5**, 42-60 (2018)

[9] Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S. & Kolter, J. Differentiable Convex Optimization Layers. *Advances In Neural Information Processing Systems*. pp. 9558-9570 (2019)

[10] Zhang, A., Lipton, Z., Li, M. & Smola, A. Dive into Deep Learning. (2021), Online Version: https://d2l.djl.ai/chapter_recurrent-modern/lstm.html

[11] Anon. CVXPY Advanced Features. https://www.cvxpy.org/tutorial/advanced/index.html

[12] Nagpal, A. L1 and L2 Regularization Methods. (2017), https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c

[13] Kociński, M. Transaction costs and market impact in investment management. *E-Finanse: Financial Internet Quarterly*. **10** pp. 28-35 (2014)

[14] Daubechies, I., Defrise, M. & De Mol, C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. (2004)

[15] Victor DeMiguel, Lorenzo Garlappi, Francisco J. Nogales, Raman Uppal, (2009) A Generalized Approach to Portfolio Optimization: Improving Performance by Constraining Portfolio Norms. Management Science 55(5):798-812.

[16] Ravi Jagannathan, Tongshu Ma. Risk Reduction in Large Portfolios: Why Imposing the Wrong Constraints Helps. *The Journal of Finance*. (Aug., 2003).

[17] Tibshirani, R. Proximal Gradient Descent (and Acceleration). https://www.stat.cmu.edu/ ryantibs/convexopt/lectures/prox-grad.pdf

[18] Chen, J. Neural Network. (2020), https://www.investopedia.com/terms/n/neuralnetwork.asp

[19] Yann LeCun, Yoshua Bengio, Geoffrey Hinton. Deep learning. Nature, 521(7553):436–444, (2015).

[20] Zhang, Z., Zohren, S. & Roberts, S. Deep Learning for Portfolio Optimization. *The Journal Of Financial Data Science*. **2**, 8-20 (2020,8), http://dx.doi.org/10.3905/jfds.2020.1.042

[21] Anon. What are Recurrent Neural Networks. (2020), https://www.ibm.com/cloud/learn/recurrent-neural-networks

[22] Brownlee, J. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. (2021), https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[23] Olah, C. Understanding LSTM Networks. (2015), https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[24] K. Cao, H., K. Cao, H. & T. Nguyen, B. DELAFO: An Efficient Portfolio Optimization Using Deep Neural Networks. (2020)

[25] James Chen. Exchange Traded Fund. (2021), https://www.investopedia.com/terms/e/etf.asp

[26] Brownlee, J. Dropout Regularization in Deep Learning Models With Keras. (2020), https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/

[27] Mwiti, D. Using a Keras Long Short-Term Memory (LSTM) Model to

Predict Stock Prices. (2018), https://www.kdnuggets.com/2018/11/keras-long-short-term-memory-lstm-model-predict-stock-prices.html

[28] Kyrillidis, A., Becker, S., Cevher, V. & Koch, C. Sparse projections onto the simplex. *Proceedings Of The 30th International Conference On Machine Learning*. **28**, 235-243 (2013,6,17), https://proceedings.mlr.press/v28/kyrillidis13.html

[29] Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep learning. MIT press, (2016).

[30] Chauhan, N. & Singh, K. A Review on Conventional Machine Learning vs Deep Learning. *2018 International Conference On Computing, Power And Communication Technologies (GUCON)*. pp. 347-352 (2018)

[31] Yaoyao Clare, D. A Multi-Objective Approach to Portfolio Optimization. *Rose-Hulman Undergraduate Mathematics Journal*. **8** (2007)

[32] Ganegedara, T. Stock Market Predictions with LSTM in Python. (2020), https://www.datacamp.com/community/tutorials/lstm-python-stock-market#average

[33] Singh, R. & Srivastava, S. Stock prediction using deep learning. *Multimedia Tools And Applications*. **76**, 18569-18584 (2017,9), https://doi.org/10.1007/s11042-016-4159-7

[34] David Mack. How to pick the best learning rate for your machine learning project. (2018), https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2

[35] Hoang, T., Barney Smith, E. & Tabbone, S. Sparsity-based edge noise removal from bilevel graphical document images. *International Journal On Document Analysis And Recognition (IJDAR)*. (2014,8)

[36] The guide to diversification. https://www.fidelity.com/viewpoints/investing-ideas/guide-to-diversification

[37] Covariance Matrix. https://en.wikipedia.org/wiki/Covariance_matrix

[38] A. Al Jumah, M. Gulam Ahamad and S. Amjad Ali, ”Denoising of Medical Images Using Multiwavelet Transforms and Various Thresholding Techniques,” Journal of Signal and Information Processing, Vol. 4 No. 1, 2013, pp. 24-32.