Lappeenranta-Lahti University of Technology LUT

School of Business and Management

Master's Programme in Strategic Finance and Analytics

Master's Thesis

Deep Reinforcement Learning in Portfolio Management: Policy Gradient method for S&P-500 stock selection

Tommi Huotari

August 2019

Supervisor and 1st examiner: Jyrki Savolainen

2nd examiner: Mikael Collan

TIIVISTELMÄ

Tekijä:	Tommi Huotari
Tutkielman nimi:	Deep Reinforcement Learning in Portfolio Management: Policy Gradient method for S&P-500 stock selection
Akateeminen yksikkö:	LUT School of Business and Management
Koulutusohjelma:	Master's Programme in Strategic Finance and Analytics
Ohjaajat:	Jyrki Savolainen ja Mikael Collan
Hakusanat:	Koneoppiminen, portfolion optimointi, deep reinforcement learning

Tämän maisterintutkielman tavoitteena on tutkia syvän vahvistusoppimisen (deep reinforcement learning, DRL) soveltuvuutta salkunhoitoon S&P500-indeksin osakkeista koostuvan osakeportfolion riskikorjatun tuoton parantamiseksi. Tarkoituksena on luoda DRL-agentti, joka pystyy rakentamaan ja hallitsemaan itsenäisesti osakesalkkua analysoimalla julkisten yhtiöiden päivittäistä tuottoa, Earnings to price-tunnuslukua, osinkotuottoa sekä kaupankäyntimääriä.

Tutkimuksen datana käytettiin vuoden 2013 tammikuun S&P500-indeksin osakkeita vuosilta 1998-2018, joista oli jätetty pois vuoden 2004 jälkeen listautuneet yhtiöt (84). Käymällä kauppaa jäljellä olevilla 416 osakkeella, tutkimuksessa kehitetty agentti pystyi kasvattamaan salkun riskikorjattua tuottoa yli indeksin sekä kaikkien vertailuportfolioiden, saavuttaen 329% tuoton 5 vuoden testijakson aikana. Osakkeiden kaupankäyntimäärät todettiin tutkimuksessa agentin suorituskyvyn kannalta hyödyttömäksi lähtötiedoksi. Agentin osoitettiin parantavan merkittävästi osakeportfolion riskikorjattua sekä kokonaistuottoa, mutta sen kaupankäyntistrategian todettiin olevan ennemmin opportunistinen, kuin jatkuvaan ylituottoon kykenevä.

Saavutetuista ylituotoista huolimatta, agentin ei kuitenkaan osoitettu ylittävän tilastollisesti merkitsevästi markkinatuottoa. Tulokset tukevat aiempia havaintoja syvän vahvistusoppimisen sovellettavuudesta salkunhoidossa.

ABSTRACT

Author:	Tommi Huotari
Title:	Deep Reinforcement Learning in Portfolio Management: Policy Gradient method for S&P-500 stock selection
School:	LUT School of Business and Management
Academic programme:	Master's Programme in Strategic Finance and Analytics
Supervisors:	Jyrki Savolainen ja Mikael Collan
Keywords:	Machine learning, portfolio optimization, deep reinforcement learning

The goal of this Master's Thesis is to investigate the applicability of deep reinforcement learning (DRL) to portfolio management in order to improve the risk-adjusted returns of a stock portfolio with S&P500 constituents. The objective is to create a deep reinforcement learning agent, able to independently construct and manage a portfolio of stocks by analysing the daily total return, Earnings to price, Dividend yield and trading volume of public companies.

The dataset in the study contained the S&P500 constituents at January 2013 covering the years from 1998 to 2018. The companies listed after 2004 (84) were left out from the final dataset. By trading the remaining 416 stocks, the agent developed in the study was able to increase the risk-adjusted returns of the portfolio over the stock index and all the benchmarks, achieving 329% returns during the 5-year test period. Trading volume was found to be an ineffective variable for the model in portfolio management. The agent was demonstrated to significantly improve the risk-adjusted and the total returns of the stock portfolio, however, its trading behaviour was found opportunistic, rather than continuously beating the market index.

Despite from the excess returns, the agent was not shown to statistically significantly outperform the market performance. The results support the previous findings about the applicability of DRL to the portfolio management.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1. Background and Motivation

Through time investors have been seeking for higher than average returns at the financial markets with active portfolio management. However, high returns go hand in hand with the risk. Fama's (1970) theory of efficient capital markets assumes that it is impossible to achieve excess returns by active stock picking and thus investors should only focus on picking the proper risk-level that fits to one's risk-taking ability.

Modern portfolio theory was introduced in 1950s by Harry Markowitz and the central idea is that an investor may maximize the expected return of the portfolio at the given risk level by selecting the combination of assets from the *efficient frontier*. Efficient frontier represents a set of portfolios, whose expected return is optimal at the given risk level. The theory suggests investors to be risk averse preferring the less risky portfolio from two alternatives with equal expected return. The weakness of the theory is the lack of correlation with historical and future stock returns and thus, an efficient portfolio constructed based on historical returns might not be efficient at all in the future.

Machine learning (ML) is a field of scientific study, which focuses on algorithms that independently learn from data to perform specific tasks without precisely defined instructions. The use of ML has been growing steadily since its earliest applications in 1950s and ML is currently used everywhere from mobile applications to home appliances. Machine learning has become popular in financial applications as well, especially in credit scoring (Tsai and Wu, 2008) and credit card fraud detection (Chan and Stolfo, 1998). Due to the increased computing power and recent breakthroughs in scientific research, *deep learning* models – organic neuro-system inspired machine learning models also referred as *artificial neural networks* (ANNs) – have taken the central position in the machine learning scene. Deep learning models are based on a multi-layer structure with neuron-like connections between the layers which are able to model complex and non-linear structures in the data.

Reinforcement learning is an area of machine learning, where the learning happens via trial and error. The model performance with the dataset is evaluated with a reward function, which is intended to maximize during the learning process. The key idea is that the model learns without pre-information the behaviour that leads to the maximal reward signal (Sutton and Barto, 2018, 1-2). Reinforcement learning field has grown tremendously during the latest years after the successful unification with the deep learning models. Although the ideas of reinforcement and deep learning have been developed ages ago, only the breakthroughs during the present decade have finally enabled the adaptation of the methods successfully together. Thereafter, deep reinforcement learning has been applied to several areas such as robotics, games (see, e.g., Mnih *et al.*, 2013; Lillicrap *et al.*,

2015; Silver *et al.*, 2016) and finance (see, e.g., Jiang, Xu and Liang, 2017; Liang *et al.*, 2018). This thesis is an attempt to apply modern machine learning methods to enhance traditional portfolio optimization methods.

1.2. The focus of this research

This study deals with the possibilities of deep reinforcement learning as a tool for active portfolio management in the stock market. The focus is on creating a deep reinforcement learning agent able to manage a stock portfolio in the New York Stock Exchange, in order to improve the return versus risk trade-off (measured here with Sharpe-ratio) and to beat the classical passive portfolio management methods. The study is conducted by embracing the findings of (Jiang, Xu and Liang, 2017; Moody *et al.*, 1998) and synthesizing their findings in order to show the somewhat unexplainable performance of neural networks in handling the almost randomly fluctuating market data. Figure 1 shows the research placement in the field of science.

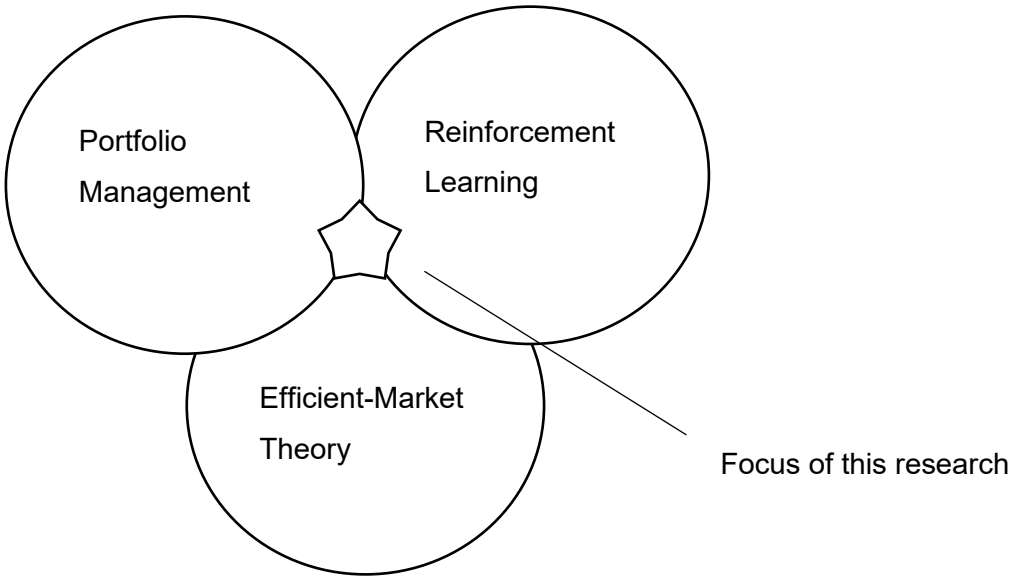


Figure 1 The research placement in the field of science

The research is mainly based on the theories of reinforcement learning and the Modern portfolio theory. Modern portfolio theory and its lack of practical usefulness creates a motivation for the study. The latest studies from reinforcement learning field suggest that, reinforcement learning might be a potential tool to improve applicability of Modern portfolio theory on a practical level, which is investigated in this thesis.

The previous research efforts concerning the topic are implemented with very small datasets, primarily intended to demonstrate the superiority of newly generated model structures. Also, they

mostly rely on the technical aspect in the trading, which is expanded in this study with the fundamental aspects. The study draws from the best practices found in the previous literature to create a trader agent and to adapt it to freely operate in a large size market environment in order to self-generate the most profitable trading behaviour as possible. After the agent's performance has been demonstrated in practice, its self-generated trading behaviour is analysed and examined, what are the reasons for the agent to be interested about a certain stock. The trader agent provided in the study may work as a potential tool for active investors in managing their portfolios and maximizing their risk-return ratio.

1.3. Objectives of this research and research questions

The objective of this study is to create a deep reinforcement learning agent, able to independently construct and manage a portfolio of stocks by analysing the daily trading data and the fundaments of public companies belonging to the S&P 500 index. Stock markets are generally considered to be highly unpredictable, but the recent studies (see, e.g., Liang *et al.*, 2018) back up the performance of reinforcement learning models to learn useful factors from the market data to be used in the portfolio management. Risk-adjusted returns measured with Sharpe ratio during the test period are used as a performance measure for the agent's actions and thus, the main research is represented as follows:

"Are the Deep Reinforcement Learning models competent to increase the risk-adjusted returns of a stock portfolio in the New York Stock Exchange?

The agent's performance is statistically tested by generating 5000 random portfolios with similar risk level as the test portfolio and comparing their Sharpe distribution to the test portfolio. The model is considered to statistically significantly outrun the random portfolios if its Sharpe ratio surpasses the average of the random portfolios by more than two standard deviations. Thus, the null hypothesis for the study is as follows:

H0: "Portfolio constructed with Deep Reinforcement Learning model does not statistically significantly outrun the random portfolios in risk-return ratio."

If the null hypothesis is rejected, the deep reinforcement learning agent is competent to increase the risk-adjusted returns of the portfolio and from this part the answer for the research question is very clear. If the null hypothesis is not rejected, the ability of Deep Reinforcement Learning models in the stock portfolio optimization can be placed under suspicion and the outcome of the study is not in a line with the previous literature.

Surpassing the random portfolios in the risk-adjusted returns proves that the model has learned useful factors from the data. However, the strong financial aspect in the research demands testing the agent performance with respect to the actual market performance, since the model will be useful in the financial markets only after it is able to surpass the market index in the risk-adjusted returns. Thus, another research question is as follows:

"Is the agent able to raise the portfolio performance over the market index?"

The research question can be answered after clarifying the statistical significance of the alpha generated by the agent with respect to the market index. The null hypothesis is as follows:

H0: "Positive alpha of the portfolio returns compared to market index is not statistically significant."

1.4. Outline of the thesis

The thesis is organized as follows: the second chapter covers the key parts of deep and reinforcement learning areas required to perform the study. The third chapter covers the relevant theories concerning the study and previous literature about the portfolio management with deep and deep reinforcement learning methods. In the fourth chapter, the dataset for the study is described and the research framework with the methods is introduced. In the fifth chapter, the agent is trained and optimized and finally tested with the test data. Thereafter agent's trading behaviour is analysed. The final chapter presents the conclusions and discussion for the study and possible future research topics.

2. THEORETICAL BACKGROUND

This chapter covers the key parts of the deep and reinforcement learning areas used to perform the study. The concept of Deep Learning is defined, and the structures of basic feed-forward, convolutional and recurrent neural networks are represented. The reinforcement learning framework is then fully illustrated and finally the combination of these two machine learning subsets, Deep Reinforcement Learning will be qualified. The overview for the methods covered in the chapter can be seen in the Figure 2.

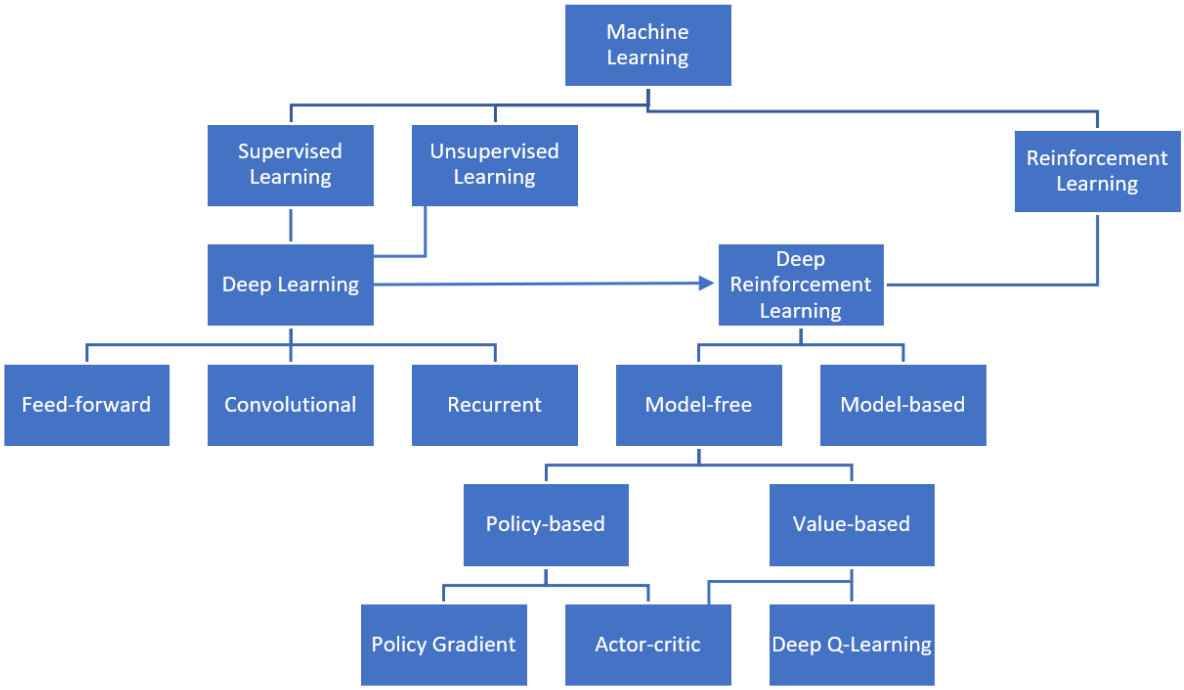


Figure 2. Machine Learning framework

2.1. Deep Learning

Deep learning (DL) is a subset of machine learning field, that is used to model high-level abstractions in data with neural network inspired computational models. Deep learning models, deep artificial neural networks, allow computational models composed of multiple processing layers to represent complex structures in data (LeCun, Bengio and Hinton, 2015). The term deep learning refers to the multi-layer artificial neural network models in contrast to the shallow learning models such as logistic and linear regression (Li, 2018). Artificial neural networks (ANN) are by their name inspired by the actual neural systems and are alike constructed of interconnected neuron units. The idea of neural system-based models is not a new, although DL has been in a central of machine learning

environment only for a while. Neural system-based models were proposed already in 1940s, but only the recent breakthroughs in ANN research and the increased computing power have allowed ANNs to come to a key position in the modern machine learning and artificial intelligence fields.

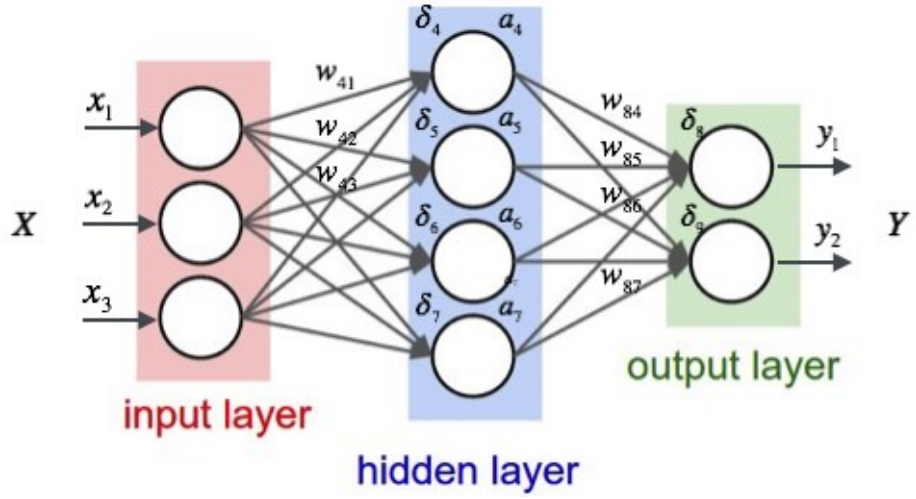


Figure 3. Structure of a feed-forward neural network with one hidden layer (Valkov 2017)

Figure 3 shows the structure of a basic feed-forward ANN with one hidden layer. ANN consists of multiple neuron units, that are connected together with weights and separated to different layers. Structure of a single neuron can be represented mathematically with equation 1,

$$Output = f\left(\sum_i (w_i \times h_i) + b\right) \qquad (1)$$

where w represents the weights, h represents the input values, b represents the bias term and f represents the activation function. The outputs from the previous layer are multiplied with the corresponding weights and then added together with the bias term and placed to the activation function, which defines the actual output.

ANNs are trained with *backpropagation algorithm*. It traces the error term back to the neuron units by calculating the partial derivative of the cost function with respect to the neuron weights and adjusts them in order to minimize the cost function. Partial derivatives are calculated through layers by exploiting the chain rule. If function x depends on the function y, which again depends on the function z, then the partial derivative of function x with respect to the function z can be solved with equation 2,

$$\frac{\partial x}{\partial z} = \frac{\partial x}{\partial y} * \frac{\partial y}{\partial z} \qquad (2)$$

2.1.1. Activation functions

The role of the activation function is to break the linearity of the dataflow, which is critical to the model functionality (Ramachandran, Zoph and Le, 2017). Without activation function, the model would only correspond to multiple stacked linear regressions and thus be unable to learn complex and non-linear structures from the data.

Figure 4 shows four different commonly used activation functions in ANNs. The most successful and commonly used activation function in deep neural networks is the *Rectified Linear Unit* (ReLU) (Nair and Hinton, 2010; Glorot, Bordes and Bengio, 2011). The main advantages by using ReLU against the other commonly used activation functions is training speed and its ability to answer to the vanishing gradient problem commonly faced with deep neural networks (Glorot, Bordes and Bengio, 2011). Derivate of the ReLU function $f(x) = \max(0,x)$ is 0, when x < 0 and 1, when x > 0. This extremely accelerates the gradient backpropagation algorithm. Other activation functions such as sigmoid and tanh suffer from the vanishing gradient problem, which is caused by the neuron output value saturating to the lower or higher border of the output range. Thus, the gradient, the derivate of the neuron weights with respect to the loss function, will draw very close to zero, which prevents the weights adjusting properly to the right direction.

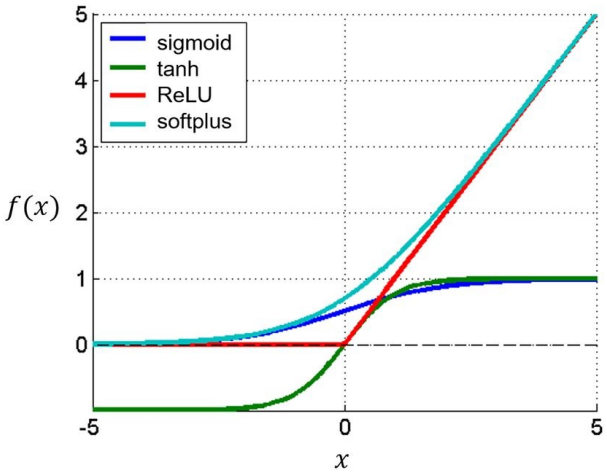


Figure 4 Different activation functions (Musiol 2016)

While the basic activation functions are enough for most of the cases, special type of models such as multi-class classifiers require of using special type of activation functions for the final layer. *Soft-max* activation function produces n-length vector containing weights or probabilities for n-amount of classes being 1 in total. Softmax is produced by equation 3:

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{n} e^{x_j}} \quad i = 0,1,2 \dots n \qquad (3)$$

2.1.2. Convolutional layer

Convolutional neural networks (CNN) are a class of artificial neural networks primarily used to detect patterns from the images. In 2012, Krizhevsky and others released their ImageNet, based on CNNs that revolutionized the image classification. Besides of image classficiation CNNs also have applications in such areas as natural language processing (Kalchbrenner, Grefenstette and Blunsom, 2014) and pattern recognition from financial time series data (Jiang, Xu and Liang, 2017).

In the convolutional layer, convolution operation is performed for the input matrix, in which the filter is used to map the activations from one layer to another. The filter is a matrix of the same dimension as the input, but with a smaller spatial extent. The dot product between all the weights in the filter and the same size spatial region of the input is performed at each location of the input matrix. (Aggarwal, 2018, 41) The products are placed to the output matrix, which is referred as a feature map. Figure 5 shows the structure of the 2-dimensional convolutional layer. It receives a 2-dimensional matrix as an input. The filter matrix slides over the input and at each location, the dot product is performed between the input matrix and the filter, which is then placed to the feature map.

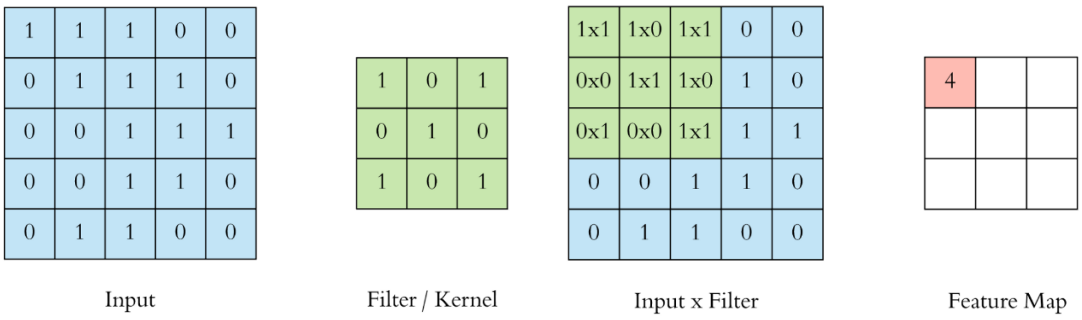


Figure 5 Structure of a 2-dimensional convolutional layer (Dertat, 2017)

Convolutional layers can be stacked on top of each other in order to detect more complex patterns from the data. The ImageNet by Krizhevsky and others (2012) contains five convolutional layers and millions of parameters to classify different images. Here, the features in the lower-level layers capture primitive shapes such as lines, while the higher-level layers integrate them to detect more complex patterns (Aggarwal, 2018, 41). This allows the network to detect highly complex features only from raw pixels and separate them to different classes.

2.1.3. Recurrent layer

Recurrent neural networks (RNN) are a class of artificial neural networks designed to model sequential data structures like text sentences and time series data (Aggarwal, 2018, 38). The idea behind the recurrent networks is that the sequent observations are dependent on each other and thus, the next value in the series depends on the several previous observations. Figure 6 shows a structure of a recurrent neural network. Sequential data is being fed to the network and for every observation the network generates an output and an initial state that affects to the next output. Recurrent neural networks have been generally used effectively in speech recognition (Graves, Mohamed and Hinton, 2013) as well as in time series prediction (Giles, Lawrence and Tsoi, 2001).

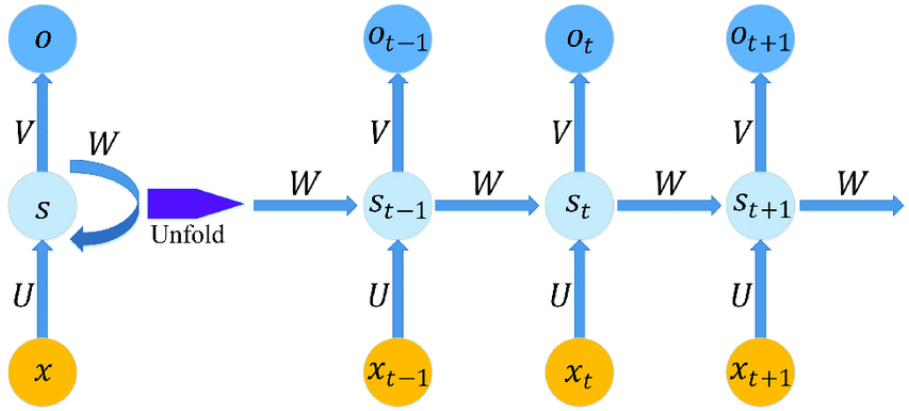


Figure 6 Structure of a recurrent neural network (Bao, Yue & Rao, 2017)

2.2. Reinforcement Learning

Reinforcement learning (RL) is considered as a reward driven process of trial and error, in which a system learns to interact with an environment in order to achieve maximal rewarding outcomes (Aggarwal, 2018, 374).

Machine learning framework is divided into three sections based on the learning process of algorithms. Supervised learning is learning from a training set with correct labels corresponding to the input features. The objective for the system is to generalize its responses in order to correctly manage the unseen observations (Sutton and Barto, 2018, 2). Unsupervised learning is used to detect patterns and clusters from an unlabelled dataset (Louridas and Ebert, 2016). Reinforcement learning differs from both sections since the learner is not provided with external knowledge about the correct acts. Instead, it is given rewards to encourage the desired behaviour, which is defined by the supervisor. The goal is to maximize the expected rewards over time, in which case the learner will achieve the global optimum. The process of trial and error is driven by this agenda, since the optimal behaviour can be learned only by exploring unprecedented actions. This type of

experience-driven training process is highly related to the psychology and to the learning mechanism of living beings in nature.

According to Sutton and Barto (2018, 13), history of the reinforcement learning can be separated into two different threads. Above-mentioned learning by trial and error was combined with optimal control theory in 1980s, which led to the emergence of the modern type of reinforcement learning. Optimal control theory deals with the problem of finding an optimal policy for a given system to maximize the measure of performance or to minimize the cost function (Stengel, 1994, 1). The system might be an economic, physical or social process, while the solution can be derived mathematically with optimization methods such as dynamic programming.

Dynamic programming was developed in 1950s by Richard Bellman and it refers to simplifying a complex problem by breaking it down into a sequence of simple decisions. This converts the problem into a *Markov decision process* (MDP), which will be defined more specifically in the next section. MDP consists of a set of different states, between which one can transfer through certain actions. By resolving the optimal actions for all the possible states, the optimal solution for the problem, the optimal policy, can be achieved (Bellman, 1954). The optimal behaviour can be derived simply by estimating the optimal value function v_* or the optimal action value function q_* for existing states with Bellman equations. A policy which satisfies the Bellman optimality equation $v_*(s)$ (equation 4) or $q_*(s,a)$ (equation 5) is considered as an optimal policy (Sutton and Barto, 2018, 73).

$$v_*(s) = \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \quad (4)$$

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a\right] = \sum_{s',r} p(s', r|s, a)\left[r + \gamma \max_{a'} q_*(s', a')\right] \quad (5)$$

where s represents the current state, s' represents the next state, a represents the action in the current state, a' represents the action at the next state, r represents the reward, p represents the probability and γ represents the discount rate.

Thus, the value functions for the optimal policy represent the expected future reward in the current state, when following the optimal policy, and more specifically, the sum of the possible future reward scenarios multiplied with their probabilities, when following the optimal policy at each state. The expected future rewards are represented as the sum of the next reward and the discounted estimate of the next successive state value. Estimating the values based on the other estimates, generally referred as bootstrapping, allows the optimal value function to be estimated iteratively, when the probabilities of the reward and state transitions are fully known. The estimation is done by assigning the expected future reward to the value function at the given state and repeating the

process comprehensively at the existing state space until the value function stops converging towards the global optimum. However, according to Sutton and Barto (2018, 67), the global optimum is very rarely achieved in practice due to the extreme computational cost.

2.2.1. Markov decision process

Sutton and Barto (2018, 47) define the *Markov decision process* (MDP) as a classical formalization of decision making, where performed actions influence besides of immediate rewards, also to subsequent situations and through those to future rewards. MDPs are used to mathematically represent the decision-making problem in order to derive the optimal solution with methods such as dynamic programming and reinforcement learning. A Markov decision process can be defined with the transition function, which is a 4-tuple (S, A, P_a, R_a), where S represents the finite set of states, A represents the finite set of actions from the state S, P_a represents the probability of action A at the state S leading to the state S' and R_a represents the immediate reward after the transition from the state S to the state S'. The key presumption of the MDP is that the current state includes all information about the past that is relevant for the future, which is known as a Markov property. Markov decision process centralizes around the concepts of the agent and the environment. The agent is the decision maker, that interacts with the environment and is given rewards based on the selected actions at the specific states. The environment contains everything the agent observes and interacts with.

Figure 7 represents the two-way interaction between the agent and the environment in the Markov decision process. At the time t the agent lies at the state S_t. Based on the agent's current policy it performs the action A_t corresponding to the state S_t.The agent transfers to the next state S_{t+1} and gets the reward R_{t+1} from the environment. The policy of the agent is adjusted iteratively based on the received rewards in order to maximize the expected future rewards for all the states, and thus to achieve the optimal policy.

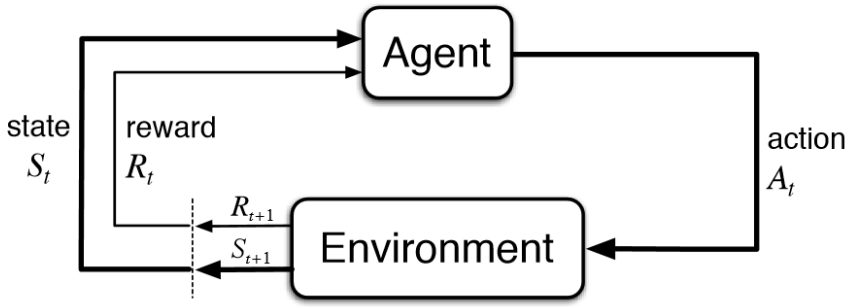


Figure 7. Interaction between the agent and environment in a Markov Decision Process. (Sutton & Barto 2018, 48)

Dynamic programming assumes a perfect model of the environment, in which case the different state transitions are fully known. This makes the dynamic programming unavailing for the MDPs with large and continuous state spaces, since the computational power is limited, and the computational expenses are growing exponentially when the state space increases. Complex MDPs can be optimized with different reinforcement learning algorithms that exploit function approximation, in which case the full model of the environment is not necessary.

2.2.2. Deep Reinforcement Learning

Deep reinforcement learning (DRL) is an area of reinforcement learning, where deep neural networks are used in function approximation to represent the value and policy functions. Deep learning models fit very well to the reinforcement learning problems due to their ability to approximate any function as stated by Hornik (1991). Although neural networks had been used before to solve reinforcement learning problems (Williams, 1992; Tesauro, 1995), the actual breakthrough happened in 2013, when Mnih *and others* released their deep alternative for the regular Q-learning algorithm. They used deep neural networks to approximate the Q-function for seven Atari 2600 games and achieved a superhuman performance in a major part. Afterwards, DRL has overtaken the reinforcement learning area relatively thoroughly and the new studies in the research field are principally focused on the deep learning systems.

2.2.3. Reinforcement learning algorithms

Reinforcement learning field encompasses different sorts of algorithms to optimize the policy in the Markov decision process. Preferred algorithms typically depend on the type of the environment. Different algorithms can be separated on the model-based and model free algorithms as well as the value-based and policy-based algorithms (Li, 2018). Reinforcement learning algorithms presented in this chapter are demonstrated in Figure 8.

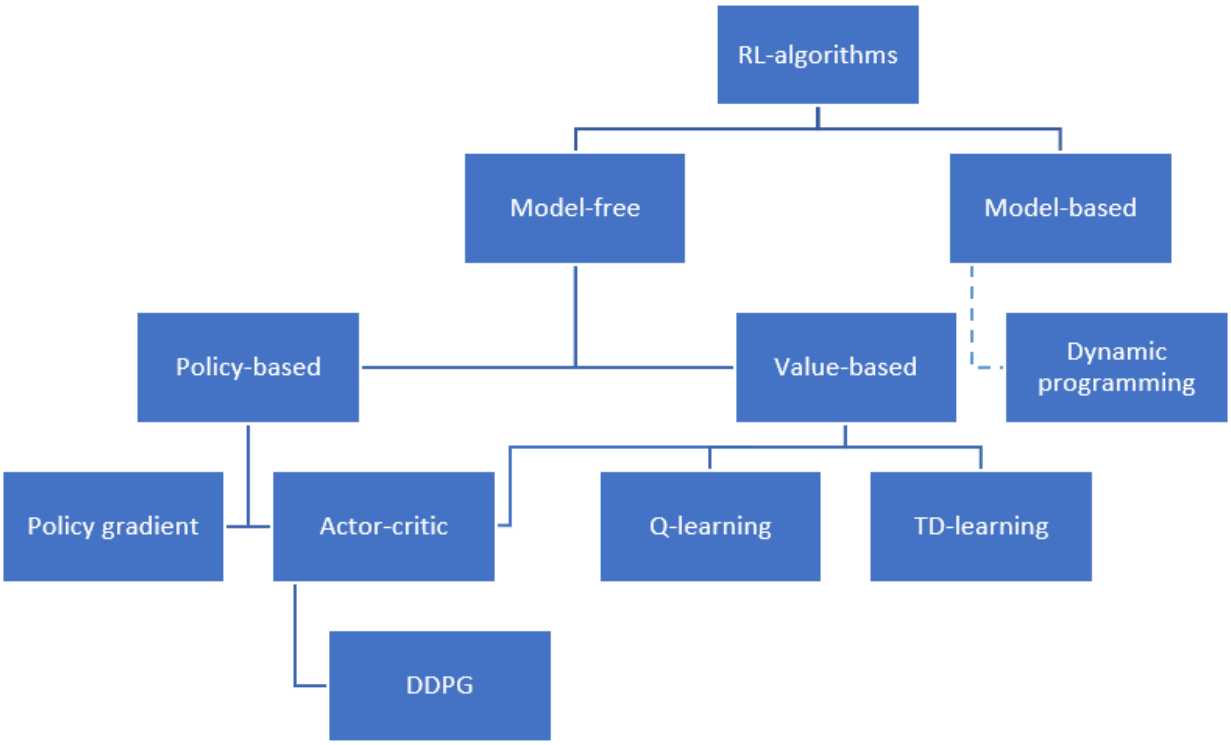


Figure 8 Reinforcement learning algorithms introduced in this chapter. Dynamic programming presented with dashed line, since it is not an actual RL algorithm

Model-based RL algorithms learn the transition function (S, A, P_a, R_a) of the environment in order to estimate the optimal policy and value function. Given the state and action, model-based methods predict the next state and reward. Model-based methods rely on planning as their primary component, which refers to searching the optimal path through the state space, when the transition function from state to another is known (Sutton and Barto, 2018, 160-161). According to Li (2018), model-based methods are data-efficient, although they may suffer from performance issues due to the inaccurate model identification.

Model-free RL algorithms rely on learning the optimal policies by estimating the expected rewards for the given states and actions via trial and error process. The algorithm does not have a knowledge about how the action A taken at the state S affects on the environment. (Li, 2018) Instead, the algorithm learns the optimal actions to perform at the specific states based on the received rewards, and step by step converges towards the global optimum. The benefit of the model-free algorithms is the computational efficiency since the complex model is not needed to learn for the whole environment. According to Sutton and Barto (2018, 12), the model-free algorithms usually are less challenging to train and may have an advantage against the model-based algorithms, when the environment is very complicated.

Value function is a prediction of the expected discounted future rewards, used to measure the goodness of each state or state-action pair (Li, 2018). Value-based algorithms estimate the value function for the model, which is used to derive the optimal actions. After the optimal value function is known, the optimal policy is achieved, when the action leading to the highest expected reward is selected in each state. *Temporal Difference* (TD) *Learning* by Sutton (1988) and its extension *Q-Learning* by Watkins & Dayan (1992) are examples of typical value-based methods. The methods are highly related to the dynamic programming, since the value functions are estimated similarly with bootstrapping method. The main difference to the dynamic programming is that TD-learning and Q-learning do not assume the fully known model of the environment (Sutton and Barto, 2018, 119). This enormously reduces the computational costs and allows the value function to be estimated only based on the experience of the state transitions. The value functions are estimated with the following update rules (equation 6 and equation 7) derived from the Bellman equation:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \quad (6)$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \quad (7)$$

Thus, the value V of the state S_t is updated after the transition to the state S_{t+1} based on the received reward R_{t+1} and the value of the next state discounted with the factor γ. Q-value of the state S_t and action A_t is updated after the transition to the state S_{t+1} based on the received reward and the Q-value of the optimal action a in the state S_{t+1}

Policy-based algorithms directly utilize the received rewards to estimate the optimal policy for the agent. Learning the value function is not needed, which may facilitate the learning process in the highly complex and large environments. According to Li (2018), policy-based methods are effective in high-dimensional and continuous action spaces and are able to learn stochastic policies. However, policy-gradient methods suffer from high variance, which slows down the learning process and they generally tend to converge only to local optima (Sutton *et al.*, 2000). Vanilla Policy-gradient algorithm can be implemented by computing the gradient of the expected reward with respect to the policy parameters (equation 8).

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta log\pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t})\right) \quad (8)$$

$$\theta_{t+1} = \theta + \alpha\nabla_\theta J(\theta)$$

Thus, the gradient of the expected reward J with respect to the policy parameters θ, equals to mean of the sums of log probabilities of the actions a, when following policy π at the state s multiplied with the sum of the received rewards r. Then, the update is done by assigning the gradient multiplied with the learning rate α to the policy parameters.

The so-called actor-critic algorithms such as Deep deterministic policy gradient (DDPG) (Lillicrap *et al.*, 2015), combine the benefits of the value- and policy-based methods. Actor-critic algorithms learn a value function, which is used to learn the optimal policy function by evaluating each state-action pair separately and computing their gradients with respect to the policy parameters. Separating state-action pairs based on the goodness is beneficial, since the gradients are now computed individually for different transitions. The method is generally seen to reduce variance and accelerate learning process compared to the policy gradients (Sutton and Barto, 2018, 331), while it also allows functioning with continuous action spaces.

Generally, when comparing reinforcement learning algorithms, the difference of their manner in policy approximation is also recognized. On-policy algorithms approximate the optimal policy by improving the policy that is used to make the decisions, whereas off-policy algorithms approximate a policy different from that used to generate the data (Sutton and Barto, 2018, 100). Thus, vanilla policy gradient is considered as an on-policy algorithm, since it exploits the received rewards to improve the current policy. Q-learning instead is considered as on off-policy algorithm, since it estimates the optimal state-action pairs separately and approximates the optimal policy by always selecting the optimal action in any given state.

2.3. Modern Portfolio Theory

Modern portfolio theory (MPT) originally based on the findings by Harry Markowitz (1952) lays a foundation for the research field in the portfolio management. The mean-variance analysis of the MPT creates a framework for this study, in which the research results will be appraised, but which also generates the primary justification for the research.

Modern portfolio theory suggests constructing a portfolio of assets such that the expected return is maximized for a given level of risk. The key assumption states that the risk-return ratio of the portfolio can be maximized by selecting the proper set of assets and by allocating the wealth among them in a right proportion. The expected return for the asset corresponds to the historical mean of the daily returns and the variance of the daily returns is used as a proxy for the asset risk. With the covariance between the assets, the optimal distribution of wealth that maximizes the expected return for the selected risk level can be determined. The optimal portfolios for the different risk levels form an efficient frontier:

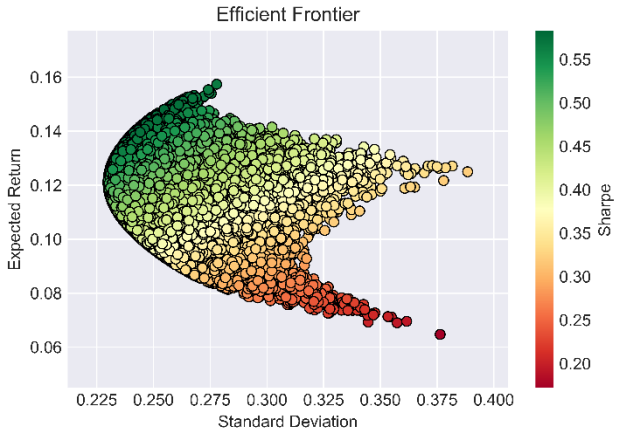


Figure 9. Efficient frontier for a five-stock portfolio from S&P-500 index list

Figure 9 presents the efficient frontier constructed for a five randomly selected stocks and shows how the Sharpe ratio representing the risk and return is maximized near the efficient frontier.

Modern portfolio theory considers the investor as a risk-averse and rational actor, that would accept higher risk level only in case of higher expected return. Thus, the investor would always select the portfolio from the efficient frontier, which practically would lead to the diversified portfolio. The intention behind the mean-variance analysis is the idea of diversification not only among the assets with high expected returns, but also with low cross-correlations. In practice, this kind of diversification adds up to selecting investments from different industries, in which case a failure in the certain industry would not affect to the whole portfolio. The suggestion was innovative at the time, when the most remarkable publications such as *The Intelligent Investor* (1949) by Benjamin Graham focused mainly on the fundamental analysis and individual undervalued stock picking. Modern portfolio theory is still one of the central theories of finance, which underpins the significance of Markowitz's efforts.

2.3.1. Criticism towards Modern Portfolio Theory

The critics of MPT ask, why the expected returns derived from the historical means would be relevant in the portfolio selection if the theory of market efficiency by Fama (1970) assumes that the future returns are independent from the past? Although it would be an incorrect conclusion to state that there exists a conflict between these two theories, since efficient market theory denies only the earnings of excess returns, it is widely questioned whether the expected return for the company could be derived from the historical information. Past returns reflect the historical development of the company and could have been affected by the incidents that do not occur in the future. Also, the concept of

expected return in the highly dynamic modern markets could be inapplicable, since the companies and consequently the expected returns are constantly progressing. To derive the mathematical mean from the volatile past returns requires using long enough time frame, during which the actual expected return and the expected risk will probably change. The historical average would thus represent only the historical conditions and not be applicable to the future situations. For further discussion, an interested reader should refer to Kasten and Swisher (2005).

The above-described problem is illustrated in practice in Figure 10. It shows a set of portfolios constructed with the train set of this thesis and then plotted again based on their performance with the test set. The colour of the points represents the Sharpe ratio implied by the train set performance, blue representing the efficient portfolios at the certain risk-levels. The plots show that the efficient portfolios implied by the train set performance are actually quite inefficient with the test set, while neither the other portfolios correlate in performance with the train and test sets.

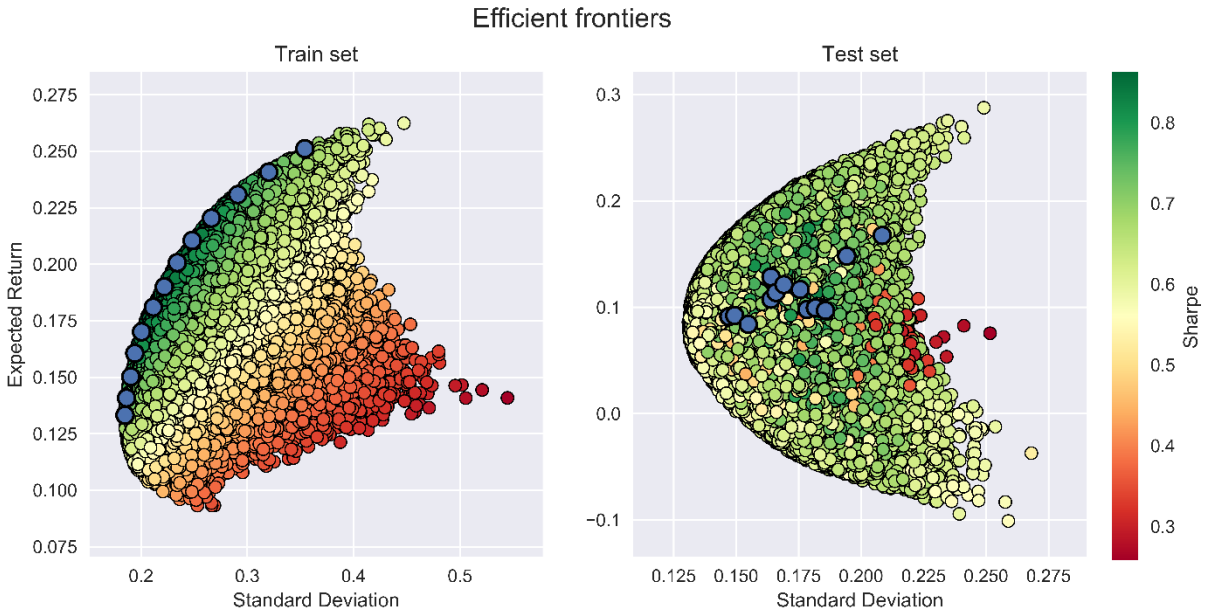


Figure 10. Efficient portfolios constructed with train and validation sets and tested with the test set of this thesis

2.4. Efficient Market Hypothesis

Theory of market efficiency suggests that the asset prices always fully reflect the available information in the markets. While the weak terms suggest the irrelevance of past returns, the semi-strong terms assume insignificance of publicly available information such as fundaments in the active portfolio management. (Fama, 1970) With these terms prevailing in the market, active portfolio management without private information would be considered ineffective and useless in the pursuit

of excess returns. Excess returns in this case are considered as risk-adjusted returns that exceed the market return after risk adjustment.

Despite the theoretical indications of the Modern portfolio theory and the efficient market hypothesis, portfolio managers have always sought for abnormal returns with active asset allocation and stock selection. No clear consensus has been obtained about the existence of the market efficiency, since studies have provided divergent results. Grinblatt and Titman investigated the performance of actively managed mutual funds in 1989 and concluded that the superior performance might in fact exist. However, the abnormal returns would be unattainable for the investors, since the high expenses of the funds neutralized the abnormal returns. Abarbanell and Bushee (1998) examined the power of fundamental analysis yielding significant abnormal returns and their findings indicated the effectiveness of the fundamental investment strategy in a one year time span. Blume, Easley and O'hara (1994) investigated a pure technical analysis and the role of volume in the trading process and drew positive conclusions about the informativeness of the past price and volume sequences. These findings by Abarbanell & Bushee (1998) and Blume and others (1994) are contradictory even to the weak terms of the efficient market hypothesis and speak for behalf of active portfolio management.

2.4.1. Anomalies

The inexplicable power of both the fundamental and technical analysis can be viewed as an anomaly in the stock markets. Schwert (2002) defines the anomaly as an empirical result being inconsistent with maintained theories of asset-pricing behaviour and indicating either the market inefficiency or the inadequacies in the underlying asset-pricing model. Various studies have reported about existing anomalies in the markets such as Jegadeesh and Titman (1993), who found the so-called momentum effect to exist in the New York Stock Exchange over the years from 1965 to 1989. Buying the past winners and selling the past losers generated abnormal returns, which could not be explained with systematic risk or delayed stock price reactions to common factors. Banz (1981) showed that small-capitalization firms in the New York Stock Market earned significantly higher average returns than large firms over the period from 1936 to 1977. Basu (1983) noted that firms with high earnings to price ratio (E/P) in the New York Stock Exchange implied higher risk-adjusted returns than firms with low E/P in the period of 1962 to 1978. Fama and French (1988) showed that over the period from 1927 to 1986, dividend yield predicted subsequent stock returns. Exploiting the detected anomalies should in theory lead to excess returns, but as Schwert (2002) states, history has taught that anomalies tend to vanish after their publication. Thus, it is a relevant question if some of the anomalies were just a statistical coincidence or did they vanish after the large masses tried to exploit them in their investments.

3. LITERATURE REVIEW

This chapter covers the relevant studies regarding the portfolio management problem with deep reinforcement learning algorithms. The theory is approached from two directions by going through the relevant classical portfolio selection methods of the Modern portfolio theory and then introducing the main breakthroughs in the field of machine learning that have finally led to the recent deep reinforcement learning applications in the portfolio management. The relevant deep learning, reinforcement learning and deep reinforcement learning solutions for the portfolio management will be identified and classified in terms of their approach method to the problem and then reviewed.

3.1. Classical methods

Various portfolio management strategies have been developed over the years. While some of them rely on the pure mathematics, such as the mean-variance analysis, the others focus on exploiting existing market anomalies. Different active and passive portfolio management and asset allocation strategies have been investigated in the history with varying success. Jorion (1991) compared the performance of classical asset allocation methods in active stock portfolio management. His results show that over the period from 1931 to 1987, the mean-variance optimization strategy led to very poor results with out-of-sample data and better results were achieved with minimum variance portfolios and CAPM portfolios, although their performance was still inferior to passive buy-and-hold strategy. DeMiguel, Garlappi and Uppal (2007) evaluated the out-of-sample performance of mean-variance portfolios and compared them, with naively diversified 1/N portfolios in the US equity market. They found the estimation errors with the optimized portfolios so high, that none of them consistently outperformed the naive 1/N portfolio in risk-adjusted return. The results are in line with Jorion (1991) and speak against the usefulness of mean-variance analysis on a practical level.

The performance of the classical optimization methods in active asset allocation has not been flattering. While several studies speak on the behalf of the fundamental analysis in the portfolio selection, the actual portfolio optimization methods of the modern portfolio theory have showed fairly poor performance on a practical level. The findings are somewhat unexpected, while the *Modern portfolio theory* remains as a one of the key financial theories. A question arises if there exist methods to improve the out-of-sample performance of the optimization methods. Studies such as DeMiguel *et al.* (2007) have introduced improvements to the classical mean-variance analysis, but over the last decades, more complex methods from the machine learning field have taken space in the portfolio management literature. The next section goes through the main breakthroughs that have ended up to the modern deep and reinforcement learning solutions in the portfolio optimization.

3.2. Deep Reinforcement Learning for Portfolio Management

Although, the largest breakthroughs in deep reinforcement learning have happened in the recent years, some early success with artificial neural networks and reinforcement learning have been obtained already in 1995, when Tesauro released his impressive backgammon playing algorithm *TD-gammon* based on temporal difference learning and neural networks. TD-gammon played backgammon at expert level despite from its tiny network size compared to modern successful DRL algorithms. Applications similar to TD-gammon were not published until 2013, when Mnih *et al.* introduced their work about the Atari playing agents based on the Q-learning with deep neural networks. With only raw pixels as input, the agents surpassed all the previous approaches on six different Atari games and achieved a super-human performance in three of them. The work was a clear breakthrough in reinforcement learning and started a new deep reinforcement learning era in the field.

After 2013, a vast number of new DRL applications have been introduced. Deep reinforcement learning has been successfully applied to multiple applications especially in games and robotics. Lillicrap *and others* (2015) introduced a model-free actor-critic algorithm *Deep Deterministic Policy Gradient* for control in continuous action spaces. Their algorithm was innovative, since the previous DRL applications handled only discrete and low-dimensional action spaces whereas their algorithm was able to solve several simulated physical control tasks such as car driving. In 2016, a program called *AlphaGo* was developed by the team of *Google DeepMind*. It was based on the deep reinforcement learning and was able to defeat the European champion in the classical Japanese board game “Go”. The achievement was of special importance, since Go had been previously considered as extremely hard for machines to master and thought to be at least a decade away. (Silver *et al.*, 2016)

In this thesis, we are interested in DRL algorithms in the financial applications and more specifically in portfolio management where DRL algorithms have already provided promising results. Deep and reinforcement learning applications in the asset allocation can be generally separated on the prediction-based and portfolio optimization-based solutions. Predictive models try to forecast near-future fluctuations in the asset prices to exploit them in the stock picking, while portfolio optimization models rather focus on the optimal portfolio selection originally inspired by the MPT. Instead of directly predicting the market movements, they aim to select the set of assets that will most likely perform better than average in the near future.

Before the reinforcement learning techniques existed, classical dynamic programming methods were applied to portfolio management with varying success. Brennan, Schwartz and Lagnado (1997) handled the asset allocation problem as a Markov decision process depending on the state variables; risk-free rate, long-term bond rate and dividend yield of the stock portfolio. The optimal strategy to

allocate wealth among the asset classes was estimated based on the empirical data, and the out-of-sample simulations provided evidence for the viability of the strategy. Neuneier (1996) adapted dynamic programming to asset allocation by using an artificial exchange rate as a state variable and found dynamic programming method to perform equivalently to the Q-learning at the German stock market.

The dynamic programming methods tested in both studies are very simple, which is partly due to the lack of general computation power to estimate more complex models. However, the main reason is the inability of dynamic programming to solve problems in large state spaces, which is referred as the "*Curse of dimensionality" by* Bellman (1954). Required computational power increases exponentially when the state space grows, which prevents even the modern computers from estimating the value function dynamically for a problem with large state space. To handle factors in the portfolio management, reinforcement learning algorithms have surpassed the dynamical methods in popularity.

Moody *et al.* (1998) used recurrent reinforcement learning for asset allocation in order to maximize future risk-adjusted returns for a portfolio. They proposed an interesting approach to the reward function: a differential Sharpe ratio that can be maximized with a policy gradient algorithm. Its performance was compared with running and moving average Sharpe ratios in asset allocation among the S&P 500 index and T-Bill during a 25-year test period. The differential Sharpe ratio was found to outperform running and moving average Sharpe ratios in the out-of-sample performance, while also enabling an on-line optimization. All the performance functions were able to significantly improve the out-of-sample Sharpe ratio in contrast to buy-and-hold strategy. Later, Moody and Saffel (2001) extended their previous studies and made further investigation about their direct policy optimization method now referred to as Direct Reinforcement. Besides of the differential Sharpe ratio they introduced a differential downside ratio to better separate undesirable downside risk from the preferred upside risk. They compared the direct reinforcement method to Q-learning and found it to outperform the value-based Q-learning method during the test period.

Findings by Moody *et al.* (1998) and Moody & Saffel (2001) have later inspired several studies about direct reinforcement such as Lu (2017) and Almahdi & Yang (2017). Lu (2017) used policy gradient algorithm to maximize differential Sharpe ratio and *Downside deviation ratio* with *Long short-term memory* (LSTM) neural networks to implement forex trading. The agent successfully built up downside protection against the exchange rate fluctuations. Almahdi and Yang (2017) instead extended the direct reinforcement to five-asset allocation problem and tested *Calmar ratio* as a performance function. Calmar ratio like Sharpe ratio compares the average returns of portfolio to the risk measure but replaces the portfolio standard deviation with the maximum drawdown of the portfolio. Calmar ratio compared to Sharpe ratio was able to increase the portfolio performance significantly.

Value-based methods for portfolio management have been implemented by Lee *and others* (2007). They used a multiagent approach with cooperative Q-learning agent framework to carry out stock selection and pricing decisions in the Korean stock market. The framework consisted of two signal agents and two order agents in charge of executing the buy and sell actions. Signal agents analysed the historical price changes represented in a binary turning point matrix to define the optimal days to execute the buy and sell orders. Order agents instead, analysed an intraday data and technical indicators to define the optimal prices to place the buy and sell orders. The agents consisted of Q-networks to predict the Q-values for discrete actions in the different states. The framework was able to outperform the other tested asset-allocation strategies and to efficiently exploit the historical and intraday price information in the stock selection and order execution. Their solution was a quite innovative for a time, since Deep Q-learning experienced the rise in popularity only several years later and only a few prominent Q-network applications had been published at that time such as *TD-gammon* (Tesauro, 1995).

Although, price fluctuations are hard or near to impossible to predict, it does not mean that they happen randomly or for no reason. As Fama (1970) stated, prices reflect the available information and thus, the price fluctuations are due to the new information in the market. Being able to exploit this information better than competitors should then generate abnormal returns if it is assumed that the price adjustments do not happen immediately. Ding *and others* (2015) embraced this factor and took a completely different approach to asset. They used deep convolutional networks to extract events from the financial news and used the information to predict the long and short-term price movements. Their model showed significant increase in S&P 500 index and individual stock prediction abilities. Although the results are very interesting, this type of asset allocation would be more profitable to implement with intraday data. If it is suggested that the price adjustments do not happen immediately and if the model exploits new information without delay, the profitability of the model should in theory raise significantly.

Nelson, Pereira and De Oliveira (2017) treated portfolio management as a supervised learning problem and used Long short-term memory networks to predict the direction of the near future price fluctuations with stocks. Their model was able to achieve 55,9% accuracy for the movement direction predictions. The common problem with this type of studies is that they generally do not publish the actual prediction behaviour of the model. Since stock prices tend to increase more often than decrease, it usually results of the model predicting only the more likely class, in which case the accuracy of 56 percent might not be so astonishing after all.

Jiang, Xu and Liang (2017) successfully adapted deep reinforcement learning to the cryptocurrency portfolio management. They proposed the method of *Ensemble of Identical Independent Evaluators* (EIIE) to inspect the history of separate cryptocurrencies and evaluate their potential growth for the

immediate future. EIIE is a model-free and policy-based approach to the portfolio management consisting of a neural network with shared parameters to evaluate separately but identically the set of assets based on their historical price information. The model is fed with a series of historical price data to determine the portfolio weights for the given set of cryptocurrencies and it is trained to maximize cumulative returns with policy gradient algorithm. The advantage of the method is its structure of shared parameters among the separate crypto coins, which allows the model to scale to a large set of assets, without notable increase in computational costs. The method was tested separately with convolutional, recurrent and long short-term memory networks and the convolutional neural networks were found to perform the best with the given dataset. Despite from high 0.25% transaction costs, the model achieved 4-fold returns during the 50-days test period.

Liang *and others* (2018) adopted the EIIE structure proposed by Jiang, Xu and Liang (2017) and adapted it to the China stock market. Instead of regular CNNs, they developed the model with deep residual CNNs introduced by He *et al.* (2015), which allow deeper network structures by utilizing short-cuts for the dataflow between non-sequential layers. Liang *and others* (2018) compared the policy gradient algorithm with off-policy actor-critic method *Deep Deterministic Policy Gradient* (Lillicrap *et al.* 2015) and with policy-based *Proximal Policy Optimization* (Schulman *et al.*, 2017) and found the policy gradient to be more desirable in the financial markets although the other methods are more advanced in general. The returns achieved by their implementation in the China stock market were not as astonishing as with the previous studies, although they found the policy gradient method to outperform the benchmarks in a 5% risk-level. The study was conducted with quite small dataset of only five assets randomly chosen from the China stock market, which was somewhat unreasonable, since the EIIE structure allows the model scaling efficiently to much larger datasets.

Previous literature suggests reinforcement learning and deep reinforcement learning to be a potential tool in the portfolio management. However, in the latest paper by Liang et. al (2018) the performance of the models was significantly lower than in the previous studies. This may possibly be affected by the fact that the use of reinforcement learning and other machine learning models in stock trading and portfolio management has increased due to the growth of deep learning and reinforcement learning fields.

Dynamic programming methods suffer generally from the Bellman's curse of dimensionality, when the state space is very large and complex. Also, the lack of knowledge about the full model of the stock market environment prevents the efficient dynamic optimization of the value function and asset allocation policy. The answer for the problem is a function approximation with neural networks that deep reinforcement learning generally exploits, but he earliest studies of reinforcement learning in portfolio management are generally suffering from similar restrictions. The lack of computational power has limited the scope of the existing studies and thus they tend to focus on very simple

allocation problems with very few assets at a time. Later, the increased computational abilities and the progress in deep and reinforcement learning fields have opened the doors for more realistic research layouts with larger datasets and more complex models. However, learning a value function to represent the different market situations seems to be a quite complex task even for the complex DRL algorithms in portfolio management. This makes the direct policy optimization methods as the preferred method for the portfolio management, since they seem to more easily learn the valuable signals from the extremely noisy market data and thus to better generalize to unseen market situations.

4. DATA AND METHODOLOGY

In this chapter, the construction of the dataset for the research is presented and the prepared dataset is statistically described. Then the research setting is described based on the theoretical background of the study. Finally, the model structure used in the research is introduced and the statistical methods to test the model performance are presented.

S&P 500 constituents were selected for the dataset due to their high liquidity in the markets. Using only high liquidity stocks minimizes the price slippage in transactions and ensures that the model cannot unrealistically exploit price fluctuations caused by individual trades. The dataset scale is significantly larger than in the previous studies to test the scalability of the trader agent and to find out if larger amount of companies positively affects on the model performance. Earnings to Price, Dividend Yield and trading volume were found useful for portfolio management in the literature review and were thus selected to the dataset to clarify their usefulness in daily trading.

Policy gradient algorithm was selected to optimize differential Sharpe ratio based on its performance in several studies (Moody *et al.*, 1998; Jiang, Xu and Liang, 2017; Liang *et al.*, 2018). Value-based methods were found not to successfully handle very noisy market data and thus the simpler policy gradient method was selected. The model structure is inspired by Jiang, Xu and Liang (2017) due to its structure of shared parameters that allows the model scalability to very high number of assets.

4.1. Data description

Besides of using purely historical price data as in the previous studies, a fundamental aspect for investing suggested by several research workers (Basu, 1983; Fama and French, 1988; Abarbanell and Bushee, 1998) is added to the study to see if Earnings to Price (EP) ratio and Dividend Yield (DY) are useful in improving the model performance. The structure of the trader agent is mostly based on the study by Jiang, Xu and Liang (2017) exploiting convolutional neural networks and using shared parameters in evaluating the different assets. The structure fits perfectly for the large-scale market data analysis due to it scalability permitted by the shared parameters.

The dataset contains 5283 daily observations for the constituents of S&P 500 stock market index at the beginning of 2013 covering 21 years in total from 1998 to the end of 2018. The original data contains historical total return indexes, closing prices, historical volume, quarterly earnings per share, the out paid dividends and the declaration dates for dividends as well as the publication dates for the quarterly reports.

Total return index is used to calculate the daily returns for each stock. The daily closing prices, quarterly earnings per share and the publication dates for the quarterly reports are used to form a daily Earning per Price ratio for each stock to represent the stock profitability based on the latest reports. The quarterly dividends with the daily closing prices and the declaration dates are used to construct the daily Dividend Yield for each stock. EP ratio and DY were selected for the features based on the findings about their stock return prediction abilities in the literature review. The data was mainly collected from Datastream -database, hosted by Thomson Reuters, while some of the missing datapoints were gathered from Nasdaq.com.

4.1.1. Cleaning

The data was splitted on the training, validation and test sets such that the training data covers 14 years in total from the 1998 to the end of 2011, the validation data covers 2 years from 2012 to 2013 and the test set covers five years from 2014 to 2018. Training data is used for training the agent, while its performance will be constantly evaluated with the validation data. Test data is used to test the actual performance of the final trained model. The dataset must be splitted in a chronological order to produce reliable results, since the actual trading actions are always implemented with the current data. Thus, splitting the data in non-chronological order would create a look-ahead-bias to the test set and lead to unrealistic results.

Several companies in the dataset have not been public from the start of the training period, and thus some of them lack an adequate amount of training data. Also, some of the companies suffer generally from a bad quality declaration and quarterly report date data and thus, 85 companies were removed from the final dataset. The final dataset contains 415 companies in total and daily observations for 5283 days. The daily observations include daily returns, daily trading volume in USD, daily Earnings per Price ratio and daily dividend yield.

4.1.2. Descriptive statistics

Table 1 shows the descriptive statistics for the training, validation and test sets. Training set shows clearly the highest volatility in returns, EP and DY, which is partially caused by the Dot-com bubble and the financial crisis during the training period, but also by the fact that the dataset was selected based on the companies belonging to the S&P 500 index in 2013. Many of the companies were still quite small in early 2000s, which is generally related to higher volatility. This arrangement also generates a survival bias to the training set, since it only includes companies that survived the both market crises and grew until 2013 to be large enough to belong to the index. This probably has a

negative impact on the model performance, but has no effect on the research reliability, since it only affects on the training and validation sets.

Table 1. Descriptive statistics for the datasets. Legend: R%=Total return, EP=Earnings to Price, DY%=Dividend Yield, Vol=Trading volume

Set	Training				Validation				Test			
Feature	R %	EP	DY %	Vol*	R %	EP	DY %	Vol*	R %	EP	DY %	Vol*
Mean	0,07	0,032	0,45	147786	0,10	0,050	0,52	210835	0,03	0,026	0,51	254372
Std Dev	2,75	0,241	0,61	350746	1,57	0,101	0,44	548287	1,71	0,227	0,42	501878
Min	-68,05	-29,142	0,00	0	-47,76	-3,338	0,00	3545	-39,34	-13,829	0,00	0
25%	-1,10	0,027	0,00	19535	-0,68	0,039	0,18	60524	-0,67	0,030	0,22	75553
50%	0,00	0,048	0,29	56770	0,07	0,054	0,48	108179	0,00	0,045	0,48	138093
75%	1,18	0,069	0,66	143314	0,87	0,072	0,76	204038	0,78	0,061	0,73	261848
Max	559,02	1,952	27,83	22884213	61,91	0,520	4,64	29875659	87,72	0,652	9,29	28581124

* in thousands (USD)

Figure 11 shows correlation plots for the features with the stock returns and the distribution of each feature. Features show no linear dependence with the daily returns. However, the dispersion of the returns seems to decrease while dividend yield and trading volume increases. Also, a highly negative Earnings to Price ratio seems to imply lower return dispersion than an EP ratio close to zero, which is an interesting observation. Distribution of features shows that all the features are clearly centred around their averages, although they seem to be highly dispersed based on the correlation plots. The connection with the highly negative EP ratio and the return dispersion may be caused by a single stock and seems irrelevant due to its rarity.

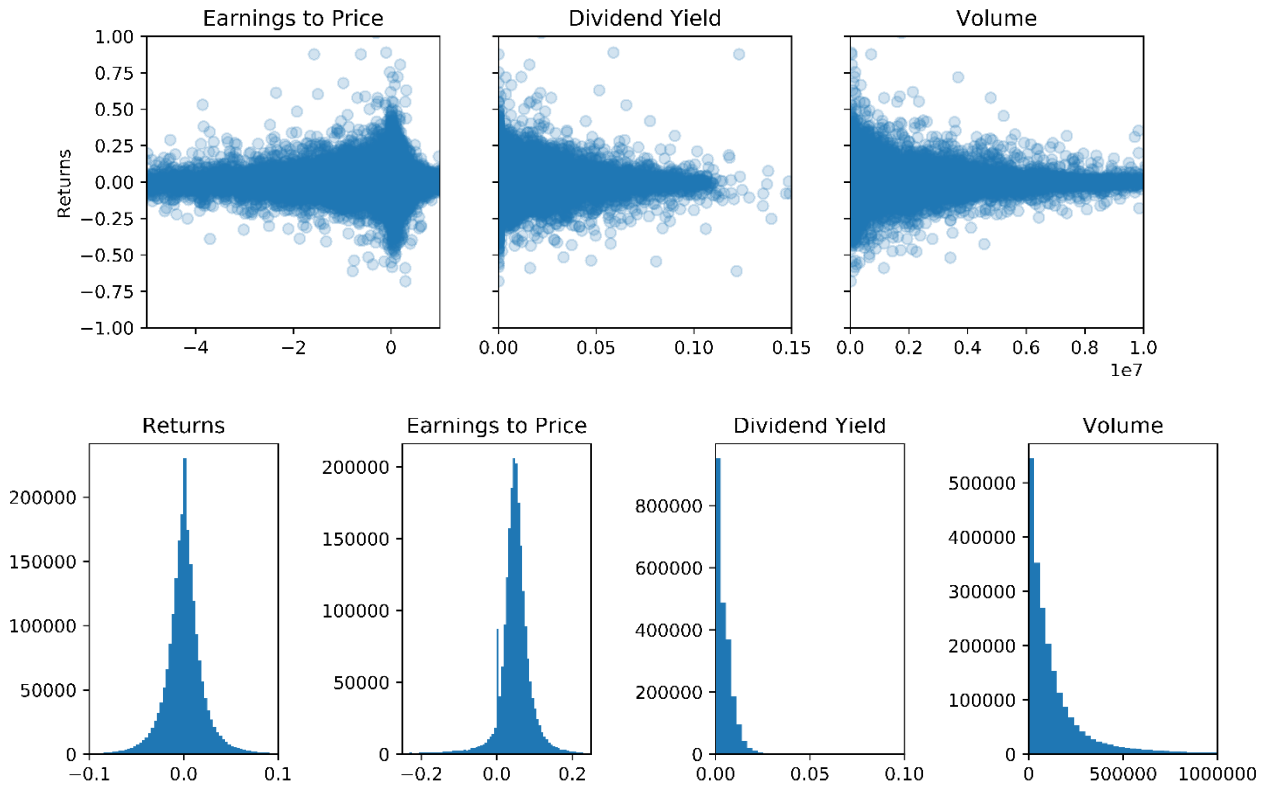


Figure 11. Feature histograms

Figure 12 shows the means and medians of the features among the observations through the observation period. During the financial crisis, EP median rises while, at the same time, EP mean dives(Fig 11a) implying that the market crisis affects dramatically only on the profitability of certain companies pulling down the average value. Dividend Yield (Fig 11b) also shows a dramatical increase during the financial crisis, which is due to the fact that stock prices plummeted in general although an individual company's performance or ability to pay dividends was not necessarily hampered. Trading volume increases through dataset (Fig 11d), which might affect negatively on the model performance.

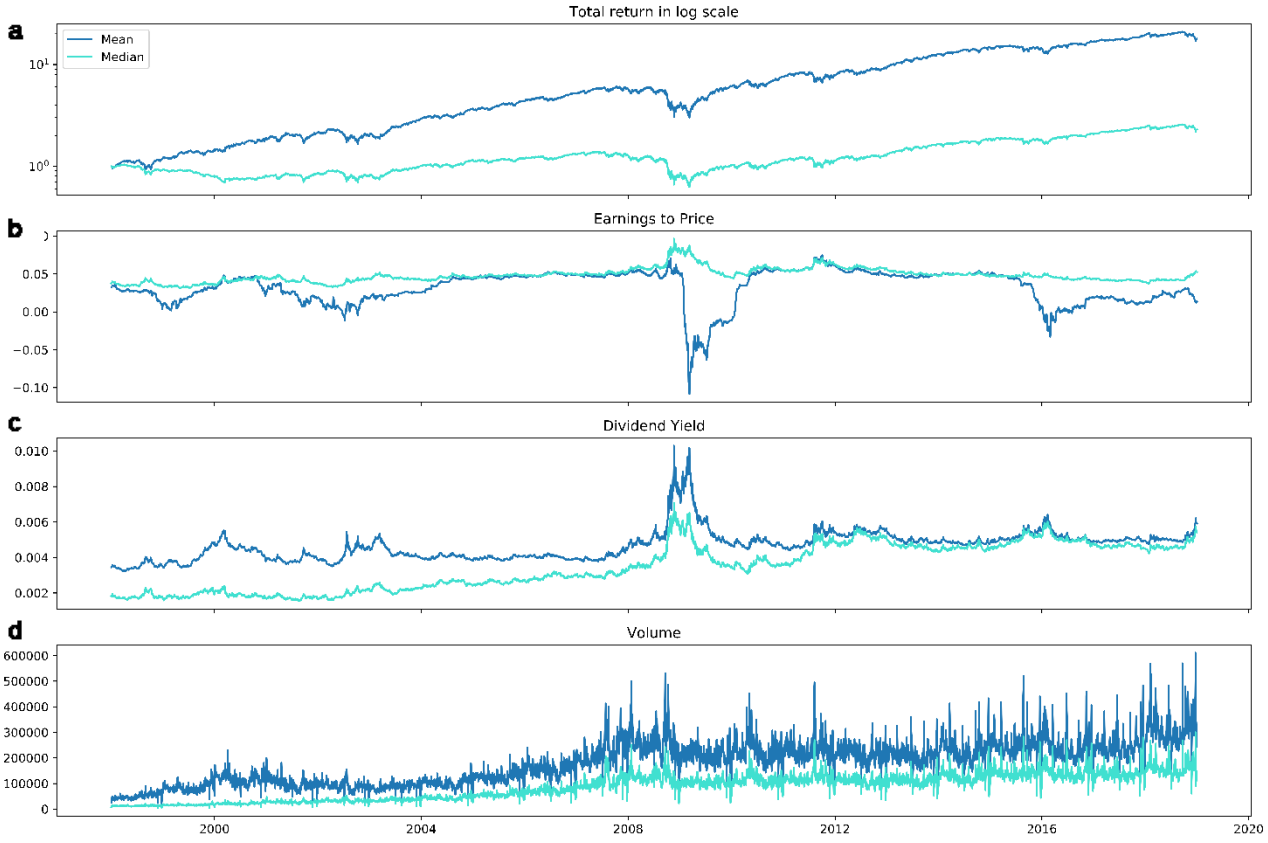


Figure 12. Means and medians for the features through dataset

4.2. Method description

The trading situation is defined as a Markov decision process, where the stock market represents the environment, in which the neural network model operates as an agent. The agent observes environment states in a form of fixed length time-series of stock prices and selected fundaments (EP and DY). Based on the observations it executes daily actions of picking a certain stock to the portfolio and defines its weight. Trader agent picks and manages a 20 stock portfolio with a cash asset among the 415 stocks in the environment and is trained to maximize the Sharpe ratio for the portfolio. Trader agent is trained with the 14-year training data from 1998 to 2011. Its performance and the parameter adjustments are validated based on the 2-year validation data from 2012 to 2013. The final agent performance is tested with the unseen 5-year test data from 2014 to 2018 and the agent is found to outperform the random portfolios as well as the S&P 500 market index in total and risk-adjusted returns.

Since the stock market, where the agent operates, is an extremely complex environment and affected by countless factors, such as other traders, all the relevant data can't be included to the

research making the environment only partially observable. Based on this limited set of observations from the environment, the agent executes daily actions that determine the portfolio weights for the next day in order to maximize the future risk-adjusted returns. This arrangement suggests that the stock portfolio can always be constructed with the exact set of weights. This does not fully apply in reality, since the prices of different stocks vary a lot preventing the construction of exactly weighted portfolios. In our model, this means that the agent is allowed to buy fractions of stocks in order the achieve the precise weights. Transaction costs are taken into a consideration as a daily fee that reduces the daily returns. The detailed calculation of transaction costs is presented in the next chapter.

The research is conducted with Python programming language using mainly *Tensorflow* library, which is an open source library for high performance numerical computation and machine learning developed by Google. Mathematical calculations in Tensorflow are handled with tensors, that are according to their definition *"a generalization of vectors and matrices to potentially higher dimensions"(*Tensorflow 2019b*).* While this definition might not be a fully correct representation, it is accurate enough to define the methods of this research mathematically.

4.2.1. Portfolio optimization model

For the purposes of portfolio optimization, we define a Markov decision process with a 4-tuple (S, A, P, R), where S is the set of states, A is the set of actions, P is the transition probability between the states and R is the reward for the action A in the state S. State S is represented with the input tensor s_t, which is a tensor of rank 3 with shape (d, n, f), where *d* is a fixed amount defining the length of the observation in days, *n* is the number of stocks in the environment and *f* is the amount of features for an individual stock.

Action A is represented with the action vector $a_t = \pi(s_t)$ with shape $(n + 1)$, where π is the current policy. It defines the portfolio weights for the full set of stocks and the cash asset. Thus, $\sum_{i=0}^{n} a_{i,t} = 1$.

Transition probability P is the probability that the action a_t at state s_t leads to the state $s_{t+1} = (x_{t+1}, w_{t+1})$. Basically, the transition probability function remains unknown for the portfolio management problem, since market data is very noisy, the environment is too complex and only partially observable. Reward R is defined with a future Sharpe ratio for a time interval $[t, t + T]$. Motivated by Moody *et al.* (1998), differential Sharpe ratio is used as a reward function, which is constructed in this study with the equation 9:

$$J_{[t,t+T]}(\pi_\theta) = R\big(s_t, \pi_\theta(s_t), \dots, s_{t+T}, \pi_\theta(s_{t+T})\big) = \frac{250}{T * \sqrt{250} * \sigma_{[t,t+T]}} \sum_{t}^{t+T} a_t \cdot (y_t + 1) \qquad (9)$$

$J_{[t,t+T]}(\pi_\theta)$ represents an expected finite-horizon reward for time period $[t, t+T]$, when following the policy π with parameters θ. It is derived from the reward R achieved from the state-action pairs during the time period. Differential Sharpe ratio is represented as a product of portfolio daily returns divided with the standard deviation $\sigma_{[t,t+T]}$ of the portfolio during the time interval $[t, t+T]$, where c represents the transaction cost percentage.

The optimal policy π^* is achieved when the expected reward $J(\pi_\theta)$ is maximized, which is done by computing the gradient of the expected reward with respect to the policy parameters $\nabla_\theta J(\pi_\theta)$ and updating the policy parameters with the equation 10 until the algorithm stops converging.

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J\big(\pi_{\theta_k}\big) \qquad (10)$$

where α is the learning rate.

Transaction costs are defined based on the commissions at the Interactive Brokers (2019), which is a popular broker for active trading. They charge a fixed commission fee of 0,005 USD per a traded stock. Average price of the stocks in the dataset is 60 USD so $\frac{0{,}005}{60} \sum_{i=0}^{n} \left| w_{i,t} - a_{i,t} \right|$ transaction cost is charged after each trade, where w_t is the current portfolio weight vector. Due to the price movements affecting to the portfolio weights, portfolio weight vector w_t is derived from the action vector a_{t-1} by equation 11:

$$w_t = \frac{a_{t-1} \times y_{t-1}}{a_{t-1} \cdot y_{t-1}} \qquad (11)$$

where y_{t-1} is the stock return vector at time $t-1$

4.2.2. Agent model

The model represents the agent's policy function π, which maps the state S into an action A. It is a policy-based and model-free solution, which maps the environment state tensor s_t to the action vector a_t. Model-free solution is used, since the stock market consists of countless factors affecting to the stock prices that are not included to the dataset and thus, modelling the entire environment would lead to a very imprecise solution. The model is trained with policy-based strategy direct reinforcement originally suggested by Moody and others (1998), learning the optimal policy directly from data without learning a value function, since it was found superior to value-based methods by

Liang *and others* (2018). The model consists of three convolutional layers and is significantly inspired by *Ensemble of Identical Independent Evaluators* presented by Jiang, Xu and Liang (2017). The first layer includes few small filters extracting very basic level patterns from the data. The second layer contains filters of same shape than the feature maps of layer one, thus producing feature maps of shape (1,). The third layer combines these feature maps as scalars to a vector length of n_stocks, which is placed to softmax activation function to define the weights for each stock. Softmax is used for the final layer to produce a vector whose sum is 1 so that the most attractive stock gets the highest weight and all the wealth is always fully invested to assets. The basic structure of the model is represented in the Figure 13 and the full structure of the model graph printed from Tensorflow is presented in appendix 2.

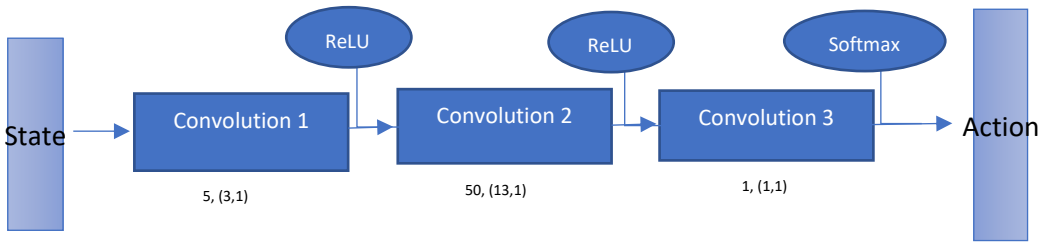


*Figure 13 Model structure. Filter amounts and shapes of the final model of this thesis presented in a form: **filter amount**, (**filter shape**)*

5. EMPIRICAL RESEARCH

In this chapter the model presented in chapter 4 is trained and optimized with the training and validation sets and afterwards the final model performance is tested with the test set.

5.1. Training and optimization

The model is trained with minibatches $(b = 50, d = 15, n = 20, f = 1 \dots 4)$ where *b* is the batch length of sequential observations, *d* is the length of observation days, *n* is the amount of stock selected randomly to the single minibatch and *f* is the amount of features used to train the model. The training is done in epochs of 500 minibatches and after each epoch the model performance is evaluated with the validation data. To improve the model performance, normal distributed noise with standard deviation of 0.001 is added to the training data before feeding the batch to the model similarly as (Liang *et al.*, 2018).

5.1.1. Feature selection

Training starts by evaluating the significance of different features for the model by training three different sized models with different feature combinations for 250 epochs. Model 1 represents the model structure suggested by Jiang, Xu and Liang (2017) for crypto markets. Since the dataset in this thesis is multiple times larger than above-mentioned, and it also contains stock fundaments instead of only price data, it is assumed that the optimal model will be more complex than the solution by Jiang, Xu and Liang (2017). Thus, Model 2 and Model 3 are created to find out if more complex models can better handle the stock fundament and trading volume data. In this part, the main focus is on the performance with different feature combinations within a single model and thus, the performance differences between the models are not very important. To achieve a balanced validation performance the model is validated with full portfolio of 415 stocks. It ensures that the portfolio performance does not fluctuate rapidly, since the stocks in hold are not changing continually. Table 2 shows the different model parameter combinations used to evaluate the features.

*Table 2 Model parameters in the format of **number of filters, (filter shape)***

	Layer 1	Layer 2	Layer 3
Model 1	2, (2,1)	20, (14,1)	1, (1,1)
Model 2	3, (2,1)	50, (14,1)	1, (1,1)
Model 3	5, (2,1)	100, (14,1)	1, (1,1)

Table 3 (also Figure 14) shows the performance of the different models with the validation set and can be seen that all the models are learning useful signals from the training data. The highest performance in terms of Sharpe ratio is achieved by using total returns and DY, but also EP improve the model performance slightly compared to using only total returns. Based on the findings, trading volume seems to be irrelevant for the model with this dataset on one-day trading frequency, since it impairs the model performance even with the most complex model structures compared to using only total returns. Thus, it is left out from the final model and the parameter optimization is done by using only Total return, Dividend Yield and Earnings to Price as the features.

Table 3 Sharpe ratios and total returns for different feature combinations

		[R%]	[R%, EP]	[R%, DY]	[R%, Vol]
Sharpe	Model 1	1,99	2,03	2,06	1,95
	Model 2	2,18	2,18	2,27	1,94
	Model 3	2,20	2,25	2,22	2,07
Total Return	Model 1	86,38 %	88,76 %	72,08 %	66,73 %
	Model 2	82,42 %	82,11 %	80,14 %	68,87 %
	Model 3	87,23 %	114,14 %	88,02 %	74,46 %

Another notable fact from the Figure 14 is that the validation performance with R% and EP feature combination is very unstable with the Model 1 and Model 3. It might be caused by too high learning rate in the training algorithm as well as by the model randomly overfitting to the validation data. This must be taken into a consideration during the optimization phase, since the model overfitting to the validation set would probably lead to a very poor performance with the actual test set. In general, the best performance with all the feature combinations and with the both performance metrics is achieved with the most complex Model 3 except for the for the R% and DY combination measured with Sharpe ratio in which the highest performance was achieved with the Model 2. This implies that with this dataset, a more complex model structure improves the performance and the parameter optimization is executed by paying an attention to this finding. Jiang, Xu and Liang (2017) in their implementation with cryptocurrencies, used a model structure that corresponds to the Model 1 in the complexity and which performed the worst in this test.

Figure 14 Validation performance with different feature combinations during training

5.1.2. Parameter optimization

The parameter optimization is executed by testing the model performance with different sets of hyperparameters and choosing the combination that achieves the highest stable validation performance measured with the Sharpe ratio and Total return with the validation set. Since, the possible amount of hyperparameter combinations is practically infinite, the optimization is started by randomly selecting similar parameter combinations that were found to perform the best in the feature

optimization phase. Hyperparameters optimized in this study are the number of convolutional layers, the convolutional filter shapes and amounts and the feature noise. The different models are trained for 300 epochs each and the model with highest stable performance will represent the final model. The problem of unbalanced validation performance is intended to minimize by using a rolling average validation performance of 10 consecutive epochs as a performance metric.

Table 4 shows hyperparameter sets and validation performances measured with maximum 10-day rolling average Sharpe ratios and Total returns. Models 3 and 6 achieved the best validation performances in terms of Sharpe ratio (2,40) and Model 5 achieved the highest total return. Figure 15 shows the validation performances during the training phase and can be seen that all the models perform similarly during the first epochs achieving approximately 1,7 Sharpe ratio and 55% returns with the validation data. This is due to the similar attributes of untrained models, since the allocate the wealth equally to all the assets. Model 6 validation performance was quite unstable during the training phase although it achieved the highest 10-day average performance. The most complex models, Model 9 and Model 10 started overfitting to the training set very early, which can be seen as the strongly decreasing validation performance. Since the training performance with Model 6 was quite unstable a small fine tuning for parameters could improve the performance. The most complex models with four layers in general achieved the poorest validation performance, which tells that the models will overfit to this dataset quite easily and the best results might be achieved with simpler models. Models 11-15 with parameters close to the Model 3, 5 and 6 are trained and is found that Model 14 achieves the highest 10-day average validation performance of all the trained models with both validation metrics. Thus, the parameters of model 14 are selected for the final model. Figure 16 shows the validation performances during the training phase.

*Table 4 Model parameters in the format of **number of filters, (filter shape)** and validation performances. Model 14 was selected as the final model version (highlighted)*

	Layer 1	Layer 2	Layer 3	Layer 4	Sharpe	Total Return
Model 1	3, (2,1)	50, (14,1)	1, (1,1)	-	2,15	80,85 %
Model 2	4, (2,1)	100, (14,1)	1, (1,1)	-	2,34	99,07 %
Model 3	3, (3,1)	50, (13,1)	1, (1,1)	-	2,25	105,44 %
Model 4	5, (2,1)	100, (14,1)	1, (1,1)	-	2,21	79,83 %
Model 5	5, (3,1)	70, (13,1)	1, (1,1)	-	2,11	115,84 %
Model 6	10, (2,1)	100, (14,1)	1, (1,1)	-	2,40	104,73 %
Model 7	20, (2,1)	200, (14,1)	1, (1,1)	-	2,26	88,84 %
Model 8	3, (1,1)	3, (2,1)	50, (14,1)	1, (1,1)	2,15	84,02 %
Model 9	3, (1,1)	4, (2,1)	100, (14,1)	1, (1,1)	2,14	69,54 %
Model 10	3, (2,1)	5, (2,1)	100, (13,1)	1, (1,1)	2,19	88,62 %
Model 11	8, (2,1)	80, (14,1)	1, (1,1)	-	2,24	101,04 %
Model 12	8, (3,1)	80, (13,1)	1, (1,1)	-	2,25	83,89 %
Model 13	5, (3,1)	100, (13,1)	1, (1,1)	-	2,32	99,23 %
Model 14	5, (3,1)	50, (13,1)	1, (1,1)	-	2,43	118,06 %
Model 15	10, (3,1)	100, (13,1)	1, (1,1)	-	2,26	79,95 %

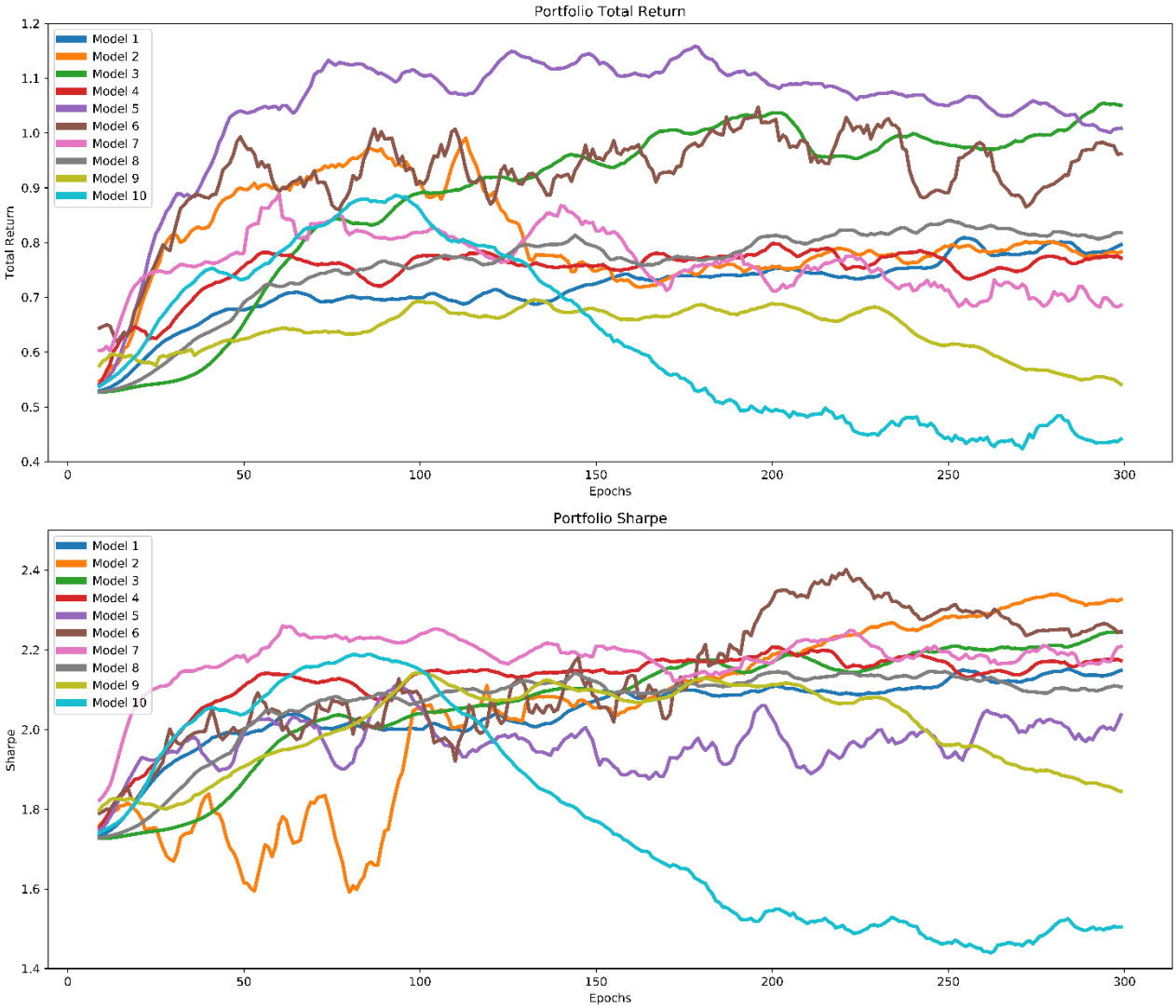


Figure 15 Validation performance of the first 10 models with the validation set during training process. (x-axis starts from epoch 10, because 10-day rolling average values are used as the performance metrics)

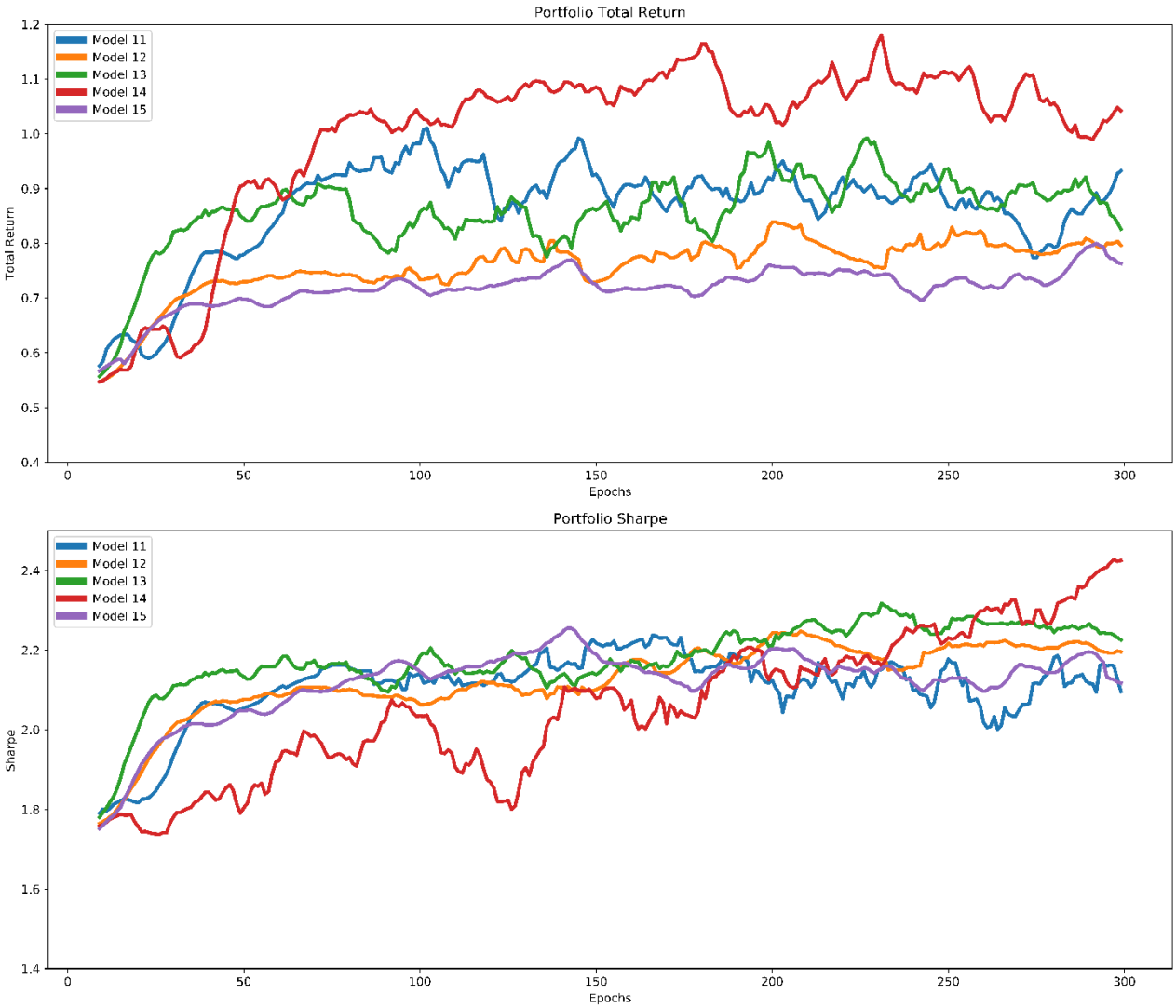


Figure 16 Validation performances for the models 11-15 with the validation set during training process. (x-axis starts from epoch 10, because 10-day rolling average values are used as the performance metrics)

The final model is trained three times for 400 epochs with similar parameters from scratch. To avoid overfitting, early stopping is used by saving the version possessing the highest validation performance measured with Sharpe ratio during the training process. The highest validation performance was achieved during the second run at the epoch 318 reaching 2.53 Sharpe ratio and 119% total return with the validation set. Figure 17 shows the validation performances for each run during training and Figure 18 shows the final trained model performance with the validation set.

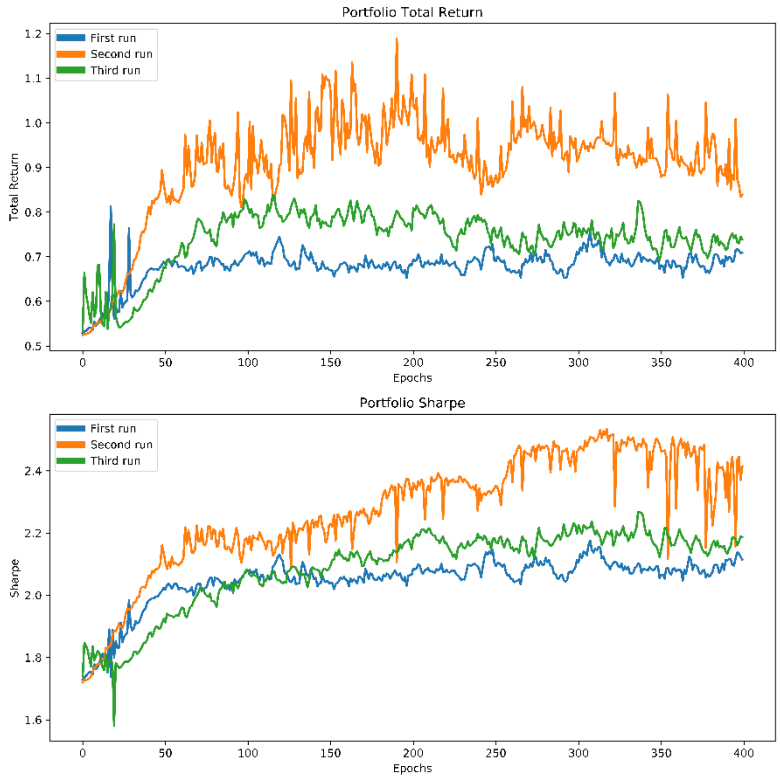


Figure 17 Final model validation performances during the different runs

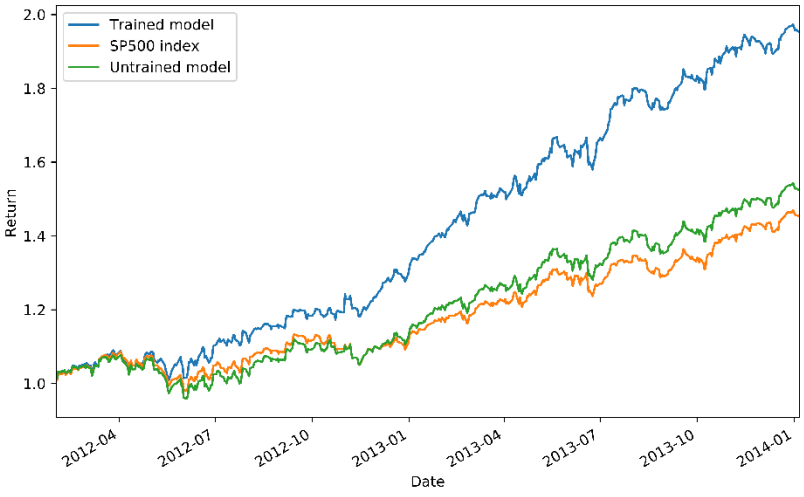


Figure 18 Final model performance with validation set without transaction costs

5.1.3. Testing statistical significance

The final model performance is first compared to the random weight portfolios with the test set to ensure the statistical significance of the model performance. The performance is compared with both Sharpe ratio and total return. The test is done by possessing a full 415 stock portfolio, which minimizes the effect of randomness affecting to the portfolio performance. The random portfolios are generated to hold similar risk profile than the trained model by allocating approximately 11 percent of wealth to the most attractive stock and approximately 95 percent of wealth to the 200 most attractive stocks. Figure 19 shows the performance of random weight portfolios with full set of 415 stocks compared to the final model performance. The model achieves Sharpe of 0.96 with the test set and the performance exceeds the average random portfolio performance by slightly more than two sigmas. Total return of the model during the test period is 370%, which exceeds the average of the random portfolios by over 16 sigmas. Thus, can be stated that the model performance is statistically significant.

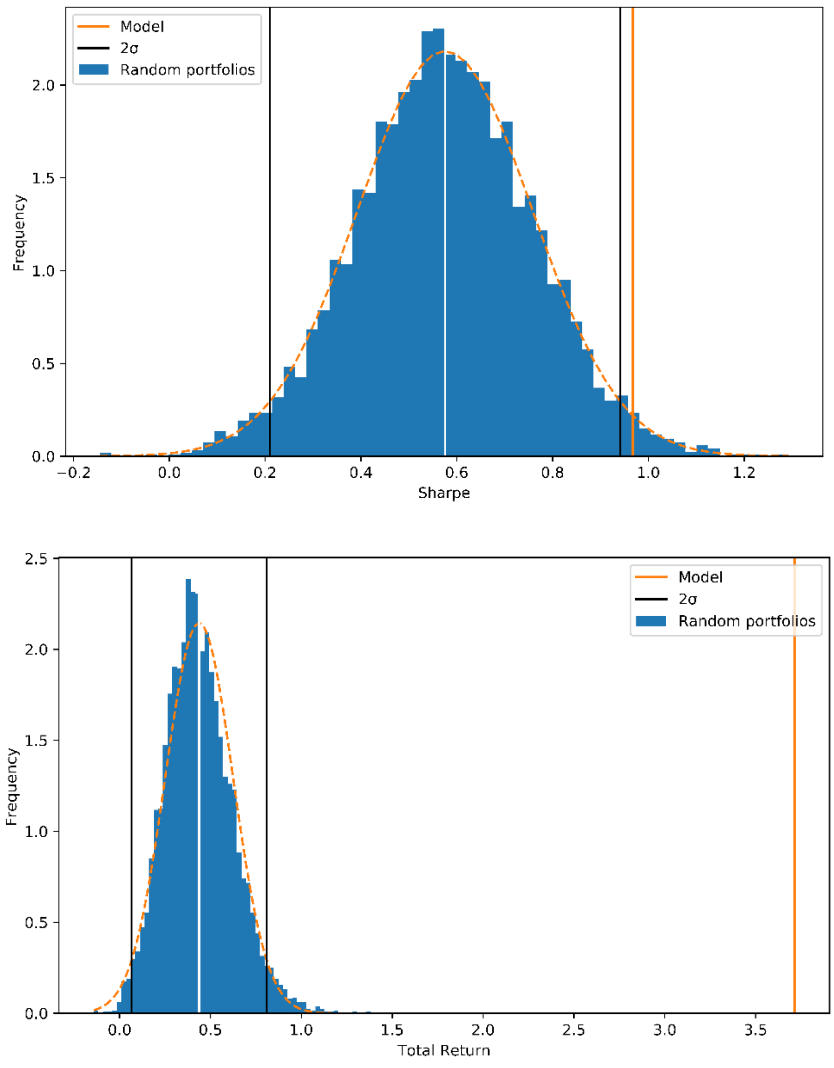


Figure 19 Frequency of Sharpe ratios and total returns of 5000 random portfolios compared to model performance

5.2. Model performance

To investigate the model profitability in practice, it is used to control a 20-stock portfolio throughout the test period. In this part, transaction costs for the portfolio are included. Model's performance is compared with SP500 index, optimal Markowitz's mean-variance portfolio and the best stock in terms of Sharpe ratio indicated by the pre-test data. Markowitz's mean-variance portfolio is created by selecting 20 stocks from the dataset with highest expected return discovered with the pre-test data and optimizing their weights by randomly generating 500 000 differently weighted portfolios and selecting the one with highest Sharpe ratio indicated by the pre-test data. The weights are used as starting weights for the portfolio, after which the weights are changing due to the difference in the stock returns. Optimal Markowitz's mean-variance portfolio created is showed in appendix 3.

Table 5 shows total return, Sharpe ratio and daily standard deviation for the model and the benchmarks. The model achieves distinctly the highest total return (328,9 %) for the test period. Although the total return for the model overwhelmingly exceeded the other benchmarks, its risk-adjusted performance measured with Sharpe ratio suggests only a slight superiority of the model. Daily standard deviation measures the riskiness of the portfolios and it can be noted that the portfolio controlled by the model actually possesses the highest risk of all the portfolios exceeding even the single-stock portfolio.

Table 5 Model performance with benchmarks

	Total Return	Sharpe	Daily StD
Model	328,9 %	0,91	2,9 %
SP500 index	54,9 %	0,73	0,8 %
Optimal portfolio	83,3 %	0,78	1,1 %
Best stock (Monster)	114,2 %	0,64	2,0 %

Figure 20 shows the performances of the model and the benchmark portfolios during the test period. During the first 21 months, the model fails to generate excess returns and actually loses in performance to all the benchmarks. However, during the next six months the model generates rapidly very high returns and is after that able to beat the market index, until the general market collapse appears, and the model suffers a severe breakdown in the returns. The effect of transaction costs to the model performance is represented in appendix 1.

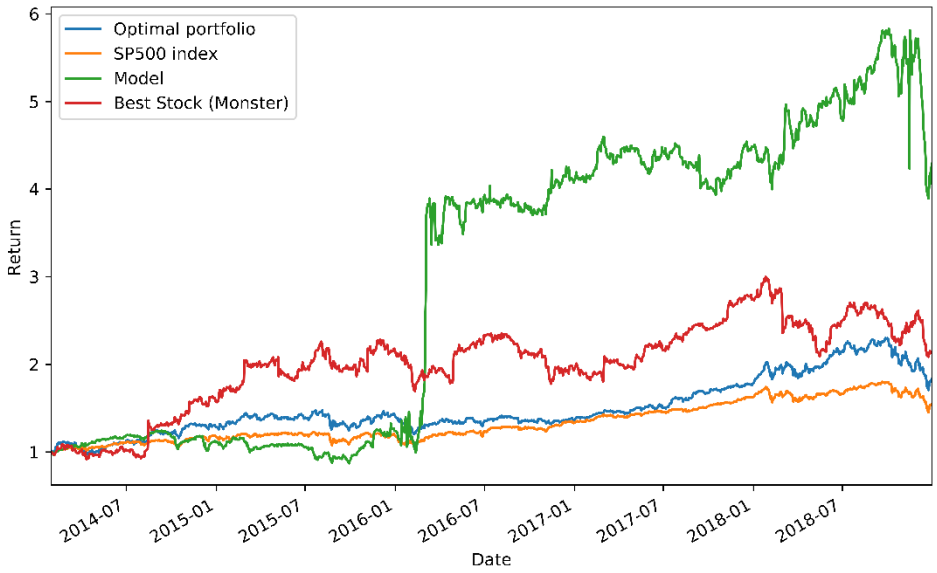


Figure 20 Performance of the model and the benchmarks during the test period

To test the model's actual ability to generate alpha against the market index, a regression to explain the model returns with the market returns is executed. Table 6 shows the results from the regressions with daily and monthly returns. Both regressions suggest a slight alpha for the model, which however is not statistically significant since the P-values from the t-Tests for both regressions exceed the 5% risk level. Thus, it cannot be stated that the alpha significantly differs from zero. Both regressions suggest a high beta for the model and the results are statistically significant, since the P-values from the t-Test are clearly below the 5% risk level.

Table 6 Regression of the 20-stock portfolio against the SP500 index

	Daily returns				Monthly returns			
	Coefficient	*Standard Error*	*t-Test*	*P-value*	*Coefficient*	*Standard Error*	*t-Test*	*P-value*
Alpha	0,00117	0,00076	1,52936	0,12643	0,01747	0,02823	0,61898	0,53827
Beta	1,28613	0,09129	14,08837	0,00000	3,03349	0,80152	3,78466	0,00036

5.3. Result analysis

In this section, the importance and the relevance of the results is discussed, and the trading strategy implemented by the DRL agent is examined.

The feature selection showed that the fundaments only slightly improved the model performance from the model tracking the only the total return. The models were learning useful factors from the

pure daily return data, which actually conflicts even the weakest terms of efficient market hypothesis introduced in chapter 2. The finding suggests that the stock prices might not after all follow a purely random walk, but instead, contain weak and often profitless, but still somehow learnable patterns in them.

Daily trading volume weakened the performance of all the models and thus was found ineffectual for the portfolio management. The result is not surprising with this dataset, since the average trading volume was shown to increase continuously from 1998 to 2018 at the data description part. The fact that a feature is continuously growing throughout the dataset seriously hampers neural networks from learning anything useful signals from them.

Parameter optimization part showed that despite from the parameters, the models are learning useful factors from the dataset at every run. This finding supports the supposition that the market behaviour and stock prices are actually learnable to a certain point and restrains the doubts about the pure randomness causing the positive performance of the model. Another fact that came up in the optimization part was the fairly low complexity of the highest performance models. The final model only contains three convolutional layers and a few feature maps including only 3375 parameters in total. For a comparison, ImageNet contains 5 layers and 60 million parameters in total (Krizhevsky, Sutskever and Hinton, 2012). The models with fourth layers started to overfit very early to the training data, which was noticeable as the early fall of the validation performance. Commonly, financial data is referred as a very complex to learn and financial markets as a very sophisticated environment in general. Whereas it might be true, it is noticeable that financial data was handled successfully in this study with remarkably simple model structures. Based on this study, financial data could be viewed as not very complex, but very noisy environment instead, and due to the noisiness, very hard to handle with supervised learning methods.

5.3.1. Observations on the agent behaviour

Few things can be noticed by examining the agent behaviour during the test period. The agent does not hold all the stocks equitably, but instead it favours certain stocks and holds them very often compared to an average stock. Table 7 shows the five most held assets by the agent during the test period. The most held stock during the test period was *Denbury Resources* that was held 12,7 times more than an average company. It was also the best-performed stock in the portfolio with 75,5 percent returns in total. The second most held stock is *Frontier Communications* being 11,7 times more favored than an average company. However, total returns with this stock were negative causing a -9,26% loss for the portfolio. The third most held stock *Chesapeake Energy* was also the second most profitable stock in the portfolio with 26,9% returns in total. It is noteworthy to mention that the agent took a very close to all-in position to these stocks several times

during the test period. The behaviour seems somewhat hazardous and will be analysed in the next section.

Table 7 The most held stocks during the test period. Legend: Holdings: Multiplier of stock held over the average. TR: Total return achieved with the stock. Max: Highest daily profit achieved. Min: Lowest daily profit achieved. Position: Highest position to the asset.

	Holdings	TR	Max	Min	Position
Chesapeake Energy	6,7	26,90 %	16,65 %	-10,61 %	0,9938
Denbury Resources	12,7	75,50 %	27,19 %	-23,20 %	0,9995
Frontier Communications	11,7	-9,26 %	6,28 %	-5,62 %	0,9965
Plum Creek Timber	3,3	0,18 %	0,15 %	-0,03 %	0,9910
Cash	4,0	0	0	0	0,0647

To better understand the agent behaviour, can be examined the states leading to the most aggressive actions by the agent. Table 8 shows three states leading to the most aggressive positions by the agent. By comparing these states can be noticed that the agent is obviously favouring very volatile stocks. It seems that very high positive and negative returns during the observation period are viewed as a positive signal by the agent. Another, and actually very surprising note from the states is that the agent is taking high positions to stocks with unnaturally low Earnings to Price ratios. Such low EP ratios can be achieved mostly after a very large extraordinary loss that makes the stock price to collapse so down that the loss exceeds the market value of the whole company several times.

Table 8 Three states leading to the highest positions during the test period

Day	R%	EP	DY	R%	EP	DY	R%	EP	DY
t-15	-8,60 %	-9,09	0,064	-3,96 %	-3,94	0	-9,53 %	-8,77	0
t-14	-3,30 %	-9,40	0,066	0,75 %	-3,91	0	-2,63 %	-9,01	0
t-13	12,09 %	-8,39	0,059	0,00 %	-3,91	0	-3,91 %	-9,38	0
t-12	11,22 %	-10,57	0,053	-3,72 %	-4,06	0	-6,87 %	-10,07	0
t-11	7,56 %	-9,82	0,049	-7,72 %	-4,40	0	-4,03 %	-10,49	0
t-10	8,60 %	-9,05	0,045	1,68 %	-4,33	0	-1,75 %	-10,68	0
t-9	28,05 %	-7,06	0,035	-0,83 %	-4,37	0	-3,56 %	-11,07	0
t-8	19,66 %	-5,90	0,030	1,25 %	-1,71	0	-2,58 %	-11,37	0
t-7	53,06 %	-3,86	0,019	-0,41 %	-1,72	0	-4,16 %	-11,86	0
t-6	12,27 %	-3,44	0,017	8,23 %	-1,59	0	-3,16 %	-12,25	0
t-5	-39,34 %	-5,66	0,028	0,38 %	-1,58	0	-6,94 %	-13,16	0
t-4	3,15 %	-5,49	0,028	8,33 %	-1,46	0	-1,32 %	-13,34	0
t-3	-3,49 %	-5,69	0,029	-0,35 %	-1,47	0	-1,78 %	-13,58	0
t-2	8,59 %	-5,24	0,026	0,35 %	-1,46	0	-1,81 %	-13,83	0
t-1	-7,08 %	-5,64	0,028	2,10 %	-1,43	0	5,99 %	-13,05	0
t	Position: 0,9988 Returns: -7,17 %			Position: 0,9995 Returns: 6,50 %			Position: 0,9965 Returns: -2,17 %		

The strategy implemented by the agent seems very interesting. By favouring the extremely low EP stocks the agent was able to highly outperform the stock index during the test period in total return. However, it is very clear that the strategy possesses a very high risk. This is confirmed for example by the fact that 2 of the three highest positions led to very high negative returns and in general the portfolio experienced very high positive and negative returns throughout the test period. The daily standard deviation of 3% tells that the agent in principle fails to manage risk in the portfolio management and mainly focuses on the very high returns while maximizing the Sharpe ratio.

The reasons leading to a such unusual strategy might be unexplainable, but one explaining factor could be the survival bias in the training set. Since the dataset includes companies belonging to the SP500 index at the 2013, the training set does not include companies that went to bankruptcy before 2013. Thus, the agent does not consider the extremely low EP ratio as a risk for the bankruptcy, but instead as a chance for very high positive returns in the future. In practice, the collapsing stock price after the high reported losses is caused by the highly increased risk level of the stock. However, the agent only observes companies during the training part that were able to get over the losses and to grow enough to be a constituent of SP500 index in 2013.

All in all, the agent behaviour seems a quite opportunistic, since it is able to beat the market only during certain periods by taking very high positions to single stocks. Managing a twenty-stock portfolio with this type of strategy might not be the optimal situation, since the transaction costs are impairing the portfolio returns, since the portfolio is continuously changing. The practicality of the agent could be improved by reducing the portfolio size to one stock and investing the rest of the wealth to the stock index. Table 9 and Figure 21 show that this strategy improves the overall performance of the agent by reducing the daily standard deviation and improving the total returns after the transaction costs.

Table 9 Performance metrics of the one-stock and twenty-stock portfolios

	Total Return	Sharpe	Daily StD
Twenty-Stock	328,9 %	0,91	2,9 %
One-Stock	367,6 %	0,98	2,5 %

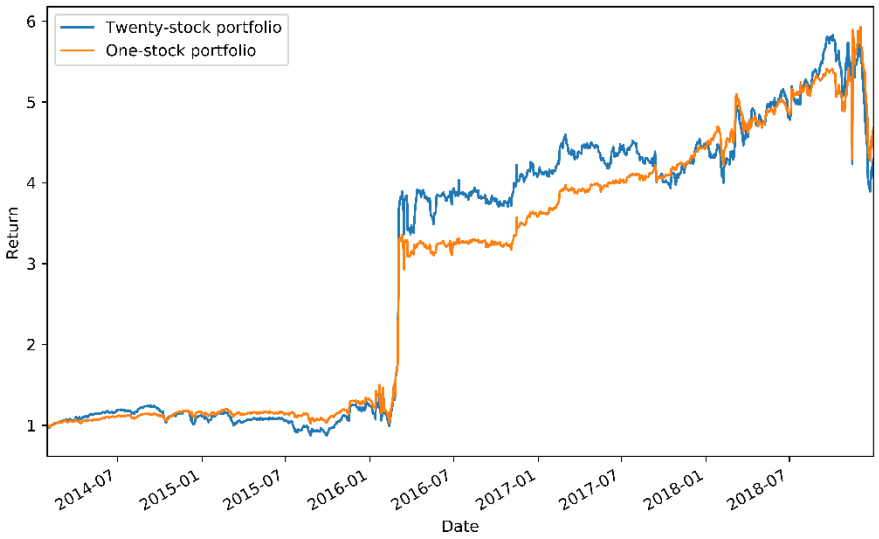


Figure 21 Comparison of twenty-stock and one-stock portfolios during the test period

5.4. Answering the research questions

The objective of the study was to create a deep reinforcement learning agent, able to independently construct and manage a portfolio of stocks by analysing the daily trading data and fundaments of the public companies belonging to the S&P 500 index. The focus was mainly on increasing the Sharpe ratio of the portfolio and thus, the main research question for the study were as follows:

"Are the Deep Reinforcement Learning models competent to increase the risk-adjusted returns of a stock portfolio in the New York Stock Exchange?

The statistical significance of the model performance was tested by comparing the Sharpe ratio of the model with testing data to 5000 random portfolios to see, if it exceeds their average performance by more than 2 standard deviations. Testing part showed that the model outruns random portfolios statistically significantly in Sharpe ratio and total return. A noteworthy fact is the massive difference in significance levels, between Sharpe ratios and total returns. While the model's ability to increase Sharpe ratio is barely significant, it is very clear that the model is able to increase the profitability of the portfolio significantly, since the model performance exceeds the mean of the random portfolios by over 16 standard deviations. This implies that, the model's ability to increase the Sharpe ratio is mainly based on increasing the total returns, instead of reducing the risk. This is a

very interesting observation, since the literature review part showed that the classical methods work the best at reducing the riskiness instead of maximizing the returns.

The secondary objective for the study was to find out if the deep reinforcement learning agent can improve the portfolio performance over the market index. Thus, the secondary research question was as follows:

“Is the agent able to raise the portfolio performance over the market index?”

The answer for the question was clarified by running a regression to explain the portfolio returns with the market returns and testing the statistical significance of the positive alpha generated by the model. Alpha was tested on daily and monthly levels and although both regressions suggested a positive alpha coefficient, the results were not statistically significant. Thus, we cannot make conclusions about the model’s ability to surpass the market return. Both regressions suggested high Beta coefficients for the model, which tells that the model’s performance is highly related to the market performance. This can be noticed during the first two years of the test period while the market index is not rising, and the model is not able to generate positive returns. After the market index starts rising, the model starts to generate very high positive returns and finally at the end of the test period, experiences a severe crash while the market index falls.

6. CONCLUSIONS AND DISCUSSION

This Master's Thesis aimed to discover the applicability of machine learning models to active portfolio management with SP-500 constituents. The focus was mainly on the area of deep reinforcement learning, which has been recently capturing the field in artificial intelligence development and found competent even for very difficult tasks such as a financial portfolio management.

Theoretical part of this thesis covered the main parts of *Deep learning* and *Reinforcement learning* and presented the theoretical side behind the methods used in the research. It as well approached the portfolio management from the financial perspective by introducing the relevant financial theories for the task and finally united the two theoretical fields by covering the previous literature of active portfolio management with deep learning and deep reinforcement learning models. Data and methodology chapter presented the formation of the final dataset from the basic variables and defined the research setting of portfolio management on the basis of the reinforcement learning theory. The empirical research started with the feature selection, which showed the ineffectiveness of trading volume for portfolio management with this dataset. *Daily total return*, *Earnings to Price* and *Dividend yield* were selected to the optimization part, where 15 models with different parameters sets were trained in order to select the best combination for the final model. The final model performance was measured with testing data and the results were statistically tested.

The deep reinforcement learning agent developed in the study was able to self-generate a trading policy that lead to 328,9% returns and 0,91 Sharpe ratio during the 5-year test period. The agent outperformed all the benchmarks and was proved to statistically significantly improve the Sharpe ratio and total returns of the stock portfolio. However, alpha generated by the agent against the SP-500 index was not shown to statistically differ from zero and thus, cannot be stated that it would surpass the market return statistically significantly. The trading policy was found to possess a high risk and to be very opportunistic by generating the high profits with individual trades rather than by continuously beating the index on the daily basis. The results suggest the applicability of deep reinforcement learning models to portfolio management and are in a line with several previous studies such as (Moody *et al.*, 1998; Lee *et al.*, 2007; Jiang, Xu and Liang, 2017; Liang *et al.*, 2018).

The goal of the empirical part was to bring together the best practices found at the literature review and to adapt them to a large size stock environment. Another intention was to add a fundamental aspect to the stock selection, which has been found effective in classical portfolio management. These factors took the scope of the study beyond the previous literature, which were mainly focusing on small datasets with purely price data. The results suggest the usefulness of the fundamentals for the portfolio management task and prove the applicability of deep reinforcement learning to a large-scale stock environment.

The results raise several questions considering the further studies. The first is related to the research period, which is 21 years and probably unnecessary long for a single model. Market behaviour twenty years ago probably has little to none matter considering the current market behaviour and thus, it may not be useful for the model in practice. The model could be trained and tested with multiple shorter time periods to find out if it benefits from more recent training data.

Another topic for further investigation comes up from the trading frequency, which was daily in this research. Since stock market is an extremely noisy environment and the stock prices cannot be assumed always to shift in line with their actual trend, the trader agent could benefit from lower trading frequency that lengthens the holding period of single stock and thus protects the model from the negative influence of white noise.

The third topic considers the reward functions and their influence on the model behaviour. In this study, Sharpe ratio was used as a reward function to maximize agent's performance and it led to very high portfolio beta. To minimize the systematic risk of the portfolio, Sharpe ratio as the reward function could be replaced with Treynor ratio r_i/β_i, where r_i is the return of the portfolio i and β_i is the beta of the same portfolio. By using the Treynor ratio as the reward function, portfolio's dependency of the market returns might possibly be lowered thus simultaneously lowering the systematic risk of the portfolio. Another way to lower the agent's risk-level is to delete the survival bias from the training set, which is caused by selecting to the dataset only the current constituents of the SP-500 list at 2013. Due to the survival bias, the agent learns that the extremely low EP ratios are mostly a signal for the high future returns, although they are also a sign from the increased risk level. The problem could be tackled by expanding the training data with companies, that actually experienced a bankrupt or were hampered by the losses enough to not get listed to the SP500 index in 2013.

The feature selection part showed that the models learn useful factors from the data even by using only total returns. This is also an interesting fact considering of the further studies. After analysing the agent behaviour, it seems possible that the agent adapted a strategy, which is mainly based on the Earnings to Price ratio, since it was the most profitable strategy during the training period. By taking this into an account and training separate agents to create the most profitable strategies with different feature combinations and finally training a master network that combines the separate agents, the agent's performance could possibly be improved remarkably.

Limitations of the study relate to the dataset and to differences of the actual stock market and the stock market environment of this study. The stocks are assumed to be bought always with the closing price of the day. In real market situation this is not the case and the closing price may differ from the purchase price even when the stock was bought just seconds before the market closing. The model is also allowed to buy fractions of stocks to construct a portfolio, which is not the case in

the real market. All in all, these limitations are minor and they assumably have a quite low impact on the study reliability.

REFERENCES

Abarbanell, J. and Bushee, B. (1998) 'Abnormal Returns to a Fundamental Analysis Strategy', *The Accounting Review*, 73, pp. 19–45.

Aggarwal, C. C. (2018) *Neural networks and deep learning : a textbook*, *Machine Learning*. doi: 10.1111/j.1464-5491.2005.01480.x.

Almahdi, S. and Yang, S. Y. (2017) 'An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown', *Expert Systems with Applications*. Elsevier Ltd, 87(June), pp. 267–279. doi: 10.1016/j.eswa.2017.06.023.

Banz, R. (1981) 'The relationship between return and market value of common stocks', *Journal of Financial Economics*.

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, *12*(7), e0180944.

Basu, S. (1983) 'The Relationship between Earnings Yield, Market Value and Return for NYSE Common Stocks: Further Evidence', *Journal of Financial Economics*, 12(180), pp. 129–156.

Bellman, R. (1954) 'The Theory of Dynamic Programming', *Bulletin of the American Mathematical Society*, 60(6), pp. 503–515.

Blume, L., Easley, D. and O'hara, M. (1994) 'Market statistics and technical analysis: The role of volume', *Journal of Finance*.

Brennan, M., Schwartz, E. and Lagnado, R. (1997) 'Strategic asset allocation'.

Chan, P. K. and Stolfo, S. J. (1998) 'Toward Scalable Learning with Non-uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection', *In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 164–168. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.5098.

DeMiguel, V. *et al.* (2007) 'Improving Performance By Constraining Portfolio Norms: A Generalized Approach to Portfolio Optimization', *SSRN Electronic Journal*, (2005), pp. 1–67. doi: 10.2139/ssrn.967234.

DeMiguel, V., Garlappi, L. and Uppal, R. (2007) 'Optimal versus naive diversification: How inefficient is the 1/N portfolio strategy?', *Review of Financial Studies*, 22(5), pp. 1915–1953. doi: 10.1093/rfs/hhm075.

Ding, X. *et al.* (2015) 'Deep Learning for Event-Driven Stock Prediction', (Ijcai), pp. 2327–2333.

Fama, E. F. (1970) ‘Efficient Capital Markets: A Review of Theory Empirical Work’, *Journal of Finance*, 25(2), pp. 383–417.

Fama, E. F. and French, K. R. (1988) ‘Dividend yields and expected stock returns’, *Journal of Financial Economics*, 22(1), pp. 3–25. doi: 10.1016/0304-405X(88)90020-7.

Giles, C. L., Lawrence, S. and Tsoi, A. C. (2001) ‘Noisy Time Series Prediction using a Recurrent Neural Network and Grammatical Inference’, *Machine Learning*, 44, pp. 161–183.

Glorot, X., Bordes, A. and Bengio, Y. (2011) ‘Deep Sparse Rectifier Neural Networks’, (15), pp. 315–323.

Graham, B. (1949) The Intelligent Investor

Graves, A., Mohamed, A. R. and Hinton, G. (2013) ‘Speech recognition with deep recurrent neural networks’, *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, (3), pp. 6645–6649. doi: 10.1109/ICASSP.2013.6638947.

Grinblatt, M. and Titman, S. (1989) ‘Mutual fund performance: An analysis of quarterly portfolio holdings’.

He, K. *et al.* (2015) ‘Deep Residual Learning for Image Recognition’.

Hornik, K. (1991) ‘Approximation capabilities of multilayer feedforward networks’, *Neural Networks*, 4(2), pp. 251–257. doi: 10.1016/0893-6080(91)90009-T.

Jiang, Z., Xu, D. and Liang, J. (2017) ‘A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem’, pp. 1–31. Available at: http://arxiv.org/abs/1706.10059.

Jorion, P. (1991) ‘Bayesian and CAPM estimators of the means: Implications for portfolio selection’, *Journal of Banking and Finance*, 15(3), pp. 717–727. doi: 10.1016/0378-4266(91)90094-3.

Kalchbrenner, N., Grefenstette, E. and Blunsom, P. (2014) ‘A Convolutional Neural Network for Modelling Sentences’. Available at: http://arxiv.org/abs/1404.2188 (Accessed: 17 April 2019).

Kasten, G. and Swisher, P. (2005) ‘Post-Modern Portfolio Theory’, *Journal of Financial Planning*, 18(9).

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012) ‘ImageNet Classification with Deep Convolutional Neural Networks’, *ImageNet Classification with Deep Convolutional Neural Networks*, pp. 1097–1105. Available at: http://papers.nips.cc/paper/4824-imagenet-classification-w%5Cnpapers3://publication/uuid/1ECF396A-CEDA-45CD-9A9F-03344449DA2A.

LeCun, Y., Bengio, Y. and Hinton, G. (2015) ‘Deep learning’, *Nature*, 521.

Lee, J. W. *et al.* (2007) ‘A multiagent approach to Q-learning for daily stock trading’, *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 37(6), pp. 864–877. doi: 10.1109/TSMCA.2007.904825.

Li, Y. (2018) ‘Deep Reinforcement Learning’, pp. 1–150. Available at: http://arxiv.org/abs/1810.06339.

Liang, Z. *et al.* (2018) ‘Adversarial Deep Reinforcement Learning in Portfolio Management’. doi: 10.1111/j.1476-5829.2011.00268.x.

Lillicrap, T. P. *et al.* (2015) ‘Continuous control with deep reinforcement learning’. Available at: http://arxiv.org/abs/1509.02971.

Louridas, P. and Ebert, C. (2016) ‘Machine learning’. IEEE.

Lu, D. W. (2017) ‘Agent Inspired Trading Using Recurrent Reinforcement Learning and LSTM Neural Networks’. Available at: http://arxiv.org/abs/1707.07338.

Markowitz, H. (1952) ‘Portfolio Selection’, *The Journal of Finance*, 7(1), pp. 77–91.

Mnih, V. *et al.* (2013) ‘Playing Atari with Deep Reinforcement Learning’, pp. 1–9. Available at: http://arxiv.org/abs/1312.5602.

Moody, J. *et al.* (1998) ‘Performance functions and reinforcement learning for trading systems and portfolios’, *Journal of Forecasting*, 17(December 1997), pp. 441–470. Available at: /citations?view_op=view_citation&continue=/scholar%3Fhl%3Den%26as_sdt%3D0,99%26scilib%3D1050&citilm=1&citation_for_view=66ZlWDwAAAAJ:_Ybze24A_UAC&hl=en&oi=p.

Moody, J. and Saffel, M. (2001) ‘Learning to Trade via Direct Reinforcement’.

Nair, V. and Hinton, G. E. (2010) ‘Rectified Linear Units Improve Restricted Boltzmann Machines’, (3). doi: 10.1.1.165.6419.

Nelson, D. M. Q., Pereira, A. C. M. and De Oliveira, R. A. (2017) ‘Stock market’s price movement prediction with LSTM neural networks’, *Proceedings of the International Joint Conference on Neural Networks*, 2017–May(January 2018), pp. 1419–1426. doi: 10.1109/IJCNN.2017.7966019.

Neuneier, R. (1996) ‘Optimal Asset Allocation Using Adaptive Dynamic Programming’, *Advances in Neural Information Processing Systems*, pp. 952--958. Available at: papers3://publication/uuid/35691D4E-CBAE-422A-998F-761634D601E6.

Ramachandran, P., Zoph, B. and Le, Q. V. (2017) ‘Searching for Activation Functions’, pp. 1–13. Available at: http://arxiv.org/abs/1710.05941.

Schulman, J. *et al.* (2017) ‘Proximal Policy Optimization Algorithms’, pp. 1–12. Available at:

http://arxiv.org/abs/1707.06347.

Schwert, G. W. (2002) ‘Anomalies and Market Efficiency’.

Silver, D. *et al.* (2016) ‘Mastering the game of Go with deep neural networks and tree search.’, *Nature*. Nature Publishing Group, 529(7587), pp. 484–9. doi: 10.1038/nature16961.

Stengel, Robert F. (1994). *Optimal Control and Estimation.* Dover Publications.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. Machine Learning, 3(1):9–44.

Sutton, R. *et al.* (2000) ‘Policy Gradient Methods for Reinforcement Learning with Function Approximation’.

Sutton, R. and Barto, A. (2018) *Reinforcement Learning: An Introduction*.

Tesauro, G. (1995) ‘Temporal difference learning and TD-Gammon’.

Titman, S. and Jegadeesh, N. (1993) ‘Returns to buying winners and selling losers: Implications for stock market efficiency’, *The Journal of Finance*, 48(1), pp. 65–91. Available at: http://onlinelibrary.wiley.com/doi/10.1111/j.1540-6261.1993.tb04702.x/abstract.

Tsai, C. F. and Wu, J. W. (2008) ‘Using neural network ensembles for bankruptcy prediction and credit scoring’, *Expert Systems with Applications*, 34(4), pp. 2639–2649. doi: 10.1016/j.eswa.2007.05.019.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. Machine Learning, 8:279–292

Williams, R. J. (1992) ‘Simple statistical gradient-following algorithms for connectionist reinforcement learning’, *Machine Learning*.

Web documents:

Dertat, Arden (2017) Convolutional Neural Networks [web document] Available at: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Interactive Brokers (2019) [web document] Available at: https://interactivebrokers.com/en/index.php?f=1590&p=stocks1

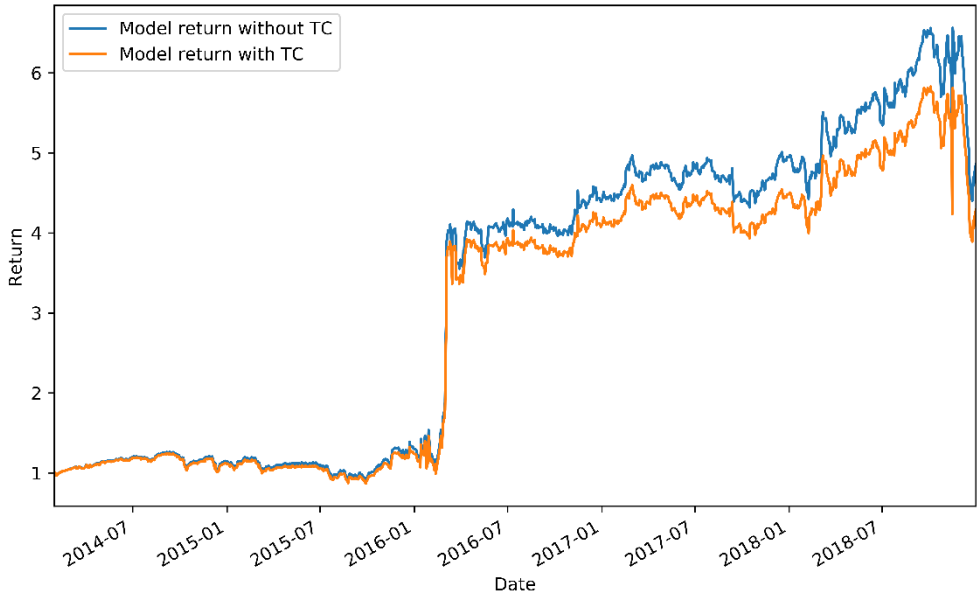
Tensorflow 2019. [web document] Available at: https://www.tensorflow.org/

Tensorflow 2019b. [web document] Available at: https://www.tensorflow.org/guide/tensors

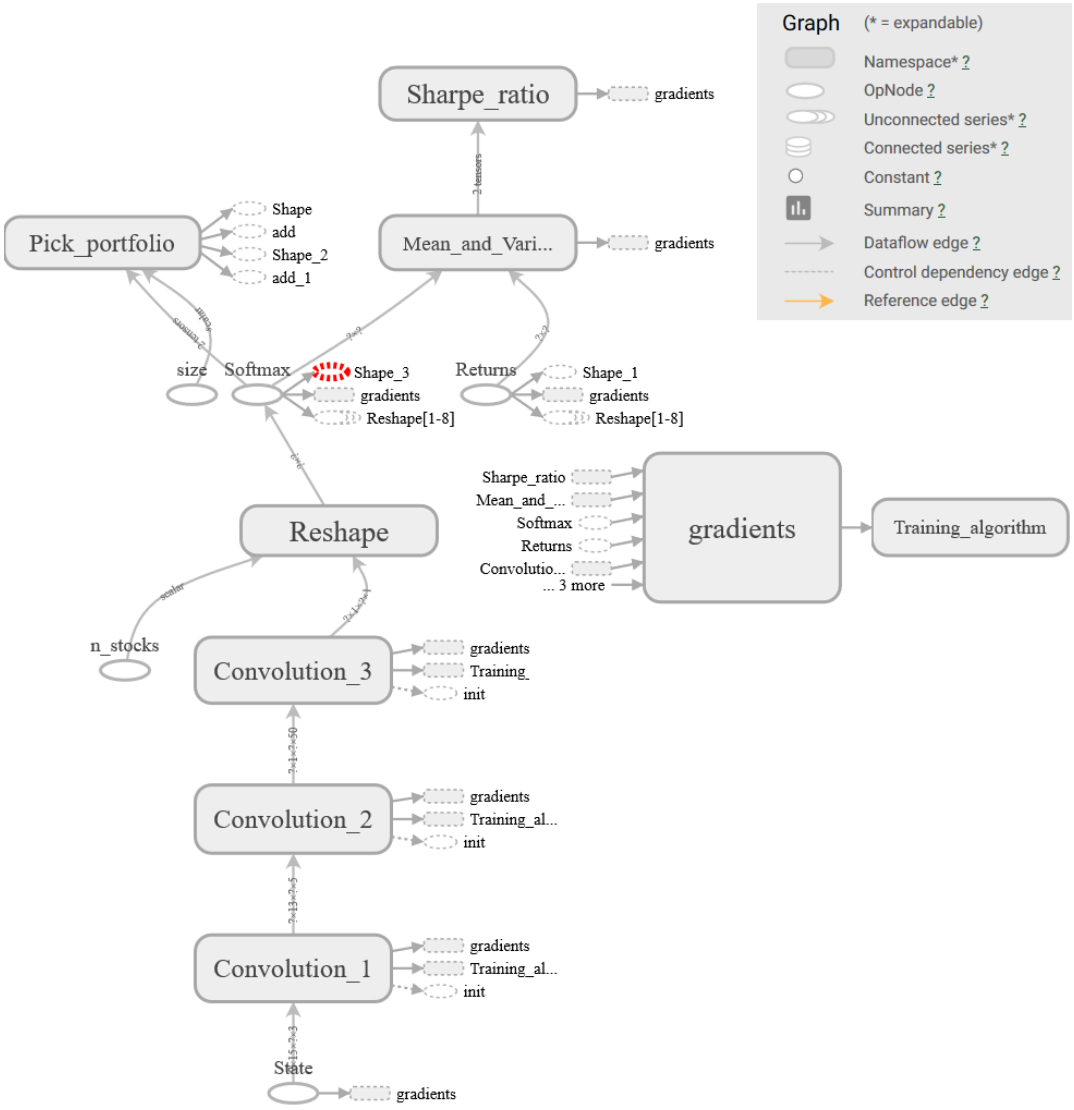
Valkov, V. (2017) [web document] Available at: https://medium.com/@curiousily/tensorflow-for-hackers-part-iv-neural-network-from-scratch-1a4f504dfa8

APPENDICES

Appendix 1. The affection of transaction costs to the model performance



Appendix 2. Full model graph printed from Tensorflow



Appendix 3. The optimal Markowitz's mean-variance portfolio

Company	Portfolio weight
INTUITIVE SURGICAL	5,99 %
KANSAS CITY SOUTHERN	1,67 %
NETAPP	1,85 %
TRACTOR SUPPLY	12,39 %
VERISIGN	0,30 %
EBAY	3,04 %
GILEAD SCIENCES	9,65 %
BIOGEN	8,16 %
NETFLIX	8,72 %
F5 NETWORKS	0,20 %
ALEXION PHARMS	4,92 %
SANDISK	1,55 %
APPLE	9,07 %
REGENERON PHARMACEUTICALS	0,79 %
AMAZON.COM	1,55 %
COGNIZANT TECH. SOLUTIONS	3,36 %
CELGENE	1,76 %
HUDSON CITY BANCORP	2,98 %
KEURIG GREEN MOUNTAIN	10,94 %
MONSTER BEVERAGE	11,11 %