Erasmus University Rotterdam
MSc Econometrics and Management Science

Master Thesis Quantitative Finance

Asset Allocation in Emerging Market Space using Wasserstein Generative Adversarial Networks

---

Author:
A. Bakx
507547

Supervisor:
S.H.L.C.G. Vermeulen
Second Assessor:
dr. M. Grith

30th June 2020

This page is intentionally left blank

Abstract

Emerging markets allow investors to profit from their increasing economic prosperity through publicly traded assets, thereby enabling them to earn superior returns. However, there are also significant risks involved with investing in emerging market assets which are often not well understood. This requires investors to follow a careful asset allocation approach that identifies risk and return accurately. In this paper I propose a novel methodology for asset allocation in emerging markets that succeeds at identifying interesting future return opportunities and constructs portfolios accordingly. My methodology yields superior portfolios that outperform the local index and the $\frac{1}{N}$-portfolio. My approach entails the Wasserstein Generative Adversarial Network to predict and the Non-Dominated Sorting Genetic Algorithm II with Monte Carlo to forecast closing prices for emerging market assets and construct emerging market portfolios, respectively. The emerging markets under consideration are South Africa and Russia, the most important emerging market countries in the EMEA region. I construct portfolios representing the most important industrial segments in the associated countries, each targeting different risk-return preferences. The portfolios are constructed from the mean-semivariance frontier. I find that the risk-averse and risk-seeking Russian portfolios outperform the benchmark and the local index and that the risk-averse portfolio in South Africa beats the local index, however all insignificantly. The latter can largely be attributed to the limited size of the data set, which adds an extra layer of complexity to this research.

Keywords: Emerging Markets, Generative Adversarial Networks, Wasserstein Distance, Non-Dominated Sorting Genetic Algorithm II, Mean-Semivariance Optimization

Preface

This paper has been written to fulfill the graduation requirements for a Master's degree in Econometrics and Management Science with a specialization in Quantitative Finance at the Erasmus School of Economics, Erasmus University Rotterdam. I was engaged in researching and writing this thesis from March to July 2020. During these months, the COVID-19 pandemic has brought unprecedented challenges requiring everyone to adapt to major changes in both business environments as well as in daily lives. I wish to show my gratitude to my thesis supervisor from Erasmus School of Economics, Erasmus University Rotterdam, Sebastiaan Vermeulen, for his guidance during these unprecedented times. He has provided accurate solutions to tackle logistical as well as content-related challenges. I have been grateful to have a supervisor who cared so much about my work and who responded to my questions promptly, despite the suboptimal circumstances. Moreover, I would like to thank dr. Maria Grith for reading my work and giving feedback towards the end of my research which enabled me to complete my work successfully.

To conclude, I would like to pay my special regards to my family and friends for their continued support and genuine interest in both my thesis content as well as my progress.

I look forward to continuing to investigate Generative Adversarial Networks. This thesis has strengthened my interests in emerging market investing and machine learning even more. I am excited to optimize the model further and to come up with novel frameworks for asset allocation in emerging markets.

Thank you,
Anne

30th June 2020

Contents

1 Introduction **1**

2 Literature Review **6**

3 Data **11**

4 Methodology **15**

4.1 Asset Price Forecasting 15

4.1.1 Wasserstein Generative Adversarial Network 15

4.1.2 Gradient Penalty 17

4.1.3 Model Setup 19

4.1.4 Robustness Check 23

4.2 Portfolio Construction 25

4.2.1 Multiobjective Optimization Problem 25

4.2.2 Model Evaluation 27

5 Results **28**

5.1 Deep Learning Architecture 28

5.1.1 Robustness Check 33

5.2 NSGA-II with Monte Carlo 34

5.3 Risk-Averse and Risk-Seeking Portfolios 35

6 Conclusion **40**

7 Discussion **41**

Appendices **46**

A Drawbacks of Traditional GANs **47**

B Adam Optimizer **51**

C Architecture **54**

D Forecasting Performance **56**

E Robustness **59**

F Frontiers from NSGA-II and MC **64**

1 Introduction

Investors consider emerging market assets as a unique and attractive investment due to the superior return opportunities that come along with investing in these assets. However, there are also significant risks involved, which are often not well understood. Emerging markets are frequently exposed to tail events caused by political upheaval or natural disasters that seriously slow down their growth and thus negatively impact asset prices. Moreover, emerging market assets often have a short history of public trading. The absence of a reliable track record causes investors to struggle with identifying the fair value of these assets. On the other hand, the rewards can outweigh the risks: if investors correctly identify an opportunity, they can greatly profit from the increasing economic prosperity. The unstable and immature character of assets in emerging markets requires a careful approach regarding their allocation. Existing techniques are often too simplistic and do not capture their unstable and immature behavior accurately. In this paper, I propose a methodology for the construction of emerging market portfolios that acknowledges the unstable and immature character of emerging markets and operates accordingly. My methodology entails a deep learning architecture for the prediction of asset closing prices and a multiobjective evolutionary algorithm with Monte Carlo for the construction of portfolios that target asymmetric risk-return preferences. Through combining the methods, my approach successfully deals with the unstable and immature character of assets in emerging markets and yields superior portfolios that outperform the local index and the $\frac{1}{N}$-portfolio. I apply this methodology to the two most important emerging markets countries in EMEA (Europe - Middle East - Africa): South Africa and Russia.

Markowitz (1952) elaborates on the construction of portfolios and introduces the mean-variance framework. He explains that the process of portfolio construction can be divided into two stages. The first stage considers predicting the performance of the securities that are available to be incorporated into the portfolio. The second stage considers the decision-making process for portfolio construction. In this research, I address both stages as I am convinced they are intertwined, i.e. the portfolio optimization problem is based on the information that can be derived from the probability distribution of the securities from the first stage. To clarify this: the popular Markowitz framework forms its allocation solely based on mean and variance (Markowitz, 1952). The only probability distribution that is fully characterized by mean and variance is the normal distribution, so by using

the Markowitz framework for portfolio construction, you implicitly assume Gaussian asset returns (Tobin, 1958).

To address the first stage, I implement a deep learning architecture to predict asset closing prices. I deploy a Wasserstein Generative Adversarial Network (WGAN). WGANs have previously been used for language models and image recognition (Arjovsky et al., 2017; Gulrajani et al., 2017), and for the prediction of financial time series (Mariani et al., 2019; K. Zhang et al., 2019; M. Zhang et al., 2012). In this research, the assets under consideration are stocks and indices and I predict up to 20 days (i.e. 1 month) ahead. The Generative Adversarial Network (GAN) was introduced by Goodfellow et al. (2014) and encompasses two Neural Networks: the generator and the discriminator. The generative model generates fake data points, while the discriminator decides upon the fit of the fake data with the real data. As GANs are found to have an unstable training process, the WGAN was introduced by Arjovsky et al. (2017). The WGAN exhibits a stable training process and produces robust results under various hyperparameter settings.

Previous research in the space of stock price prediction focuses on both traditional econometric techniques as well as machine learning techniques. Regarding traditional econometric techniques, the Geometric Brownian Motion is one of the most popular models to generate asset paths (Hull, 2003). This model requires input variables such as the mean and variance, which are estimated by looking at historical price information. A recurrent problem in financial markets is that history does not accurately represent the future. Moreover, the variance is assumed to be constant over time, and the simulated path is assumed to be continuous. However, often both assumptions do not hold in real-world dynamics as the asset variance changes over time and asset paths often exhibit jumps, especially in emerging market space, caused by unpredictable events or news updates (Bekaert et al., 1998). There are also other functions that can be implemented to predict closing prices instead of the Geometric Brownian Motion. However, these functions all have in common that they pose restricted assumptions on the general pattern and the noise terms of the closing prices. Moreover, to find distribution that describes the closing prices best, maximum likelihood estimation (i.e. minimization of the Kullback-Leibler (KL) divergence) is often used (Arjovsky et al., 2017). This technique has its drawbacks for both high-dimensional problems as well as problems restricted to low-dimensional manifolds. In complex high-dimensional problems, maximizing the log-

likelihood becomes either unfeasible or inaccurate due to the large amount of parameters that need to be optimized (Salakhutdinov & Hinton, 2009). We also encounter problems with maximum likelihood estimation in situations where the distributions are supported by low-dimensional manifolds (up to 4-dimensional) in high-dimensional spaces. For distributions supported by low-dimensional manifolds, you can observe that the manifolds of both the model as well as the true distribution either intersect transversally (yielding the intersection to have measure 0 in both manifolds) or not intersect at all. In both cases, the KL divergence vanishes causing the maximum likelihood problem to be degenerate. An extensive explanation of this phenomenon can be found in Appendix A. Adding a noise term to the model distribution could alleviate this problem, however, this negatively impacts the quality of the generated samples (Arjovsky et al., 2017). There is a strong use case for WGAN instead of traditional econometric techniques as it does not perform maximum likelihood estimation, and therefore, it is feasible in high-dimensional spaces, even if the real distribution is supported by low-dimensional manifolds. In emerging market space, the upside of using WGANs over traditional econometric techniques is even amplified, as this architecture does not pose restricted assumptions on the noise terms and loss functions, and because it is able to analyze hidden patterns and to capture nonlinear relations in the data.

In the field of machine learning, architectures such as Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs) (Kara et al., 2011), and Long Short-Term Memory (LSTM) Networks and Convolutional Neural Networks (CNNs) (Selvin et al., 2017), have been deployed. However, GAN, and especially WGAN, is preferred over these single network architectures as it models the uncertainty of the market as a whole rather than forecasting single stocks, and enables us to investigate various simulation runs to identify possible future market scenarios. K. Zhang et al. (2019) and Zhou et al. (2018) both propose GAN architectures to predict closing prices of financial instruments, and Mariani et al. (2019) propose the WGAN architecture. Whereas K. Zhang et al. (2019) forecast various regressors for the closing price and Zhou et al. (2018) forecasts the direction (up or down), Mariani et al. (2019) forecast closing prices directly. In this work, I deploy an architecture that is inspired by the architecture from Mariani et al. (2019), as their architecture yields promising results for the developed market portfolios. i.e. their architecture significantly outperforms the Markowitz framework.

To address the second stage, I introduce a portfolio optimization framework that captures the behavior of the asset returns generated by WGAN. Portfolio optimization techniques focus on solving a multiobjective function. The multiobjective function yields the efficient frontier, i.e. the Pareto front. The Pareto front resembles a trade-off between portfolio risk and return. To find the Pareto front, we need to specify functions that approximate risk and return. In this work, I use the mean of the portfolio return as a measure for return. The risk measure can be derived from the investor's utility function. As a measure of risk, I use the semivariance which is able to capture the volatile, skewed, and asymmetric behavior of the return distribution of the emerging market assets as it takes into account higher-order moments (Markowitz, 1991). It does so by only capturing downward deviation, which makes sense in this application as investors only worry about losing. Mariani et al. (2019) implement the mean-variance framework. The mean-variance framework has been one of the most influential models in the history of finance, however, it is sometimes criticized as it assumes quadratic utility preferences. Thereby, implicitly it assumes Gaussian stock returns (Tobin, 1958). As with the normality assumption for stock returns, the quadratic utility function is characterized solely by the mean and variance parameters. The variance measures the probability of an upward or downward move equally, i.e. it assumes that the underlying distribution of returns is symmetric. These assumptions are unlikely to materialize in the setting of asset return distributions in emerging market space, as return distributions are often skewed with many outliers due to the unstable character of these markets.

Following my procedure, I show that WGAN succeeds at predicting the general pattern of the asset closing prices, but that it underestimates their semivariance. Moreover, I demonstrate that the portfolios in the Russian market outperform both the local index as well as the benchmark portfolio (i.e. the $\frac{1}{N}$-portfolio) up to 55% and 40% compounded return respectively for an investment period of 21 months. In South Africa, only the portfolio that follows a risk-averse approach outperforms the local index and the benchmark portfolio returns remain superior. The risk-averse approach considers a portfolio with a return that is 16,7% bigger than the lowest estimated return. However, due to the limited size of the data set, I do not accomplish to get significant results.

To summarize, the contribution of this research to the existing literature is twofold. Firstly, I extend the work of Mariani et al. (2019) by combining the WGAN architecture

with the mean-semivariance framework, which accounts for higher-order moments in the asset return distributions and captures the asymmetric desirability for variance that investors exhibit. Secondly, the methodology is applied to emerging market assets, which brings along additional difficulties with respect to developed market assets due to their unstable and immature character.

2 Literature Review

Emerging market assets are often considered as a separate asset class in portfolio management, due to their unique behavior compared to developed market assets. Bekaert et al. (1998) show there are a significant skewness and kurtosis in the returns of emerging market assets. They conclude that the normal distribution for asset returns is not suitable for emerging market asset returns, and therefore, they suggest that portfolio optimization techniques in the presence of non-normality should be investigated further.

Deep learning techniques offer an elegant method to approximate asset return distributions. In the field of single network deep learning techniques, there is a distinction between the Feedforward Backpropagation Neural Networks and the Recurrent Neural Networks (RNNs). The Feedforward Backpropagation Network does not take into account the sequence in which inputs are presented. On the contrary, in the nodes of the RNN there exists a feedback mechanism that enables the network to use information from previous patterns along with the present inputs, i.e. RNNs can learn spatiotemporal patterns. Long Short-Term Memory (LSTM) models were first introduced by Hochreiter and Schmidhuber (1997) as a slightly modified candidate for the RNN architecture. RNNs often face the problem of vanishing gradients during training. LSTMs overcome this by using memory cells. They are designed to model temporal sequences and long-range dependencies more accurately than RNNs, by implementing memory cells composed of gates. These memory cells distinguish between long-term and short-term time-dependent information. Jiang et al. (2018) perform a time series analysis on the closing prices of the Dow Jones Index and the Shanghai Composite Index. They implement both an RNN and LSTM model and find that the LSTM model fits the data better than the RNN model. Besides the LSTM model, Convolutions Neural Networks (CNNs) are also used in the setting of time series analysis. CNNs belong to the class of the Feedforward Backpropagation Neural Networks and contain one or more convolutional layers, which apply a convolution operation on the input of the respective layer. This operation allows for the identification of deep dependence structures with relatively few parameters. Through the application of kernels, CNNs can capture both spatial and temporal dependence structures, depending on the direction in which the kernels are applied. Selvin et al. (2017) implement the CNN architecture for stock price forecasting. They compare the performance of the CNN model with a sliding window implementation with respect to the LSTM and RNN models. They

find that the CNN performs better as it only uses the given input sequence and adjusts its prediction according to the patterns occurring in the current window. On the contrary, the LSTM architecture identifies both long and short term dependencies and uses these for prediction. As the environment of stock price forecasting is highly dynamical, it is not always a good idea to include long-term dependencies in the prediction of stock prices as these dependencies might develop over time.

As pointed out by Mariani et al. (2019), single network techniques lack a stochastic component that models the uncertainty that comes along with forecasting stock prices. To solve this issue, they propose the Wasserstein Generative Adversarial Network (WGAN) with convolutional layers for the generator as well as the discriminator. The WGAN is a multi-network deep learning architecture consisting of a generator network and a discriminator network. By training the generator and the discriminator simultaneously, the generator aims to generate data points that approach the real data points as good as possible, thereby adjusting its generative process based on the discriminator's feedback. The discriminator tells the generator to what extent the synthetic data points fit the real data points (Arjovsky et al., 2017). Mariani et al. (2019) use the historical daily closing prices for a set of portfolio assets. Conditioning on this information, they model the uncertainty and simulate various asset paths reflecting the most recent information as well as the uncertainty associated with predictions in the financial markets. They show that their methodology is capable of generating asset paths under various scenarios and that their predictions are updated based on the most recent market information, thereby not assuming that the future probability density function equals the historical probability density function. K. Zhang et al. (2019) also propose a methodology for stock market prediction with a GAN. They use an LSTM network for the generator and a Multilayer Perceptron (MLP) for the discriminator. However, instead of considering a set of assets simultaneously, they only look at S&P500 Index data. Zhou et al. (2018) deploy the LSTM network for the generator and the CNN for the discriminator. However, they again generate future values for only one asset at a time.

One major difficulty with training (W)GANs is mode collapse, also called the Helvetica scenario. If this occurs, the generator produces the same data points over and over again. One solution is to make the discriminator function 1-Lipschitz, i.e., the absolute value of the first derivative of the discriminator loss is smaller than or equal to 1, across its whole

domain. To enforce this constraint, Arjovsky et al. (2017) implement weight clipping. Gulrajani et al. (2017) propose a method to enforce Lipschitz constraints alternative to weight clipping. They penalize the norm of the gradient of the discriminator loss with respect to its input: the Wasserstein GAN Gradient Penalty (WGAN-GP) framework. They argue that two problems arise with weight clipping. The first problem is that the discriminator loss is pushed towards much simpler functions that do not incorporate higher-order moments in the data distribution. The second problem is the presence of exploding and vanishing gradients. Gulrajani et al. (2017) show that the WGAN-GP framework performs better than the WGAN framework proposed by Arjovsky et al. (2017) in terms of training speed and sample quality.

In the field of asset allocation, the mean-variance framework, also referred to as the Modern Portfolio Theory (MPT), is widely used (Markowitz, 1952). The MPT shows both that the investor should diversify and that he should maximize expected return. The portfolio with the maximum expected return is not necessarily the one with minimum variance. There is a trade-off between expected return and variance, i.e. an investor can take on extra risk (increase the portfolio' variance) resulting in a higher expected return according to its risk appetite. The traditional Markowitz framework fails to capture higher-order moments in the return distributions, as it restricts its risk assessment to the variance measure. Moreover, it treats upward and downward movements equally. However, investors prefer upward movements in stock returns, whereas they consider downward movements as undesirable. Therefore, I discuss some alternative risk measures for the variance, namely the Value at Risk (VaR), the Conditional Value at Risk (CVar), and the semivariance.

The VaR is a widely used alternative risk measure for the variance. The VaR treats downward movements differently with respect to the desired upward movements and can be calculated in a variety of manners, which entail both parametric and nonparametric approaches (Linsmeier & Pearson, 1996). The variance-covariance method is a parametric approach and places strong assumptions on the return distribution. The historical simulation method and Monte Carlo simulation are nonparametric methods. The historical simulation method assumes that the past reflects the future, whereas the Monte Carlo simulation simulates several possible future scenarios and approximates the VaR based on these simulations. Campbell et al. (2001) propose VaR to measure risk as a replacement

for the variance for portfolios consisting of assets with non-normal asset returns, and show that making parametric assumptions on the VaR measure greatly affects the outcomes.

The CVaR, or Expected Shortfall, is a VaR based measure that overcomes some limitations of the VaR. The VaR is not a coherent risk measure as it lacks subadditivity and convexity. As the CVaR is coherent, it eases mathematical computations. However, Lim et al. (2011) show that the mean-CVaR optimization framework does not yield robust results due to estimation errors in the mean and CVaR and that this is even amplified for heavily tailed distributions. Therefore, despite that the CVaR is coherence, it does not necessarily improve the outcomes with respect to the non-coherent VaR measure. Moreover, Markowitz (2010) shows that a small shift in probability distributions result in different optimal portfolio compositions, as both the VaR and CVaR measures are not continuous functions of the probability distribution. This new portfolio, however, does not necessarily yield significantly better results in terms of utility. Because of their discontinuous behavior, Markowitz (2010) argues that the VaR and CVaR should not replace the traditional risk measure, the variance.

To handle the asymmetric desirability of gains and losses and the higher-order moments in the return distribution, Markowitz (1991) proposes the semivariance as a substitute for the variance. The major challenge in the implementation of the semivariance is the existence of endogeneity between the portfolio semivariance matrix and the weights given to each asset. Whereas mean-variance problems have easy-to-use closed-form solutions, mean-semivariance problems cannot be solved analytically, so numerical algorithms are required. Markowitz et al. (1993) implement the Critical Line Algorithm to compute the mean-semivariance frontier. By including additional variables, they tackle the endogeneity problem. They show that it takes the Critical Line Algorithm twice as long to compute the mean-semivariance frontier compared to the mean-variance frontier. Another tool that is often used in the field of portfolio optimization is the multiobjective evolutionary algorithm (MOEA). MOEAs aim at finding a set in the field of portfolio optimization of Pareto optimal solutions in one single run. The Pareto front is the set of solutions of the multiobjective optimization problem where an objective function can only improve at the expense of the value of another objective function (Macedo et al., 2017). There are a lot of different MOEAs. With MOEAs, there is no single algorithm that consistently outperforms the other ones, and the performance is mainly studied experi-

mentally. Macedo et al. (2017) propose the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) and the Strength Pareto Evolutionary Algorithm 2 (SPEA2). By using these techniques, the endogeneity problem is tackled by construction as the asset weights are known *a priori* since they are generated by the algorithm in the previous iteration. They conclude that NSGA-II outperforms SPEA2, as NSGA-II stimulates diversity as it approaches the Pareto front due to its fast nondominated sorting procedure, whereas SPEA2 shrinks the solutions to the middle of the Pareto front.

Li and Zhang (2008) argue that the shape of the Pareto front determines the performance of the MOEA to a large extent. They introduce a set of test problems and investigate the performance of various MOEAs for complicated Pareto sets. They find that MOEAs experience a lot of difficulties in finding the solutions for complicated Pareto fronts. Therefore, they introduce the MOEA with Decomposition (MOEA/D). This technique decomposes the multiobjective optimization problem into single-objective optimization subproblems and simultaneously optimizes these subproblems. Moreover, they introduce a differential evolution (DE) and polynomial mutation operator with this technique, MOEAD/D-DE, and also for NSGA-II, i.e. NSGA-II-DE. They show that MOEA/D-DE outperforms NSGA-II-DE for complex Pareto shapes due to the decomposition of the multiobjective problem. H. Zhang et al. (2018) propose MOEA/D with Constraint Programming (MOEA/D-CP) to solve optimization problems in the field of portfolio optimization. MOEA/D-CP is based on MOEA/D and introduces a constraint on the weight vector to generate an evenly distributed set of weight vectors. Therefore, their optimization problem includes a set of additional restrictions. They assign a bigger weighting to recent price information as this information greatly determines the future price of the reference price calculation. Moreover, they introduce upper and lower limits to each asset's share in the portfolio. By introducing these additional restrictions, the shape of the Pareto optimal solution becomes extremely complex. They show that the MOEA/D-CP performs much better than the MOEA/D. One major drawback of both the MOEA/D and the MOEA/D-CP algorithms is the extensive and barely insightful parameter set that these algorithms require. Therefore, implementing this algorithm only pays off for complex Pareto fronts that cannot be solved by less parameter intense MOEAs.

3 Data

This section describes the data that is used in this research, followed by a brief discussion on how the data is used for training and testing the deep learning architecture.
In this research, I use daily closing prices for two types of financial instruments: stocks and indices. I consider two portfolios. The first portfolio consists of South Africa assets and focuses on mining and materials companies. The second portfolio focuses on Russian oil and gas companies. Both sectors are extremely important for the local economies, and therefore, many foreign investors are interested in investing in these sectors to profit from the country's increasing economic prosperity. Moreover, I include the JSE All Share Index (ALSI) in the South African portfolio and the MOEX Russia Total Return Index (MCFTR) in the Russian portfolio. These indices reflect the overall performance of the South African and Russian stock markets, respectively. Investing in these assets is considered to be less risky compared to investing in individual stocks as they comprise a set of assets, which introduces diversification opportunities.

To obtain the daily closing prices of the stocks in the emerging market portfolios, I use the Global - Daily dataset from the Compustat - Capital IQ database on the Wharton Research Data Services (WRDS) website (Wharton Research Data Services, 2020). The shares' closing prices are obtained using their corresponding ISIN code. The stocks to be included in both portfolios must meet the following set of criteria: (1) the stock should be publicly tradeable in the period under consideration, i.e. 2010-02-01 to 2019-05-31; (2) the stock should be publicly tradeable on the local stock exchange in the local currency, as in this research I consider single-currency portfolios with their associated assets trading on the same exchange; (3) the float-adjusted market cap should be at least $2 billion: I restrict the portfolios to mid and large-cap stocks, as small-cap stocks often exhibit illiquid characteristics, and I want to avoid trading issues regarding liquidity constraints. The portfolio constituents for this research are listed in Table 1. The period under consideration is 2010-02-01 to 2019-05-31, i.e. $2,435$ observations for the South African portfolio and $2,437$ for the Russian portfolio due to a small discrepancy in stock market holidays.

As WRDS does not provide daily index data for emerging market regions, I extract the index data directly from the local stock exchange websites, i.e. the daily closing prices for the MOEX Russia Total Return index are obtained from the Moscow Exchange

website (Moscow Exchange, 2020), and the daily closing prices for the JSE All Share Index are obtained from the Johannesburg Stock Exchange website (Johannesburg Stock Exchange, 2020). Both indices are capitalization-weighted, which means that the index constituents are weighted according to their market capitalization, i.e. the value of their outstanding shares. The indices under consideration are both total return indices. Total return indices reinvest dividends back into the index. This methodology increases the accuracy of the performance representation of the index as there is accounted for stocks that do not issue dividends, and instead, reinvest their earnings (if applicable) back into the company. In total return indices, therefore, dividend-paying and non-dividend paying stocks are treated equally. The JSE All Share Index (ALSI) consists of 164 companies. The MOEX Russia Total Return Index (MCFTR) consists of approximately 50 stocks.

(a) *The South African portfolio. Sector: mining and materials*

Company name	Symbol	ISIN CODE	Type
African Rainbow Minerals	ARM	ZAE000054045	Share
Anglo American Platinum Limited	AMS	ZAE000013181	Share
AngloGold Ashanti	ANG	ZAE000043485	Share
Assore Limited	ASR	ZAE000146932	Share
Gold Fields Limited	GFI	ZAE000018123	Share
Kumba Iron Ore Limited	KIO	ZAE000085346	Share
JSE All Share Index	ALSI	I1ZAF003 [a]	Index

[a]The ticker instead of the ISIN code as this is used by the database

(b) *The Russian portfolio. Sector: oil and gas*

Company name	Symbol	ISIN CODE	Type
Gazprom	GAZP	RU0007661625	Share
Lukoil	LKOH	RU0009024277	Share
Novatek	NVTK	RU000A0DKVS5	Share
Rosneft	ROSN	RU000A0J2Q06	Share
Surgutneftegas	SNGS	RU0008926258	Share
Tatneft-3	TATN	RU0009033591	Share
MOEX Russia Total Return Index	MCFTR	RU000A0JWY86	Index

Table 1. *The emerging market assets*

In Table 2, general statistics are summarized for the return of each portfolio constituent in the South African and Russian portfolios. The asset returns are calculated from the daily closing prices, thereby accounting for dividend payments:

$$R_{i,t} = \frac{P_{i,t} - P_{i,t-1} + D_{i,t}}{P_{i,t-1}}, \tag{1}$$

where $R_{i,t}$ represents the return for stock i at time t, $P_{i,t}$ the closing price for stock i at time t, and $D_{i,t}$ the dividend payment of stock i at time t (which is set to 0 if there is no dividend payment). For the indices, $D = 0$ as I consider total return indices. We see that all asset returns have skewnesses different from 0 and kurtoses larger than 3. This indicates that the normality assumption for asset returns is not feasible for this asset selection.

(a) *The South African portfolio. All stocks have a positive skewness, except for ANG. AMS, ANG, GFI and KIO have a kurtosis larger than* 3.

Symbol	Mean	Stdev	Kurtosis	Skewness
ARM	0.00046	0.00050	3.34	0.32
AMS	0.00033	0.00050	4.52	0.69
ANG	0.00022	0.00053	2.99	0.45
ASR	0.00064	0.00069	118.12	-4.49
GFI	0.00030	0.00056	4.28	0.28
KIO	0.00084	0.00060	10.35	0.79
ALSI	0.00037	0.00020	1.34	-0.12

(b) *The Russian portfolio. All stocks have a positive skewness. GAZP and NVTK have a kurtosis larger than* 3*, and ROSN and TATN have a kurtosis that is substantially lower than* 3.

Symbol	Mean	Stdev	Kurtosis	Skewness
GAZP	0.00035	0.00033	7.96	0.47
LKOH	0.00078	0.00030	3.16	0.04
NVTK	0.00107	0.00039	9.87	0.05
ROSN	0.00050	0.00033	2.09	0.26
SNGS	0.00020	0.00033	2.92	0.23
TATN	0.00107	0.00040	2.34	0.32
MCFTR	0.00051	0.00026	5.55	-0.64

Table 2. *The mean, standard deviation, kurtosis, and skewness for each constituent return in the portfolios. The normality assumption is not feasible.*

To get the overall performance of WGAN, the data set is split into two sets: the train set and the test set. Following Chong et al. (2017), I use an 80-20 split between the train and the test data. First, the model is trained on the train subset. During training, the weights and biases in each layer are optimized. These parameters are fixed at their optimal values, and subsequently, the model is tested on the test subset. I predict up to 20 days, conditioning on historical price information of the last 40 days. I use a rolling window to split both the train and test data into samples of 60 days. As I have $2,435$ observations covering the period 2010-02-01 to 2019-05-31 for the South African portfolio, I have $1,948$ observations in the train set, covering the period 2010-02-01 to 2017-07-19, and 487 observations in the test set, covering the period 2017-02-20 to 2019-05-31. This yields $1,889$ samples for the train set, and 428 samples for the test set. For the Russian portfolio, I have $2,437$ observations. This means that I have $1,949$ observations in the train set, and 488 observations in the test set, yielding $1,890$ samples in the train set, and 429 in the test set.

4 Methodology

This section gives an extensive explanation of the methodology implemented to optimize the asset allocation for emerging market portfolios. First, I elaborate on the Neural Network architecture that is deployed to forecast the closing prices of the assets in the portfolios, starting with explaining the architecture, followed by discussing the problem-specific model setup. Secondly, I address the framework that is used to construct the portfolios.

4.1 Asset Price Forecasting

4.1.1 Wasserstein Generative Adversarial Network

In this research, I deploy the Wasserstein Generative Adversarial Network (WGAN) for the prediction of asset closing prices. WGAN is a variation of GAN that avoids vanishing gradients during training by implementing the Wasserstein distance rather than the Jensen-Shannon (JS) divergence as with traditional GANs.

The main idea of GAN is that it generates samples from stochastic input that approximate real data samples accurately, without knowing the probability density function of the data explicitly. GAN consists of two neural networks: the generator and the discriminator. The generator generates samples and the discriminator distinguishes between generated samples and samples from the data. Hereby the discriminator assigns the value 1 if it considers samples to be real (i.e. from the data) and 0 if it considers them to be fake (i.e. generated by the generator). By using feedback from the discriminator while training the generator, the distribution of the generator is optimized such that it approximates the real data distribution better. The discriminator and generator are being trained simultaneously. Goodfellow et al. (2014) explain that the GAN solves a minimax two-player game:

$$\begin{aligned} \min_G \max_D L(G, D) &= E_{\boldsymbol{x}\sim P_{\mathrm{r}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + E_{\boldsymbol{z}\sim P_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \\ &= E_{\boldsymbol{x}\sim P_{\mathrm{r}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + E_{\boldsymbol{x}\sim P_{\boldsymbol{g}}(\boldsymbol{x})}[\log(1 - D(\boldsymbol{x}))], \end{aligned} \tag{2}$$

where L represents the loss function of the GAN, G represents the generator, D represents the discriminator, p_{r} the data distribution (i.e. the real distribution), $p_{\boldsymbol{z}}$ the stochastic distribution, and p_{g} the generator distribution. From Equation 2, you can see that the

stochastic variables are passed through the generator. In this research, I sample $\boldsymbol{z}$ from the standard uniform distribution, i.e. $\boldsymbol{z} \sim U(0, 1)$, as this distribution is often used for sampling purposes for a broad range of distributions via inverse transform sampling.

In the minimax two-player game of GANs, the generator can only win at the expense of the discriminator, and the other way around. The discriminator aims to maximize the probability of correctly identifying samples as being real or fake, whereas the generator optimizes its distribution function such that it produces samples that match samples from the real data as good as possible, thereby minimizing $E[\log(1 - D(G(\boldsymbol{z})))]$. If the generator succeeds at generating realistic samples, the discriminator struggles to identify whether samples come from the real or generative distribution.

In the traditional GAN architecture, under the optimal discriminator, minimizing with respect to the discriminator boils down to minimizing the Jensen-Shannon (JS) divergence. However, minimizing this divergence in this application often leads to vanishing gradients, as the probability distributions of the generator and discriminator are supported by low-dimensional manifolds in a high-dimensional space (Arjovsky & Bottou, 2017). Therefore, Arjovsky et al. (2017) propose the WGAN, which deploys the Wasserstein distance rather than the JS divergence measure. The huge advantage of using the Wasserstein distance is that it still produces an accurate measure of distance even when the probability distributions are supported by low-dimensional manifolds. An elaborate explanation of the problem with the JS divergence is described in Appendix A. The Wasserstein distance is described by:

$$W(P_r, P_g) = \inf_{\gamma \sim \Pi(P_r, P_g)} E_{(x,y)\in\gamma}[||x - \bar{x}||], \tag{3}$$

where $\Pi(P_r, P_g)$ denotes the set of all joint distributions $\gamma(x, \bar{x})$ whose marginals are respectively P_r (the data distribution) and P_g (the generative distribution). P_g generates $\bar{\boldsymbol{x}} = G(\boldsymbol{z}|\boldsymbol{X}_G), \boldsymbol{z} \sim p(\boldsymbol{z})$, i.e. $\boldsymbol{z}$ is conditioned on X_G. The Wasserstein distance is the minimum cost of transporting mass when converting P_g into P_r. For all possible joint distributions $\gamma(x, \bar{x}) \in \Pi(P_r, P_g)$, a real sample (x) and a generative sample $(\bar{x})$ can be obtained, and, therefore, the distance can be calculated. By taking the expectation of all samples x and $\bar{x}$ from the same joint distribution, and subsequently taking the infimum across all joint distributions, the Wasserstein distance is computed.

As it is impossible to evaluate $\inf_{\gamma\sim\Pi(P_r,P_g)}$ analytically, Arjovsky et al. (2017) suggest

an alternative form by using the Kantorovich-Rubinstein duality:

$$W(P_r, P_g) = \sup_{||f||_L \leq 1} E_{x \sim P_r}[f(x)] - E_{\bar{x} \sim P_g}[f(\bar{x})], \tag{4}$$

where $||f||_L \leq 1$ represents a set of 1-Lipschitz functions. A function f is 1-Lipschitz continuous if for any two elements x_1 and x_2 in the domain of f, it holds that: $|f(x_1) - f(x_2)| \leq |x_1 - x_2|$, i.e. the absolute value of the derivative of f is smaller than or equal to 1. By implementing a Lipschitz constraint on f, the maximum rate of change of the function is restraint. In this application, f represents the discriminator network. It makes sense to restrict the rate of change of the discriminator as it needs to provide constructive feedback to the generator during training. To get rid of the supremum function, Arjovsky et al. (2017) define a function that is parameterized by the vector w, i.e. f_w. f_w still represents the discriminator, now explicitly parameterized by w. With a Neural Network, we can approximate almost all the possible f from Equation 4. Therefore the maximum accurately approximates the supremum, which translates into an alternative, more practical, form for Equation 4, namely:

$$W(P_r, P_g) \approx \max_{w:||f_W||_L \leq 1} E_{x \sim P_r}[f_w(x)] - E_{\bar{x} \sim P_g}[f_w(\bar{x})]. \tag{5}$$

Here, we can restrain the derivative of f with respect to an arbitrary input x, i.e. $\frac{\partial f_w}{\partial x}$ to 1. Taking the maximum yields the following WGAN loss function, in terms of the discriminator:

$$L = E_{x \sim P_r}[D(x)] - E_{\bar{x} \sim P_g}[D(\bar{x})]. \tag{6}$$

4.1.2 Gradient Penalty

Equation 6 is the output of the discriminator (i.e. the discriminator loss) and is also called the critic. To enforce the 1-Lipschitz constraint on the loss function L, I deploy the Wasserstein GAN Gradient Penalty (WGAN-GP) framework, introduced by Gulrajani et al. (2017). Following Proposition 1, convex combinations between samples of the real and generated distributions have a gradient norm equal to 1. Adding a term that increases the discriminator loss if $||\nabla_{\hat{x}} D(\hat{x})||$ deviates from 1, with $\hat{x}$ a convex combination between a real and generated sample, therefore enforces the 1-Lipschitz constraint. This yields the following expression for the discriminator loss function:

$$L = E_{\bar{x} \sim P_g}[D(\bar{x})] - E_{x \sim P_r}[D(x)] + \lambda E_{\hat{x} \sim P_{\hat{x}}}[(||\nabla_{\hat{x}} D(\hat{x})|| - 1)^2], \tag{7}$$

where $(||\nabla_{\hat{x}} D(\hat{x})|| - 1)^2$ represents the regularization term that penalizes a deviation of the norm of the gradient of the discriminator loss with respect to its input from 1. $\hat{x}$ reads $\hat{x} = \epsilon x + (1 - \epsilon)\bar{x}$ with $\epsilon \sim U[0, 1]$.

Proposition 1 *(Gulrajani et al., 2017) Let P_r and P_g be two distributions in $\mathcal{X}$, a compact metric space. Then there is a 1-Lipschitz function f^* which is the optimal solution of* $\max_{||f||_L \leq 1} E_{x \sim P_r}[f(x)] - E_{\bar{x} \sim P_g}[f(\bar{x})]$. *Let π be the optimal coupling between P_r and P_g defined as the minimizer of* $W(P_r, P_g) = \inf_{\pi \in \Pi(P_r, P_g)} E_{(x,\bar{x}) \sim \pi}[||x - \bar{x}||]$ *where $\Pi(P_r, P_g)$ is the set of joint distributions $\pi(x, \bar{x})$ whose marginals are P_r and P_g, respectively. Then, if f^* is differentiable, $\pi(x = \bar{x}) = 0$, and $\hat{x} = \epsilon x + (1 - \epsilon)\bar{x}$ with $0 \leq \epsilon \leq 1$, it holds that* $P_{(x,\bar{x}) \sim \pi}[\nabla f^*(\hat{x}) = \frac{x - \hat{x}}{||x - \hat{x}||}] = 1$.

As Gulrajani et al. (2017) found that a two-sided penalty on the gradient norm (i.e. the discriminator loss gradient norm should go towards 1 rather than staying below 1) increases the performance of the WGAN-GP discriminator loss, I also apply a two-sided penalty in this work. λ is set to 10 (Gulrajani et al., 2017).

The main difference between the GAN and the WGAN is the function of the discriminator. In the GAN, the discriminator classifies samples as being real or fake. On the contrary, in the WGAN framework, the discriminator scores the realness of fakeness of a given input sample. Inspired by the reinforcement learning community, the discriminator is often renamed to the critic, as in reinforcement learning the loss function which measures how good the input is, is called the critic. In this research, I will use the discriminator specification.

Training the generator aims at minimizing the distance between the real data distribution and the generative distribution. Therefore, scoring the realness and fakeness of input better describes the overall goal of the network, as the feedback the WGAN discriminator provides is more constructive with respect to the feedback from its GAN counterpart. Arjovsky et al. (2017) show that this yields a training process more stable and less sensitive to the model architecture and the hyperparameter settings. Translating this into Neural Network layers, this means that the last layer in the WGAN discriminator network is a Dense layer with a linear activation function rather than the Sigmoid activiation function as with GANs. To give a concrete example: a GAN discriminator would say "I am 95% confident this is a sample coming from the real data", whereas the WGAN discriminator would say "This sample looks 5 times better than the previous fake input sample, even

though it still looks 7 times worse than a real input sample". Therefore, as the WGAN discriminator describes the process of improvement with respect to previous generative values, the WGAN discriminator is better able to teach the generator how it should adjust its parameters to improve the quality of its generated samples.

4.1.3 Model Setup

The main idea of a Neural Network is to create a mapping between an input and an output layer, through a series of hidden layers. Each layer consists of neurons. The neurons apply mathematical operations on the inputs they receive and subsequently push the result of this operation through the implementation of an activation function onto the next layer. In this research, I consider two types of layers: the Dense Layer and the Convolutional layer. The relation between the neurons in Dense layers described by:

$$\begin{aligned} \gamma_j^l &= \sum_{i=1}^{M^{l-1}} w_{ij}^l a_i^{l-1} + b_j^l \qquad m \in \{1, ..., M^l\} \;\; l \in \{1, .., L\} \\ a_j^l &= f(\gamma_j^l) \qquad\qquad\qquad\quad m \in \{1, ..., M^l\} \;\; l \in \{1, .., L\}, \end{aligned} \tag{8}$$

where the subscript j and superscript l reflect that we are looking at the jth neuron in the l^{th} layer, with l ranging from 1 to L, the number of layers, and j ranging from 1 to M^l, the number of neurons in layer l. a is the activation value of a neuron, w the weights by which activation values from the previous layer get multiplied, and b the bias term. For Convolutional layers, the relation between neurons can be described by:

$$\begin{aligned} \gamma_j^l &= \sum_{i=1+s(j-1)}^{k+s(j-1)} \text{conv1D}(w_i^l, a_i^{l-1}) + b_j^l \qquad m \in \{1, ..., M^l\} \;\; l \in \{1, .., L\} \\ a_j^l &= f(\gamma_j^l) \qquad\qquad\qquad\qquad\qquad\quad m \in \{1, ..., M^l\} \;\; l \in \{1, .., L\}, \end{aligned} \tag{9}$$

where conv1D denotes the one-dimensional convolution operation, s the stride, and k the kernel size. One-dimensional Convolutional Neural Networks (CNNs) are an effective tool to process time series and outperform other network architectures (e.g. LSTM) in terms of both quality of the results and network performance (Selvin et al., 2017). Therefore, in this research, I process the time series using CNNs, in which the networks process the time information by convolving along the time dimension. Equation 9 differs on three important aspects from Equation 8. Firstly, we have an extra operation on the input weights and values, i.e. the convolution operation. Secondly, we sum over k, the kernel

size, instead of n, the number of neurons in the previous layer: whereas in Equation 8 we see that each neuron from the previous layer is connected to the next layer (i.e. fully connected), in CNNs each neuron in the next layer is connected to a small region of the previous layer. The size of this region is described by the kernel size. The next layer is formed by sliding the kernel over the previous layer. The number of neurons that this window gets displaced in each iteration is called the stride. Thirdly, the weights are independent of the input neuron, i.e. weights are shared among neurons. Weight sharing reduce computational costs as we have lesser weights to backpropagate on. Applying the kernel in the temporal dimension translates in the ability of the kernel to detect structures across time.

In Figure 1a, you can see a schematic overview of the generator. The $\boldsymbol{X}_G$-matrix of the generator has dimensions [n x k], where n denotes the number of observations and k denotes the number of assets in the portfolio. In this research, I take historical information up to 40 days ($n = 40$) and I consider seven assets in both portfolios ($k = 7$). Inspired by Mariani et al. (2019), I construct the generator consisting of two subparts: the conditioning and the simulation part. The conditioning part takes in n normalized daily closing prices of k assets. Optimization techniques work better after normalization, as difficulties arise with raw closing prices. The difference in scales across the stocks' closing prices will cause optimization techniques to assign disproportional high weight values to large input values, causing unstable training. Therefore, in this research, the daily closing prices are normalized to the $[-1, 1]$ using min-max normalization.

I firstly implement a set of one-dimensional CNN layers. The last layer of the conditioning part is a Dense layer with an output dimension of k. The Dense layer output is then concatenated with the excess value (φ) of each asset: $\varphi_i = \frac{P_{i,\max} - P_{i,\min}}{P_{i,\text{mean}}}$ where $P_{i,\max}$, $P_{i,\min}$ and $P_{i,\text{mean}}$ are respectively the maximum, minimum and mean value of the price of asset i during the time window under consideration. I add this excess value to minimize the information loss from the normalization. The conditioning part outputs a vector of length $2k$. This information serves as conditioning information for the simulation part. WGAN generates data from an input of random variables ($\boldsymbol{z}$). In this research, we condition these variables on historical information, i.e.:

$$\hat{\boldsymbol{Y}}_G = G(\boldsymbol{z}|\boldsymbol{X}_G), \tag{10}$$

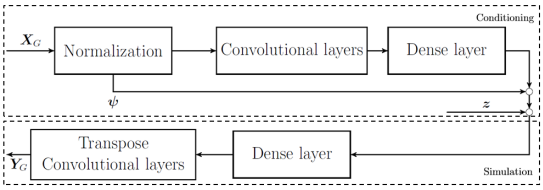
where $\hat{\boldsymbol{Y}}_G$ are the simulated future closing prices by WGAN and $\boldsymbol{z}$ is a stochastic vector

generated from the standard uniform distribution. Applying various transposed one-dimensional convolutional operations in the simulation part on this information, yields the matrix $\hat{\boldsymbol{Y}}_G$ of future closing prices with dimensions [f x k], where f represents the number of future days. In this research, I predict up to 20 days ahead ($f = 20$). I make 250 simulations for future closing prices ($S = 250$), thereby generating future closing prices under various scenarios captured by $\boldsymbol{z}$. Thus, for each simulation, a new sample is drawn from the standard uniform distribution. In Figure 1b, you can see a schematic overview the discriminator. The discriminator uses either a matrix that fully consists of real market data ($\boldsymbol{X}_D$) or a matrix that consists of a concatenation of real market data, and synthetic data generated by the generator ($\hat{\boldsymbol{X}}_D$). The discriminator deploys a CNN architecture to score the realness and fakeness of the input data. Following the WGAN-GP framework the discriminator loss is described by Equation 7.

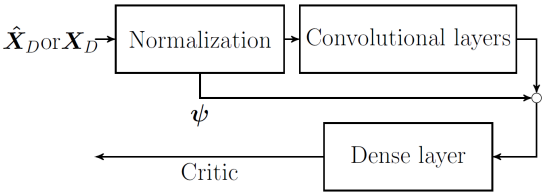
From Equation 8 and Equation 9, you can see that each neuron requires an activation function a. Inspired by Goodfellow et al. (2016), I implement the ReLU ($\max(0, x)$) activation function for the neurons in the generator and the Leaky ReLU activation function for the neurons in the discriminator. Goodfellow et al. (2016) argue that the ReLU activation function is an appropriate default choice as the ReLU is easily optimized with a large and consistent derivative at its active points and a second derivative equal to zero in almost all situations. However, with ReLU we often face the Dying ReLU problem, i.e. by construction, outputs are equal to zero for each input ≤ 0. For these neurons, there are no gradient flows, and thus network performance gets affected if the number of *dead* neurons is large. To alleviate the Dying ReLU problem, the Leaky ReLU activation function is implemented, for which the slope is set to a for $x \leq 0$, causing a leak that extends the range of ReLU. An elaborative description of the architecture is described in Appendix C.

The training procedure of the WGAN-GP framework is described in Algorithm 1. I train the network for 11 hours straight. Optimizing the discriminator for each optimization of the generator is computationally prohibitive and would result in overfitting, i.e. the generator should not be trained too much without updating the discriminator. In this research, the number of discriminator iterations per generator iteration, n_{discr}, is set to 5 (Gulrajani et al., 2017).

The weights and biases for each neuron in a Neural Network are optimized through



(a) Schematic representation of the generator. The generator consists of the conditioning and simulation part.



(b) Schematic representation of the discriminator

Figure 1. Schematic overviews of the generator and the discriminator

minimizing the loss function. This loss function is set by the researcher and is minimized using the backpropagation (BP) algorithm (Leung & Haykin, 1991). The BP algorithm consists of two steps. In the first step, the output is calculated using the layered architecture, the weights with their input values, and biases. In the second step, the errors are calculated and propagated back to the earlier layers. The weights are updated accordingly. The optimization problem in Neural Networks is non-convex, meaning that the loss function does not have a global optimum. The absence of a global optimum makes the optimization complex, as you have to distinguish between local and global optima. Moreover, as the optimization problem in Neural Networks is high-dimensional, the probability of getting stuck in saddle points is substantial. Saddle points are more problematic than local minima, as local minima can yield a relatively good solution, whereas saddle points are surrounded by small gradients such that the BP algorithm is not able to update its parameters. To converge to the global optimum, the optimization technique plays an important role. For example, you can include momentum and an adaptive learning rate in the optimization algorithm. By including momentum, the optimization process is speed up by taking the average gradient with respect to previous gradients. By including adaptive learning rates, the optimization process performs updates depending on the importance of each parameter. Adam optimization is an adaptive moment estimation optimization technique that computes adaptive learning rates for each parameter and includes momentum by keeping an exponentially decaying average of past gradients. In this research, I use the Adam optimizer with learning rate 2×10^{-5}, and $\beta_1 = 0.5$, $\beta_2 = 0.9$ (Mariani et al. (2019), Gulrajani et al. (2017)). In Appendix B, you can find an extensive explanation of this optimization procedure.

A huge advantage of the WGAN-GP architecture is its ability to deploy a stable

training process that is robust under various hyperparameter settings (Gulrajani et al., 2017). In this work, I follow the hyperparameter settings proposed in previous research. For example, the number of Convolutional layers is set to 4 and 5 for the generator and the discriminator respectively, as proposed by (Mariani et al., 2019) and extensively described in Appendix C.

During training, the discriminator provides feedback to the generator, with the objective to minimize the Wasserstein distance between generated samples $\bar{x}$ and real samples x, conditioned on historical information $\boldsymbol{X}_G$. In other words, during training the generator learns the probability distribution of the data through receiving feedback from the discriminator. After training, I simulate 250 runs from the standard uniform distribution and pass these through the generator, thereby simulating possible future scenarios the closing prices for each scenario. Traditional econometric methods rely on restricted assumptions on the noise terms of the asset returns, and do not succeed at predicting closing prices for different assets in one simulation run, while capturing non-linear iterations between the different assets. WGAN, however, does not pose restricted assumptions on the asset return distributions, captures the stochastic component (i.e. noise) that comes into play when predicting asset' closing prices without imposing restricted assumptions on the behavior of this noise, and considers interactions between the assets through the implementation of both vanilla as well as transpose Convolutional layers.

4.1.4 Robustness Check

The superior portfolio performance that can be achieved by implementing GANs for the construction of future asset prices does not come for free, as GANs structures tend to be unstable due to the nonlinear dynamics in the training algorithm (see Algorithm 1), causing adversarial training often failing to converge towards an equilibrium (Mariani et al., 2019). Moreover, the results depend on the weight initialization, which is randomly set. Traditional robustness checks (e.g. by Mariani et al. (2019)) include multiple training iterations and subsequently evaluating the median portfolio performance. However, due to the limited capacity of my local engine, I propose an alternative, however suboptimal, check. The check comprises of evaluating the forecasting performance of WGAN on 3 sets: the original test set, the test set with closing prices +10%, and the test set with closing prices −10%.

Algorithm 1: WGAN-GP with Adam

Set $\lambda = 10$, $n_{discr} = 5$, $\alpha = 2e^{-5}$, $\beta_1 = 0.5$, $\beta_2 = 0.9$

Initialize w_0 (discriminator parameters) and θ_0 (generator parameters)

while θ *has not converged* **do**

  for $t = 1, ..., n_{discr}$ **do**

    for $i = 1, ..., m$ **do**

      Sample real data $\boldsymbol{x} \sim P_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$

      $\bar{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z}|\boldsymbol{X}_G)$

      $\hat{\boldsymbol{x}} \leftarrow \epsilon\boldsymbol{x} + (1 - \epsilon)\bar{\boldsymbol{x}}$

      $Ł^{(i)} \leftarrow D_w(\bar{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(||\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})|| - 1)^2$

    end

    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} Ł^{(i)}, w, \alpha, \beta_1, \beta_2)$

  end

  Sample the latent variables for the next iteration $z^{(i)} \sim p(\boldsymbol{z})$ for $i = 1, ..., m$

  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$

end

4.2 Portfolio Construction

4.2.1 Multiobjective Optimization Problem

In this research, the criteria for asset allocation are mean and semivariance. The mean illustrates the expected return, and the semivariance serves as an approximation for risk. The semivariance only considers adverse deviations and is defined as (Markowitz, 1991):

$$S_{P,B} = E[\min(0, R_P - C)^2] = \frac{1}{T}\sum_{t=1}^{T}[\min(0, R_{P,t} - B)]^2, \tag{11}$$

where R_P is the portfolio return, B is a benchmark, and T is the number of periods. In this research, I call a set of weights that is assigned to the portfolio constituents a diversification. Once we know the diversification x, we can calculate the portfolio return as follows:

$$R_{P,t}(\boldsymbol{w}_x, \hat{\boldsymbol{Y}}_G) = \boldsymbol{w}_x' R_t(\hat{\boldsymbol{Y}}_G), \tag{12}$$

where $R_{P,t}$ is the overall portfolio return at day t, R_t is the return vector that includes the returns of the assets at day t, determined by $\hat{\boldsymbol{Y}}_G$, the matrix with closing prices predicted by WGAN (Equation 10), and $\boldsymbol{w}_x = (w_{1,x}, w_{2,x}, ..., w_{k,x})$ a vector of weights for diversification x. From Equation 11 and Equation 12, we see that the semivariance is endogeneous, i.e. it depends on the weights given to each assets, and a change in weights leads to a change in periods during which the portfolio return under- and outperforms the benchmark B.

WGAN yields a set of future closing prices simulations, N. I calculate the daily return of the assets ($R_{i,t}(\hat{\boldsymbol{Y}}_G)$) for each simulation by $\frac{P_{i,t}-P_{i,t-1}}{P_{i,t-1}}$, where $P_{i,t}$ is the closing price of asset i at day t, extracted from the matrix $\hat{Y}_G$. Calculating the daily returns yields an $[(f-1)$ x $k]$ matrix, as in this research, I forecast asset prices for a time window of f days for k assets.

Taking into account the return and risk objectives, this yields the following multiobjective optimization problem (Macedo et al., 2017):

$$\max E[\boldsymbol{w}_x|R_P] = \sum_{i=1}^{k} w_{i,x} E[R_i] = \sum_{t=1}^{T}\sum_{i=1}^{k} w_i \frac{R_{i,t}}{T} \tag{13}$$

$$\min S(\boldsymbol{w}_x|R_P) = \sum_{i=1}^{k}\sum_{j=1}^{k} w_{i,x} w_{j,x} S_{P,B} \tag{14}$$

subject to:

$$w_{i,x} \geq 0,\ i \in \{1, ..., k\} \qquad \sum_{i=1}^{k} w_{i,x} = 1 \tag{15}$$

where P stands for portfolio, $w_{i,x}$ is the weight for asset i for diversification x in the portfolio, R_P is the portfolio return, $R_{i,t}$ is the return of asset i in simulation for day t, and k is the number of assets, S is the semivariance as defined in Equation 11, and T is the total amount of days. Here, I take $B = E[R_{P,f}]$, and $T = f$, the number of days I forecast ahead.

To solve the multiobjective problem, I implement a multiobjective evolutionary algorithm (MOEA), as this overcomes the endogeneity problem introduced by the semivariance (Equation 11) by construction. The MOEA algorithm calculates the semivariance for a large number of iterations. In each iteration, the weights are known *a priori* as they are already generated by the algorithm in the previous iteration. Knowing these weights, we can calculate the portfolio return and implement this in Equation 11 (Macedo et al., 2017).

The shape of the mean-semivariance frontier is comparatively straightforward, as there are only two restrictions on the weights, i.e. portfolio weights are non-negative and should add up to one (Equation 15). Therefore, for this frontier, there is a strong use case for using the Non-Dominated Sorting Genetic Algorithm II (NSGA-II).

NGSA-II yields a diversification strategy based on all the WGAN asset price simulations S, i.e. a set of weights with Pareto-optimal diversifications, $x \in X(S)$, with the number of WGAN simulations S equals 250. This strategy presents a set of diversifications that tradeoff risk and return, i.e. a higher level of risk yields a higher level of expected return. The overall portfolio return for diversification x can be calculated by Equation 12. It is up to the investor which diversification strategy to choose, based on its risk preferences. With MOEAs it is important to verify whether the frontier that is found by the algorithm actually is the optimal frontier. To verify this, I perform a Monte Carlo (MC) simulation that generates $3,000$ additional weights elaborating on the weights given by the NSGA-II algorithm: $1,000$ weights elaborate on the weights of the riskiest portfolio, $1,000$ weights elaborate on the weights of the most risk-averse portfolio, and $1,000$ weights elaborate on a random point on the frontier.

4.2.2 Model Evaluation

The goal of the methodology described is to come up with a diversification that earns the highest portfolio return possible for a certain level of risk. To evaluate the model, I introduce two investment approaches and track their associated returns. The first investment approach is considered to be the risk-averse approach, i.e. the approach for which the investor prefers a low-risk portfolio. The second approach is the risk-seeking approach. The risk-seeking investor has a high-risk tolerance and desires to earn superior returns. Inspired by Mariani et al. (2019), I quantify both approaches by introducing a set of risk levels $\eta \in [1, ..., Z]$, where $Z = 12$. Here, $Z = 1$ belongs to the most risk-averse strategy, and $Z = 12$ belongs to the riskiest strategy. For the risk-averse approach, I set $\eta = 2$ and for the risk-seeking approach, I set $\eta = 10$. I then identify the risk-averse and risk-seeking portfolio based on the returns on the optimal frontiers found by NSGA-II and MC, i.e. the corresponding return for each risk-level is calculated by:

$$R(\eta) = R_{\min} + \frac{\eta(R_{\max} - R_{\min})}{Z}, \tag{16}$$

where $R(\eta)$ is the return for risk level η and $R_{\min}$ and $R_{\max}$ are the minimum and maximum return levels on the frontiers, respectively. I consider a 20-day (1 month) investment horizon.

I compare these portfolio returns against the index returns for each portfolio (i.e. JSE All Share Index for the South African allocation and MOEX Russia Total Return Index for the Russian allocation), and against a benchmark portfolio. The benchmark is the $\frac{1}{N}$-portfolio. This portfolio assigns equal weight to each asset in the portfolio. The $\frac{1}{N}$-portfolio has a good track record regarding out-of-sample performance in the traditional mean-variance framework. DeMiguel et al. (2009) investigate the out-of-sample performance for fourteen parametric models that implement the mean-variance framework, and compares their performance with the $\frac{1}{N}$-portfolio in terms of Sharpe ratio and turnover. They conclude that none of these models consistently outperform the $\frac{1}{N}$-portfolio.

5 Results

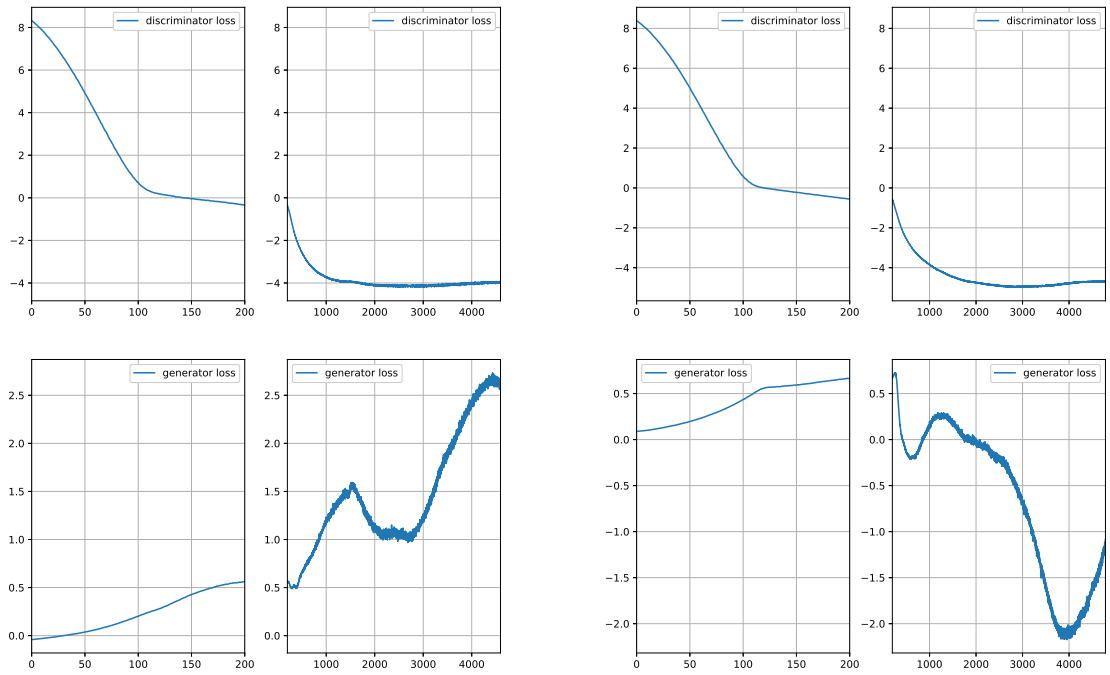
In this section, I firstly provide the forecasting results of the deep learning architecture, i.e. the Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP). Thereafter, I elaborate on the fronts constructed by the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) and the Monte Carlo (MC) simulations. To conclude, I document the performance of the risk-averse and risk-seeking portfolios.

5.1 Deep Learning Architecture

The WGAN-GP framework yields losses for the generator and the discriminator. As explained in Section 4, the discriminator implements the Wasserstein distance. The discriminator aims to correctly distinguishing between real samples and fake samples. It assigns positive values for real samples and negative values for fake samples. The overall discriminator loss is given in Equation 7 and can be summarized as $D_{\text{loss}} = f_{\text{loss}} - r_{\text{loss}} + \textit{penalty}$, where f_{loss} represents the discriminator output for fake samples, r_{loss} represents the discriminator output for real samples, and the penalty term represents a regularization to enforce the Lipschitz constraint, as described in Section 4. From Figure 2a and Figure 2b, you can see that for both the Russian and the South African data, the discriminator loss is stabilized and minimized. For the Russian data, the loss stabilizes at around -4, for the South African data at around -5. The left Figure of Figure 2a and Figure 2b represent the losses for the first 200 training iterations, the right Figure of 2a and Figure 2b represent the losses from training iteration 200 onwards.

A positive discriminator value for fake samples (f_{loss}) can mean two things: either the generator succeeds at generating high-quality samples such that it becomes extremely hard for the discriminator to assign a negative values to these samples, or the discriminator is not well-trained, meaning that it simple cannot distinguish between real and fake samples because of its low distinguishing power.

The generator loss is calculated as $G_{\text{loss}} = -f_{\text{loss}}$. From Figure 2a, you can see that for the Russian assets, the generator loss is increasing and ends at around 2.5. This means the generator does not accomplish to generate samples that are characterized as real by the discriminator. Therefore, either the generator is bad-performing or the discriminator is extremely powerful. For the South African data, the generator loss becomes negative,



(a) The Russian asset data. The discriminator loss is stabilized and minimized at around -4. The generator loss is increasing, and has a value of around 2.5 at the end of training.

(b) The South African asset data. The discriminator loss is stabilized and minimized at around -5. The generator loss is decreasing, and has a value of around -1 at the end of training.

Figure 2. The discriminator and generator loss for the Russian and South African asset data. The left Figures represent the losses for the first 200 training iterations, the right Figures represent the losses from training iteration 200 onwards.

ending at -1, meaning that the discriminator assigns positive values (so real sample values) to fake samples. Therefore, in this market, either the generator succeeds at generating high-quality samples, or the discriminator is not performing well. However, for both markets, the generator loss is not very reliable as it exhibits large swings and is not stabilized at the end of the training period.

The penalty term for both the South African as well as the Russian data is almost equal to 0, meaning that in the discriminator is (almost) 1-Lipschitz continuous.

In Figure 3a and Figure 3b, you can see the predictions of WGAN for the closing prices of the African and Russian stocks. I run 250 simulations, of which the average across all these simulations is depicted here. The predictions are up to 20 days ahead, i.e. to

construct these plots I have concatenated 21 time frames each consisting of 20 days. This results in 420 observations which cover the period 2017-09 to 2019-04. From the plots, we see that WGAN succeeds at predicting the general pattern of the closing prices. Moreover, we see that the WGAN underestimates the variability of the closing prices, i.e. during a period of heavy fluctuations the WGAN chooses to take a conservative view and aims at predicting the correct closing prices at the end of its forecasting period. This behavior is better visible if we look at the normalized forecasted closing prices in Figure 4a and Figure 4b. I depict 3 randomly chosen simulations from the 250 simulations performed. You can see that for the South Africa data, WGAN is immune to large price shocks and most of the times yields a final normalized price level that accurately represents the realized normalized final level. For the Russia data, we see that for one scenario, WGAN predicts a large swing upwards, and subsequently predicts more conservative values. This also explains the low predicted semivariance values in Table 3a and Table 3b, i.e. the low predicted semivariances indicates that the algorithm underestimates the closing price variability.

(a) *The South African assets*

Semivariance / Stock	ARM	AMS	ANG	GFI	KIO	ASR	ALSI
Historical ($\times 10^{-3}$)	12.3	7.32	9.12	11.2	12.9	14.7	1.67
Predicted ($\times 10^{-3}$)	1.33	0.48	0.94	0.38	0.76	2.35	0.05
Realized ($\times 10^{-3}$)	5.85	3.70	4.17	5.41	6.19	6.98	0.82

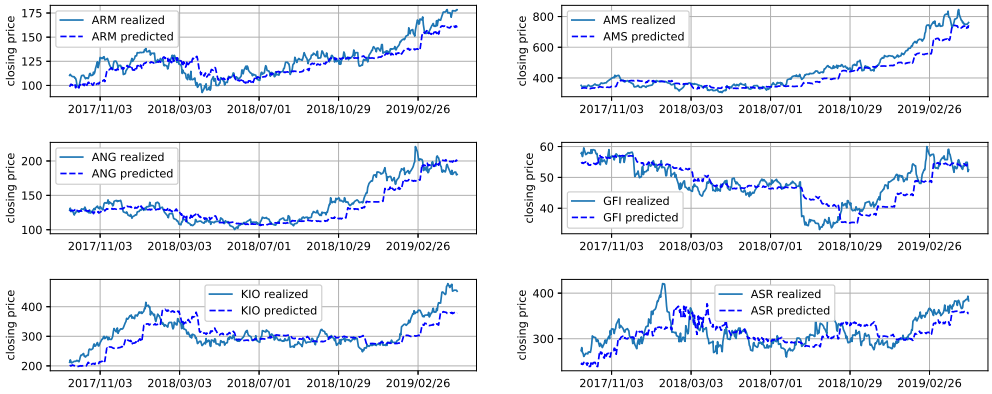
(b) *The Russian assets*

Semivariance / Stock	GAZP	LKOH	NVTK	ROSN	SNGS	TATN	MCFTR
Historical ($\times 10^{-3}$)	3.14	3.32	4.01	3.78	2.56	4.29	1.88
Predicted ($\times 10^{-3}$)	0.25	0.66	0.15	1.11	0.50	0.57	0.27
Realized ($\times 10^{-3}$)	1.58	1.60	1.96	1.83	1.19	1.98	0.90

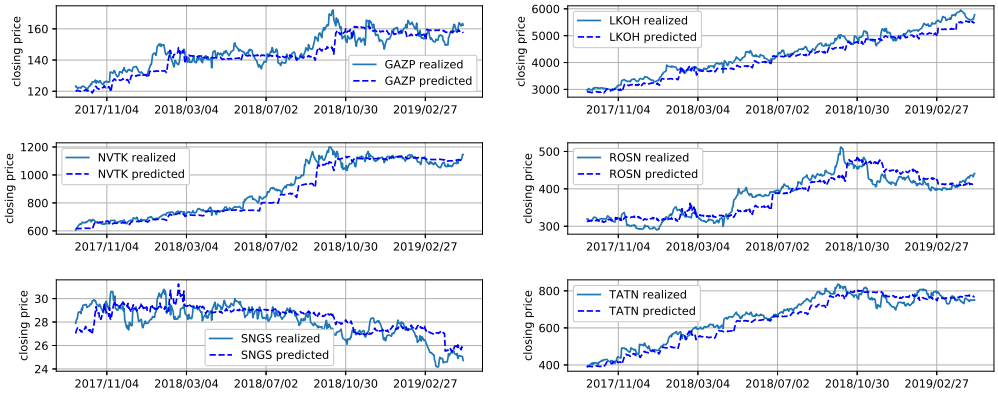
Table 3. *The historical, predicted, and realized semivariance for the assets. WGAN is underestimating the return variability as the predicted semivariance is lower than the realized semivariance.*

Moreover, you can see from Figure 3 that a period of relatively stable closing price predictions is followed by a quite heavy shock upwards or downwards, yielding in a movement resembling a stairs. This behavior can be assigned to the denormalization process rather than to the mechanics of the network. In Section 7, I discuss suggestions to im-

prove the network. These improvements aim at, among others, mitigating the effect of the denormalization process, and, therefore, at reducing the stairs-like appearance of the closing price predictions.

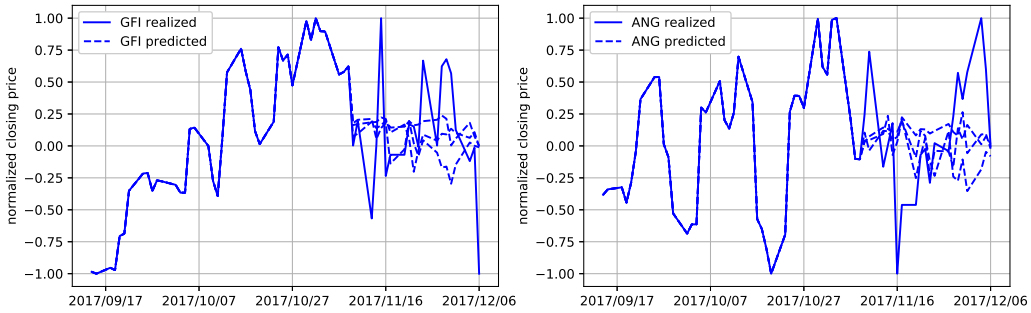


(a) South African stocks in the mining and materials sector

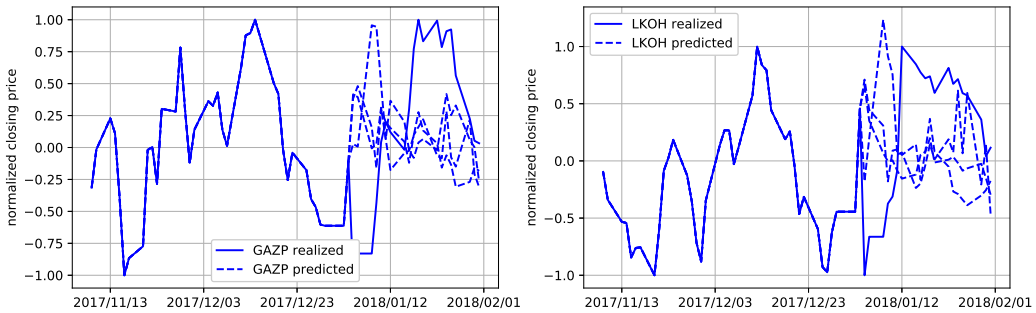


(b) Russian stocks in the oil and gas sector

Figure 3. The realized and 20-day ahead predicted closing prices of the stocks. The predictions cover the period 2017-09 to 2019-04.



(a) Normalized predicted (dashed) and realized (solid) for GFI (left) and ANG (right) covering the period 2017-09-14 to 2017-12-06.



(b) Normalized predicted (dashed) and realized (solid) for GAZP (left) and LKOH (right) covering the period 2017-11-09 to 2018-01-31.

Figure 4. Predicted and realized closing prices for 2 stocks in the South African mining and materials market as well as the Russian oil and gas market.

In this research, WGAN is designed to predict up to 20 days ahead. From Table 4a, you can see that, in general, WGAN is not leaning towards an overly bullish or bearish view on South African assets, as it succeeds at predicting both upwards and downwards movements. However, for ALSI, WGAN is having an bullish view as it predicts $3,650$ upward movement whereas only $2,250$ upward movements are realized. As a result, WGAN performs worst at predicting this constituent with only 44.8% of correctly predicted directions. From Table 4b shows that WGAN is bearish on Russian assets, as it predicts more downward movements than actually realized for all constituents except for SNGS. For SNGS, the number of upward swings predicted approximately equals the number of upward swings realized, however, its timing is wrong approximately 30% of the times. It exhibits the best performance for TATN, with 83.5% correctly predicted movements, which can partly be explained by the large amount of realized downward movements

(3, 248) from which WGAN only missed 36.

To get a better insight in the overall predicting performance for different time horizons of WGAN, you can find the performance for 4 different time horizons, i.e. 1 day, 1 week (5 days), 2 weeks (10 days), and 1 month (20 days), together with the general return characteristics, in Appendix D.

(a) *The South African portfolio*

Constituent	ARM		AMS		ANG		GFI		KIO		ASR		ALSI	
% correct	66.3		61.7		81.2		77.4		71.3		74.6		44.8	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	1,316	835	1,250	508	1,588	323	1,465	403	1,918	675	1,783	614	1,500	2,150
down	684	2,165	1,250	1,992	662	2,677	535	2,597	832	1,825	717	2,136	750	850

(b) *The Russian portfolio*

Constituent	GAZP		LKOH		NVTK		ROSN		SNGS		TATN		MCFTR	
% correct	68.1		52.6		68.5		66.0		60.9		83.5		61.2	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	1,349	275	278	19	845	249	1,444	479	2,004	1,059	1,169	36	802	87
down	901	2,225	1,772	2,481	1,405	2,751	1,306	2,021	996	1,191	831	3,214	1,698	2,413

Table 4. *Characteristics and performance of WGAN per portfolio constituent*

5.1.1 Robustness Check

From Table 5, you can see that WGAN has similar results in terms of the percentage of correctly predicted upward and downward movements when the data is adjusted by adding and subtracting 10% with respect to the original data. The biggest difference is only 5.1%, which is for AMS. For the unadjusted closing prices, WGAN yields a score of 61.7%, whereas for the closing prices that are adjusted +10%, it yields 66.8%.

(a) *The South African assets*

Constituent	ARM	AMS	ANG	GFI	KIO	ASR	ALSI
unadjusted	66.3	61.7	81.2	77.4	71.3	74.6	44.8
+10%	66.3	66.8	81.6	82.2	71.5	74.2	44.8
10%	66.6	61.9	81.7	82.0	71.2	74.7	45.1

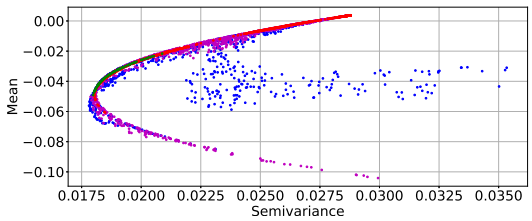
(b) *The Russian assets*

Constituent	GAZP	LKOH	NVTK	ROSN	SNGS	TATN	MCFTR
unadjusted	68.1	52.6	68.5	66.0	60.9	83.5	61.2
+10%	72.4	52.6	68.8	65.7	65.6	83.6	61.0
-10%	72.5	52.5	68.9	66.3	65.8	83.6	61.6

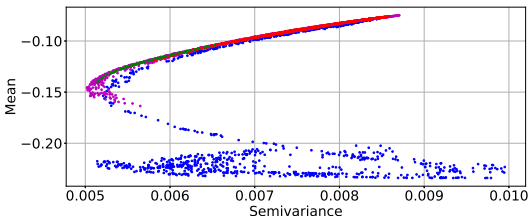
Table 5. *The robustness check for WGAN. The results for data varying between the* -10% *and* $+10\%$ *range are stable.*

5.2 NSGA-II with Monte Carlo

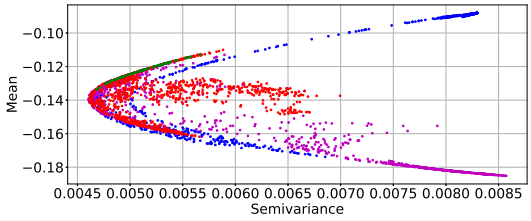
With the Non-Dominated Sorting Genetic Algorithm II (NSGA-II), portfolios are formed for various risk-return preferences by constructing the mean-semivariance frontier. With multiobjective evolutionary algorithms (MOEAs) it is important to verify whether the frontier that is found by the algorithm actually is the optimal frontier. To verify this, I perform a Monte Carlo (MC) simulation. In Figure 5, three different scenarios of the performance of NSGA-II (green) and the MC simulations (blue, purple, red) are shown for the Russian market. MC generates $3,000$ additional weights elaborating on the weights given by the NSGA-II algorithm: $1,000$ elaborate on the weights of the riskiest portfolio (blue), $1,000$ elaborate on the weights of the most risk-averse portfolio (purple), and $1,000$ elaborate on a random point on the frontier (red). In all scenarios, the frontier that is constructed by NSGA-II is optimal. However, NSGA-II does not yield the entire optimal frontier. Therefore, when constructing the portfolios, I include the MC portfolios that extend the optimal frontier. In Figure 6, you can find the frontiers from Figure 5 that are obtained after a careful selection that combines NSGA-II and MC portfolios. In Appendix F, you can find the full set of frontiers. I obtain frontiers for every 20 days in the test period, as WGAN is designed to predict up to 20 days ahead. This yields 42 frontiers: 21 for the South African allocation and 21 for the Russian allocation. The mean return on the frontiers is small and often negative. This is due to both that WGAN



(a) Example 1: the frontier for 2017-10. The NSGA-II frontier is extended, the inefficient part is formed, and inefficient portfolios are formed within the efficient frontier.



(b) Example 2: the frontier for 2018-04. The NSGA-II frontier is extended and inferior frontiers are formed.



(c) Example 3: the frontier for 2018-11. The NSGA-II frontier is extended, the inefficient part is formed, and inefficient portfolios are formed within the efficient frontier.

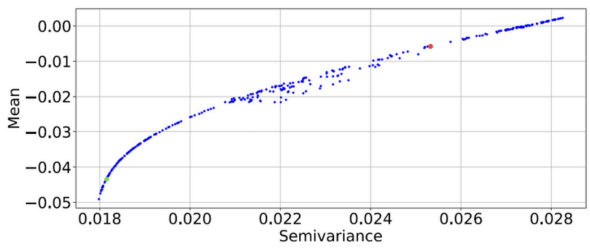
Figure 5. Three mean-semivariance frontiers constructed by NSGA-II (green) and extended by MC (red, purple, blue). The MC simulations confirm that the frontier that is found by NSGA-II is optimal but not complete.

underestimates the return variability (Table 3b) as well as the bearish view of WGAN on the Russian stock market (Table 4b).

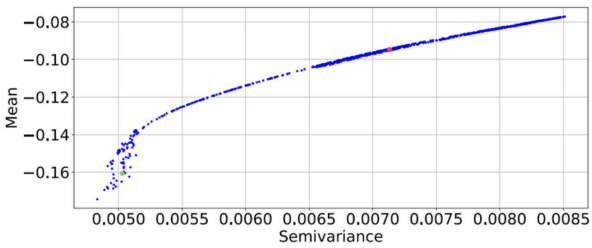
5.3 Risk-Averse and Risk-Seeking Portfolios

In this research, I construct a portfolio for two types of investors: the risk-averse and risk-seeking investor. On a scale from 1 to 12, where a higher number reflects a higher risk tolerance, the risk-averse investor has a risk tolerance of 2, whereas the risk-seeking investor has a risk tolerance of 10.

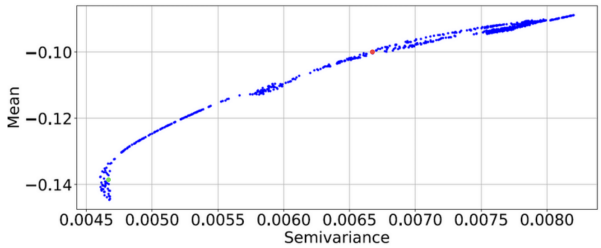
From Figure 6, you can see that on the frontier, the risk-seeking investor chooses the portfolio with a relatively high semivariance, whereas the risk-averse investor chooses the



(a) Front 1: covers 2017-10.



(b) Front 2: covers 2018-04.



(c) Front 3: covers 2018-11.

Figure 6. Three final frontiers. The risk-averse investor (green) chooses a low-risk portfolio and agrees upon lower returns. The risk-seeking investor (red) aims to earn superior returns. The total set of final frontiers can be found in Appendix F.

one with the lower semivariance, thereby also accepting a lower return.

In Figure 7, you can see the stock allocation for both investors in the South African and Russian stock market. From Figures 7a and 7b, you can see that the algorithm assigns a higher index weight to the risk-seeking investor compared to the risk-averse investor. This is because of its attractive risk-return characteristics predicted by WGAN. From Table 4a, you can see that WGAN is extremely bullish on the index constituent (i.e. ALSI), with $3,650$ predicted upward movements from which only $2,250$ are realized. ARM is also predicted to have attractive risk-return characteristics. It predicts $2,151$ upward movements, from which $2,000$ are realized. Moreover, its semivariance is underestimated by 78%, resulting in a semivariance that equals 1.33×10^{-3}. This explains why the algorithm favors both ALSI as well as ARM.

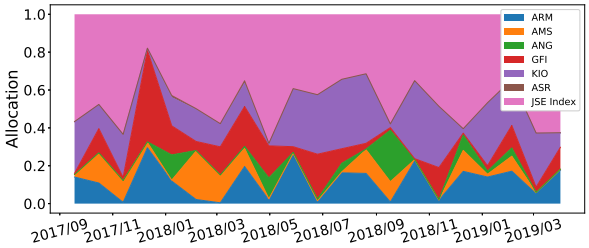
For both investors, the algorithm barely assigns weight to ASR. From Table 3a and Table 4a, it becomes clear that the algorithm, on one hand, succeeds at identifying the high associated downward risk (with a predicted semivariance only approximately 3 times smaller than the realized semivariance), and on the other hand, that it has a bearish view on this stock (with $2,853$ predicted downward movements, and only $2,750$ realized).

WGAN considers ASR as an unattrative investment and therefore allocates almost no wealth to this stock during the entire investment period.

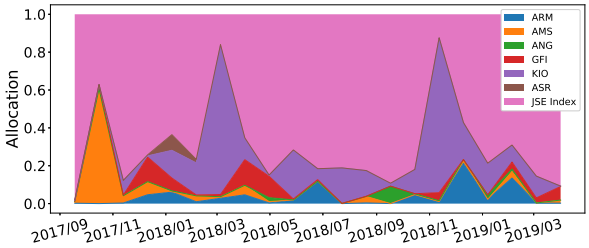
The view that the algorithm constructs turns out to be a profitable one but it is insignificantly underperforming with respect to the $\frac{1}{N}$-portfolio (Table 6a): following this allocation leads to a compounded return of 48% (t-statistic = -0.22) for the risk-averse investor and 5% (t-statistic = -1.51) for the risk-seeking investor, whereas the $\frac{1}{N}$-portfolio yields 52% compounded return. The risk-averse portfolio does outperform the JSE All Share Index, but also insignificantly with a t-statistic of 1.45 (Figure 8a). The risk-seeking investor underperforms with respect to the local index, but not significantly as the corresponding t-statistic equals -0.01.

Considering the allocations in the Russian market (Figure 7c and Figure 7d), I would like to highlight a few interesting results. First of all, it becomes clear that WGAN allocates an enormous part to NVTK for the risk-seeking investor, whereas for the risk-averse investor it rather divides most of its allocation among NVTK, SNGS, and GAZP. However, the risk-averse allocation is versatile. For example, 70% of the allocation was assigned to GAZP in 2018-10, whereas the position is reduced to almost 0% in the consecutive month. The huge allocation for the risk-seeking investor to NVTK is driven by the low predicted overall semivariance (0.15×10^{-3}, see Table 3b), which results in a attractive risk-return characteristics. Although NVTK has the lowest forecasted semivariance, assigning a large chunck of your allocation to a single stock is a risky move. The choice for LKOH and GAZP for the risk-averse investor is rather interesting. LKOH has the highest rate of wrongly predicted downward movements, indicating that WGAN is extremely bearish on this stock. GAZP has worse risk-return performance compared to NVTK, as its perceived risk is almost twice as big (2.51×10^{-2} see Table 3b) as for NVTK whereas the amount of predicted upward movements is only 1.5 as big as for NVTK (see Table 4b). However, Table 3 and Table 4b only show general characteristics, not period specific. As the LKOH and GAZP allocations vary a lot across different periods, it is reasonable to argue that NSGA-II only allocates wealth to these stocks in prosperous times.

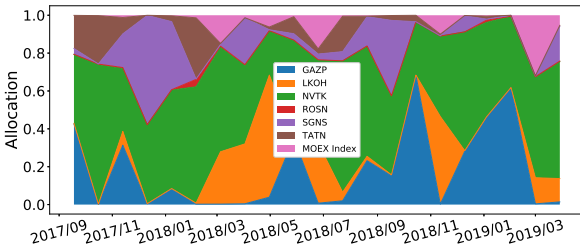
The allocation to SNGS is not stable either, but does not exhibit fluctuations as large as with LKOH and GAZP. With the highest rate of both correctly $(2,004)$ and falsely $(1,059)$ upward predicted movements (Table 4b), indicating that WGAN is bullish on this stock, and a relatively high predicted semivariance (4.96×10^{-2}, Table 3b), it is reasonable



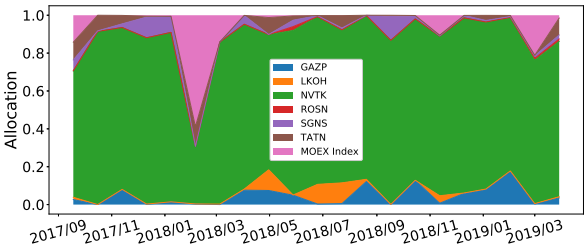
(a) South African portfolio: risk-averse allocation



(b) South African portfolio: risk-seeking allocation



(c) Russian portfolio: risk-averse allocation



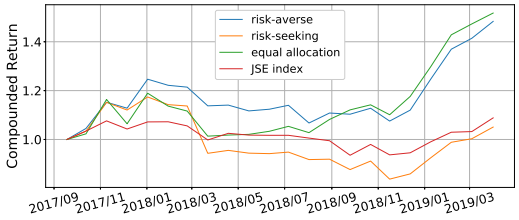
(d) Russian portfolio: risk-seeking allocation

Figure 7. Allocations for the risk-averse and risk-seeking investor in the Russian and South African stock market. The investment period covers 2017-09 to 2019-04.

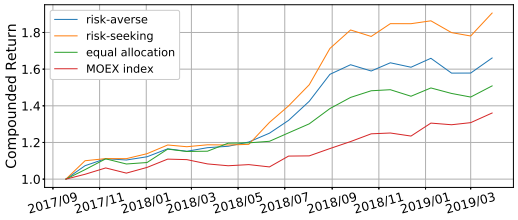
to conclude that NSGA-II carefully assigns parts of the risk-averse portfolio to SNGS.

Moreover, for both investors, it assigns small weights to the index constituent. This can be explained by the extremely bearish view on MCFTR, as WGAN predicts $4,111$ downward swings whereas only $2,500$ are realized (Table 4b).

The risk-averse and risk-seeking portfolios by WGAN for the Russian oil and gas market both outperform the $\frac{1}{N}$-portfolio, by 15% and 40% respectively (Figure 8b). However, the outperformance is not significant, with t-statistics of 0.59 and 1.14 for the risk-averse and risk-seeking investors respectively (Table 6b). The portfolios also insignificantly outperform the local index, with an outperformance of 30% (t-statistic = 1.27) for the risk-averse investor and 55% (t-statistic = 1.68) for the risk-seeking investor.



(a) The South African WGAN risk-averse portfolio outperforms the index by 39%. The $\frac{1}{N}$-portfolio is superior.



(b) The Russian WGAN risk-averse and risk-seeking portfolios outperform the $\frac{1}{N}$-portfolio by 15% and 40% respectively.

Figure 8. The performance of the WGAN risk-averse and risk-seeking portfolios compared to the $\frac{1}{N}$-portfolio and the index.

	Mean	Stdev	Sharpe	t-stat Index	t-stat Benchmark
Risk-Averse	0.0188	0.0532	0.04	1.45	-0.22
Risk-Seeking	0.0020	0.0591	-0.25	-0.01	-1.51
Index	0.0021	0.0319	-0.47	x	-2.77
Benchmark	0.0214	0.0614	0.07	1.44	x

(a) *The South African portfolios*

	Mean	Stdev	Sharpe	t-stat Index	t-stat Benchmark
Risk-Averse	0.0237	0.0357	0.19	1.27	0.59
Risk-Seeking	0.0302	0.0447	0.30	1.68	1.14
Index	0.0138	0.0249	-0.13	x	-0.97
Benchmark	0.0191	0.0293	0.07	0.83	x

(b) *The Russian portfolios*

Table 6. *Return statistics for the risk-averse and risk-seeking allocations, compared with the* $\frac{1}{N}$*-portfolio and the local index. The mean, standard deviation, and Sharpe ratio are for a* 20*-day (i.e.* 1 *month) time scale.*

6 Conclusion

In this research, I propose a novel methodology consisting of a deep learning architecture (WGAN) combined with a multiobjective evolutionary algorithm with Monte Carlo to optimize asset allocation in emerging market space. I find that WGAN succeeds at predicting the general pattern of the closing prices, however, that it underestimates the variability of the closing prices. This is explained by the behavior of WGAN in forecasting periods during which closing prices fluctuate heavily. WGAN then chooses to focus on predicting the correct closing price at the end of the forecasting period, thereby paying less attention to the fluctuations in between. This results in high percentages of correctly predicted up- and downward movements, with up to 81.2% in the South African stock market and 83.5% in the Russian stock market. However, the semivariance is consistently underestimated, having its worst performance for GFI in South Africa, with a predicted semivariance of 0.38 whereas the realized semivariance is 5.41.

Moreover, I find that the risk-averse and risk-seeking portfolio in Russia outperforms both the benchmark as well as the local index by 15% and 40% compounded return respectively for the benchmark and by 30% and 55% respectively for the local index. The outperformance is not significant, with t-statistics for the risk-averse portfolio of 0.59 and 1.27 for the benchmark and local index respectively, and for the risk-seeking portfolio of 1.14 and 1.68. For the South African market, I find that the risk-averse portfolio outperforms the local index with 39% compounded return and that its performance is inferior with respect to the benchmark (-3% compounded return difference). Both results are again insignificant: t-statistics equal 1.45 and -0.22 respectively. The results for the risk-seeking investor are disappointing, with underperformance with respect to both the local index as well as the benchmark. Again, these results are insignificant with t-statistics of -0.01 and -1.51 respectively. Since the risk-seeking investor aims to earn superior returns by taking on extra risk, it is disappointing that it even underperforms with respect to the risk-averse investor. The underperformance can be explained by the huge dependence on the index constituent for this allocation, which performs not as good as WGAN predicted. The insignificance of the results can be explained by the limited size of the data set that was available for the emerging market assets in this research.

7 Discussion

In this section, I would like to discuss my research and suggest topics for further research.

Firstly, I would like to comment on the generator loss of WGAN in both markets. The generator loss has not stabilized by the end of the training period yet, with training covering 11 hours, yielding over $4,000$ training iterations. To extend this research, one could improve WGAN by implementing techniques that aim to improve the training stability, for example by implementing Spectral Normalization (Miyato et al., 2018).

Secondly, increasing attention has been paid to include econometric techniques (e.g. deseasonalization) into Neural Network architectures, the so-called hybrid models. For example, Smyl et al. (2018) introduces the Exponential Smoothing-Recurrent Neural Network (ES-RNN) architecture. With this architecture, he won the renowned Makridakis (M) Competition, one of the most important events in the field of forecasting. What distinguishes the M competition from other forecasting competitions is that developers are challenged to come up with strategies that solely focus on time series forecasting, thereby not considering additional regressors. This perfectly elaborates on this research, as I do not take into account additional regressors either. I encourage further research to focus on extending the architecture proposed in this research with an econometric component to cover seasonal patterns, thereby enhancing forecasting performance.

Besides this, I would like to highlight one last area for further consideration concerning WGAN. In this research, for the random input of the generator, I sample from the standard uniform distribution. From a statistical point of view, this makes sense as the standard uniform distribution is the most universal distribution among all probabilities distributions, and, therefore, eminently deployed to sample from more specific distributions by inverse transform sampling. However, I find that with this distribution the various simulations exhibit similar characteristics and are quite conservative in their predictions. Mariani et al. (2019) sampled from the standard normal distribution and their predictions are less conservative, with simulations representing various extreme market scenarios. One could investigate the effect of different sampling distributions and conclude on a distribution that gives optimal results for emerging market studies.

Moreover, in Section 1, I highlight that one of the main contributions to the existing literature is that the WGAN architecture together with NSGA-II and MC is applied to emerging market data. The emerging market data adds an extra layer of difficulty to this

research, as emerging markets are immature markets, causing the exposure to reliable, historical data to be restricted. In Section 3, I document the conditions the securities should satisfy to be eligible for inclusion in this research, thereby aiming to increase the reliability of the data that is considered in this research. However, to make these conditions feasible, I restrict the period under consideration from 2010-02-01 to 2019-05-31. WGAN requires an extensive training set to minimize the loss function, thereby solving for the optimal network parameters (i.e. the weights and biases). This leads to a reduction in the size of the final test set to only 21 months. Although the results are promising, with outperformance in terms of compounded return of up to 40%, this research fails to get conclusive results. This can largely be explained by the limited range of the test period, and, therefore, I encourage further research to implement data augmentation.

The last point for discussion concerns the multiobjective evolutionary algorithm (MOEA) deployed in this research, i.e. the Non-Dominated Sorting Genetic Algorithm II (NSGA-II). As shown in Section 5, NSGA-II yields the optimal frontier, however it fails to yield the entire frontier. This problem is resolved by performing Monte Carlo (MC) simulations alongside NSGA-II. However, an MOEA that yields the entire optimal frontier directly is preferred. Macedo et al. (2017) find complete and optimal fronts for the mean-semivariance problem using NSGA-II. One could investigate the impact of emerging market data on the performance of NSGA-II and implement its findings accordingly, in order to come up with a superior technique to construct portfolios in emerging market space.

References

Arjovsky, M. & Bottou, L. (2017). Towards principled methods for training generative adversarial networks. arxiv e-prints, art. *arXiv preprint arXiv:1701.04862*.

Arjovsky, M., Chintala, S. & Bottou, L. (2017). Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Bekaert, G., Erb, C. B., Harvey, C. R. & Viskanta, T. E. (1998). Distributional characteristics of emerging market returns and asset allocation. *Journal of portfolio management*, *24*(2), 102–+.

Campbell, R., Huisman, R. & Koedijk, K. (2001). Optimal portfolio selection in a value-at-risk framework. *Journal of Banking & Finance*, *25*(9), 1789–1804.

Chong, E., Han, C. & Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, *83*, 187–205.

DeMiguel, V., Garlappi, L. & Uppal, R. (2009). Optimal versus naive diversification: How inefficient is the 1/n portfolio strategy? *The review of Financial studies*, *22*(5), 1915–1953.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Generative adversarial nets, In *Advances in neural information processing systems*.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. C. (2017). Improved training of wasserstein gans, In *Advances in neural information processing systems*.

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hull, J. C. (2003). *Options futures and other derivatives*. Pearson Education India.

Jiang, Q., Tang, C., Chen, C., Wang, X. & Huang, Q. (2018). Stock price forecast based on lstm neural network, In *International conference on management science and engineering management*. Springer.

Johannesburg Stock Exchange. (2020). Jse market data [(Accessed on 15/04/2020)].

Kara, Y., Boyacioglu, M. A. & Baykan, Ö. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert systems with Applications*, *38*(5), 5311–5319.

Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Leung, H. & Haykin, S. (1991). The complex backpropagation algorithm. *IEEE Transactions on signal processing*, *39*(9), 2101–2104.

Li, H. & Zhang, Q. (2008). Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE transactions on evolutionary computation*, *13*(2), 284–302.

Lim, A. E., Shanthikumar, J. G. & Vahn, G.-Y. (2011). Conditional value-at-risk in portfolio optimization: Coherent but fragile. *Operations Research Letters*, *39*(3), 163–171.

Linsmeier, T. J. & Pearson, N. D. (1996). Risk measurement: An introduction to value at risk, (1629-2016-134959), 45. https://doi.org/10.22004/ag.econ.14796

Macedo, L. L., Godinho, P. & Alves, M. J. (2017). Mean-semivariance portfolio optimization with multiobjective evolutionary algorithms and technical analysis rules. *Expert Systems with Applications*, *79*, 33–43.

Mariani, G., Zhu, Y., Li, J., Scheidegger, F., Istrate, R., Bekas, C. & Malossi, A. C. I. (2019). Pagan: Portfolio analysis with generative adversarial networks. *arXiv preprint arXiv:1909.10578*.

Markowitz, H. (1952). Modern portfolio theory. *Journal of Finance*, *7*(11), 77–91.

Markowitz, H. (1991). Foundations of portfolio theory. *The journal of finance*, *46*(2), 469–477.

Markowitz, H. (2010). Portfolio theory: As i still see it. *Annu. Rev. Financ. Econ.*, *2*(1), 1–23.

Markowitz, H., Todd, P., Xu, G. & Yamane, Y. (1993). Computation of mean-semivariance efficient sets by the critical line algorithm. *Annals of Operations Research*, *45*(1), 307–317.

Miyato, T., Kataoka, T., Koyama, M. & Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Moscow Exchange. (2020). Moex russia total return indices [(Accessed on 11/04/2020)].

Salakhutdinov, R. & Hinton, G. (2009). Deep boltzmann machines, In *Artificial intelligence and statistics*.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E., Menon, V. K. & Soman, K. (2017). Stock price prediction using lstm, rnn and cnn-sliding window model, In *2017 international conference on advances in computing, communications and informatics (icacci)*. IEEE.

Smyl, S., Ranganathan, J. & Pasqua, A. (2018). M4 forecasting competition: Introducing a new hybrid es-rnn model. *URL: https://eng. uber. com/m4-forecasting-competition*.

Tobin, J. (1958). Liquidity preference as behavior towards risk. *The review of economic studies*, *25*(2), 65–86.

Wharton Research Data Services. (2020). Compustat - capital iq [(Accessed on 11/04/2020)].

Zhang, H., Zhao, Y., Wang, F., Zhang, A., Yang, P. & Shen, X. (2018). A new evolutionary algorithm based on moea/d for portfolio optimization, In *2018 tenth international conference on advanced computational intelligence (icaci)*. IEEE.

Zhang, K., Zhong, G., Dong, J., Wang, S. & Wang, Y. (2019). Stock market prediction based on generative adversarial network. *Procedia computer science*, *147*, 400–406.

Zhang, M., Nan, J. & Yuan, G. (2012). The geometric portfolio optimization with semivariance in financial engineering. *Systems Engineering Procedia*, *3*, 217–221.

Zhou, X., Pan, Z., Hu, G., Tang, S. & Zhao, C. (2018). Stock market prediction on high-frequency data using generative adversarial nets. *Mathematical Problems in Engineering*, *2018*.

Appendices

A Drawbacks of Traditional GANs

In this Appendix, I elaborate on the challenges that arise in the training process of the original GAN framework. The challenges include vanishing gradients in the loss function and unstable training of the generator, both causing the generative samples to be of bad quality.

From Equation 2 in Chapter 4, you can see that the GAN loss function reads (Goodfellow et al., 2014):

$$E_{x\sim P_r}[log(D(x)] + E_{x\sim P_g}[log(1 - D(x))], \tag{17}$$

where D represents the discriminator, P_r the real distribution, and P_g the distribution of the generator. Intuitively, it would make sense to first find an optimal discriminator before optimizing the generator, as you would expect the discriminator to give reliable feedback to the generator once it is trained optimally. However, as will be explained below, generator updates gets worse as the discriminator gets better. Under the optimal discriminator, the problem with $log(1 - D(x))$ is that it saturates early in the training process, when the generated data is still very poor and, therefore, the discriminator rejects these samples with high confidence. You will then see $D(x) \to 0$ for $x \sim P_g$, yielding $log(1 - D(x)) \to log(1) = 0$. To solve this, Goodfellow et al. (2014) do the $-log(D(x))$ trick, i.e. they replace $E_{x\sim P_g}[log(1 - D(x))]$ with $E_{x\sim P_g}[-log(D(x))]$. However, this form causes unstable training as it involves the asymmetric KL divergence. As both forms raise problems during the training process, WGAN has been proposed as a suitable alternative for the GAN by Arjovsky et al. (2017).

Firstly I will elaborate on the problem that arises with the first notation of the loss function. Secondly, I will elaborate on the problem with the second notation. I will comment on the problems that arise under the optimal discriminator. To get an expression for the optimal discriminator, I investigate from Equation 17 the contribution to the overall loss of an arbitrary sample x, i.e. $P_r(x)\log(D(x)) + P_g(x)\log(1 - D(x))$. By first taking the derivative with respect to $D(x)$ and then setting the derivative equal to 0, I find the following expression for the optimal discriminator:

$$D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)}. \tag{18}$$

From this Equation, you see that the optimal discriminator only looks at the relative ratio between the likelihood of the sample coming from the generative distribution and

the real distribution. At the point that the generative distribution fully follows the real distribution, $P_r(x) = P_g(x)$, the optimal discriminator expresses its indecisiveness by giving an output value of 0.5 as the chance is half-half that x comes from either the real or generative distribution.

The problem with $E_{x\sim P_g}[log(1 - D(x))]$

Implementing Equation 18 in Equation 17, we get:

$$E_{x\sim P_r} log(\frac{P_r(x)}{\frac{1}{2}[P_r(x) + P_g(x)]}) + E_{x\sim P_g} log(\frac{P_g(x)}{\frac{1}{2}[P_r(x) + P_g(x)]}) - 2log(2). \tag{19}$$

Next, I write this equation in terms of the Kullback-Leibler (KL) and Jensen-Shannon (JS) divergence (Arjovsky & Bottou, 2017). The KL divergence reads: $KL(P_1||P_2) = E_{x\sim P_1} log(\frac{P_1}{P_2})$ and the JS divergence reads: $JS(P_1||P_2) = \frac{1}{2}KL(P_1||\frac{P_1+P_2}{2}) + \frac{1}{2}KL(P_2||\frac{P_1+P_2}{2})$. Therefore, I get the following expression for the loss function of under the optimal discriminator (Arjovsky & Bottou, 2017):

$$E_{x\sim P_r}[log(D^*(x)] + E_{x\sim P_g}[log(1 - D^*(x))] = 2JS(P_r||P_g) - 2log(2). \tag{20}$$

From this Equation, you see that minimizing Equation 17 boils down to minimizing the JS divergence when the discriminator is optimal. This implicates a minimization of the distance between P_g and P_r and therefore, yields a network generating fake samples that approach the real samples as accurately as possible.

However, the JS divergence vanishes when P_r and P_g have disjoint supports or when their supports lie in low-dimensional manifolds (Arjovsky & Bottou, 2017). This leads to a constant loss function under the optimal discriminator (Equation 20), i.e. -$2log(2)$. The gradient then equals 0, which means that the generator is unable to learn from the optimal discriminator. In other words, we experience diminishing gradients during in the generator during GAN training as the discriminator gets better, eventually leading to a gradient equal to 0 under the optimal discriminator. This makes it extremely hard to train in the GAN using the loss function described in Equation 17.

Having stated that the JS divergence vanishes if the supports of P_r and P_g either are disjoint or lie in low-dimensional manifolds, I will now explain why this is the case. For disjoint supports, the samples from P_r and P_g do not share a neighborhood, causing the KL divergence, and thereby the JS divergence, to vanish. This is also the case when the

supports of P_r and P_g lie in non-matching low-dimensional manifolds. If the two manifolds are perfectly matched, this would not be the case, however, the chance that this occurs is 0 as for extremely small perturbations this property is already violated. Arjovsky and Bottou (2017) prove that two manifolds intersect transversally or don't intersect at all in the presence of an arbitrarily small random perturbations. This means that the two manifolds do not perfectly match and implicates that their intersection is a finite union of manifolds with dimensions lower than the dimension of the manifolds of P_r and P_g, meaning that their intersection has null-measure in both manifolds. This again implicates the KL divergence measure to vanish, which means that the JS divergence measure also vanishes.

The problem with $E_{x\sim P_g}[-log(D(x))]$

To illustrate the problem with the loss function in terms of $E_{x\sim P_g}[-\log(D(x))]$, I rewrite $E_{x\sim P_g}[-\log(D(x))]$ in terms of the KL and JS divergence. First, I implement Equation 18 in the KL divergence, to get the KL divergence in terms of the optimal discriminator:

$$KL(P_g||P_r) = E_{x\sim P_g}\log(1 - D^*(x)) - E_{x\sim P_g}\log(D^*(x)). \tag{21}$$

Next, I combine Equation 20 with Equation 21, to get:

$$E_{x\sim P_g}[-log(D^*(x))] = KL(P_g||P_r) - E_{x\sim P_g}[log(1 - D^*(x))]$$
$$E_{x\sim P_g}[-log(D^*(x))] = KL(P_g||P_r) - 2JS(P_r||P_g) + 2log(2) + E_{x\sim P_g}log(D^*(x)), \tag{22}$$

where the last two terms in Equation 22 are independent of the generative distribution and therefore can be left out of consideration when optimizing the discriminator loss. The first two terms contradict each other, i.e. the first part wants to pull P_r toward P_g whereas the second part wants to push P_g away from P_r two times harder, i.e. the minus in front of the JS enforces P_g and P_r to be different. Moreover, if we solely consider the KL divergence, we also encounter problems due to the asymmetric character of this divergence. The extreme case for $P_r(x) > P_g(x)$, meaning that the sample x has a higher probability of coming from the data than from the generator, is $P_r(x) > 0$ and $P_g(x) \to 0$. This yields a KL divergence measure that is going to infinitiy, which means a severe punishment on the network if the generator does not cover parts of the real data distribution. The extreme case for $P_r(x) < P_g(x)$, i.e. the generator outputs a sample

that does not resemble the data well, is $P_r(x) \to 0$ and $P_g(x) > 0$. This yields a KL divergence measure that is going to 0, which means an extremely low punishment on the network if the generator generates fake looking samples. This means that the network is severely punished if it generates a broad spectrum of samples, including some real-looking samples, thereby failing to describe the whole real data distribution, as this results in a KL divergences approaching infinity. Therefore, the network favors generating rather safe fake looking samples p that approximate the real sample to a certain extent, instead of trying to improve on these samples by exploring a large set of possibilities. This effect is described as mode collapse and often occurs in GANs.

B Adam Optimizer

In this Appendix, I elaborate on the optimization technique that is used in this research: namely the Adam optimizer.

Adam optimization is an adaptive moment estimation optimization technique that computes adaptive learning rates for each parameter and includes momentum by keeping an exponentially decaying average of past gradients. The weights and biases of the network that minimize the loss are the optimal parameters. Stochastic Gradient Descent (SGD) is an optimization method used for the optimization of the loss function in neural network architectures. The SGD algorithm calculates the gradient of the loss function for a parameter value and then takes a small step into the direction of the negative gradient to get an updated value for the parameter. The partial derivatives of the loss functions with respect to the weights and biases, i.e. $\frac{\partial \vartheta}{\partial w_{ij}^l}$ and $\frac{\partial \vartheta}{\partial b_j^l}$ serve as input for the optimizer. As calculating the gradient itself is computationally intense for a large number of observations, SGD uses a random subsample of observations, i.e. a mini-batch. If the subsample is taken large enough, the average gradient over the mini-batch approximates the average gradient of the whole sample.

The gradient is then used to update the parameter value (Kingma & Ba, 2014):

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta \vartheta(\theta), \tag{23}$$

where θ^{new} represents the updated parameter value, θ^{old} the old parameter value, α the learning rate, and $\nabla_\theta \vartheta(\theta)$ the gradient.

The partial derivatives with respect to the weights and bias, i.e. $\frac{\partial \vartheta}{\partial w_{ij}^l}$ and $\frac{\partial \vartheta}{\partial b_j^l}$, serve as input parameters for the optimizer. In vector notation for layer l, they and can be expressed as follows:

$$\frac{\partial \vartheta}{\partial \boldsymbol{b}^l} = \frac{\partial \vartheta}{\partial \boldsymbol{a}^l} \frac{\partial \boldsymbol{a}^l}{\partial \boldsymbol{\gamma}^l} \frac{\partial \boldsymbol{\gamma}^l}{\partial \boldsymbol{b}^l} \qquad\qquad \frac{\partial \vartheta}{\partial \boldsymbol{W}^l} = \frac{\partial \vartheta}{\partial \boldsymbol{a}^l} \frac{\partial \boldsymbol{a}^l}{\partial \boldsymbol{\gamma}^l} \frac{\partial \boldsymbol{\gamma}^l}{\partial \boldsymbol{W}^l}. \tag{24}$$

In Equation 24, you can see the term $\frac{\partial \vartheta}{\partial \boldsymbol{a}^l} \frac{\partial \boldsymbol{a}^l}{\partial \boldsymbol{\gamma}^l}$ in both partial derivative expressions. I define this term as the shared error s_j^l:

$$s_j^l = \frac{\partial \vartheta}{\partial a_j^l} \frac{\partial a_j^l}{\partial \gamma_j^l} = \frac{\partial \vartheta}{\partial a_j^l} f'(\gamma_j^l). \tag{25}$$

Or in vector notation:

$$\boldsymbol{s}^l = \nabla_a \vartheta \odot f'(\boldsymbol{\gamma}^l), \tag{26}$$

where $\nabla_a \vartheta$ is a vector with n partial derivatives $\frac{\partial \vartheta}{\partial \boldsymbol{a}_j^l}$ for all the n neurons in the layer, $f'(\gamma^l)$ the partial derivatives $f'(\gamma_j^l)$ for $j \in 1, ..., n$, and $\odot$ the Hadamard product. Next, I define the shared error in layer l in terms of the shared error in layer $l+1$:

$$\boldsymbol{s}^l = ((\boldsymbol{w}^{l+1})^T \boldsymbol{s}^{l+1}) \odot f'(\boldsymbol{\gamma}^l). \tag{27}$$

From Equation 27, you can see that the weight matrix facilitates the transportation of the shared error backward through the network, i.e. from layer $l+1$ to layer l. By taking the Hadamard product with $f'(\boldsymbol{\gamma}^l)$, you can also include the effect on the activation function in layer l. Combining Equation 26 and 27, I obtain the following expressions for $\frac{\partial \vartheta}{\partial w_{ij}^l}$ and $\frac{\partial \vartheta}{\partial b_j^l}$:

$$\frac{\partial \vartheta}{\partial w_{ij}^l} = a_k^{l-1} s_j^l \qquad\qquad \frac{\partial \vartheta}{\partial b_j^l} = s_j^l. \tag{28}$$

The Adam optimizer is an adaptive optimization method, which means that it updates the learning rate for each parameter. On the contrary, SGD deploys a constant learning rate. To get the expression for the adaptive learning rate, I firstly estimate the first and second order moment by means of exponentially moving averages:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \end{aligned} \tag{29}$$

where m and v are moving averages, g the gradient, and β_1 and β_2 hyperparameters of the Adam optimizer. In this research, I set β_1 equal to 0.9 and β_2 to 0.999, these are default values and rarely adjusted.

Equation 29 now describes the relation between $t-1$ and t. If we expand this relation ranging from $t=0$ to $t=t$ we get: $m_t = (1-\beta_1) \sum_{i=0}^{t} \beta_1^{t-i} g_i$ and $v_t = (1-\beta_2) \sum_{i=0}^{t} \beta_2^{t-i} g_i^2$. Taking the expectation, including a correction ζ as we approximate g_i by g_t and using the formula for the sum of a finite geometric series, yields:

$$\begin{aligned} E[m_t] &= E[g_t](1-\beta_1^t) + \zeta_1 \\ E[m_t] &= E[g_t](1-\beta_1^t) + \zeta_2. \end{aligned} \tag{30}$$

Equation 29 describes the estimates of the first and second moments of the gradient, m and v respectively. We want these moments to approximate the true first and second moment of the gradient, i.e.:

$$E[m_t] = E[g_t] \qquad\qquad E[v_t] = E[g_t^2]. \tag{31}$$

By combining Equation 30 and Equation 31 I get the following biased corrected estimators for the first and second moments:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \qquad\qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t}. \tag{32}$$

Equation 32 is then used to scale the learning rate individually for each parameter by the weight parameter w_t:

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{33}$$

where η is the step size, and ϵ an error term.

C Architecture

In this Appendix, I provide details on the WGAN networks, i.e. the generator and the discriminator.

The Generator

The generator consists of a conditioning and simulation part, as depicted in Figure 1a in Section 4. In Table 7, you can find an outline of the layers in the generator. The layers use the ReLU activation function. Layers 1 to 7 form the conditioning part. The simulation part consists of layers 10 to 13. The convolutional layers have stride 2 and kernel size 5, and cover 7 channels as both portfolios consist of 7 assets. Moreover, the padding methodology in the first 2 convolutional layers is set to *valid*. Valid padding adjusts the input size of the convolutional layer if the original size is not compatible with the kernel size and stride settings by discarding input values. In convolutional layers No. 3 and No. 4, the padding methodology is set to *same*. Same padding adds zeros in a symmetric manner to the beginning and end of the input values to match the input size with the kernel and stride settings. Followed by a flatten layer which translates the three-dimensional output of the last convolutional layer into its two-dimensional equivalent, the conditioning part is concluded by a dense layer.

Next, I have 2 input layers to incorporate the excess values and the random values. The processed historical closing prices and the excess values serve as conditioning information for WGAN. Conditioning the random values on this information, it is processed by one Dense layer and two transpose convolutional layers, which compress the number of channels, thereby allowing the channels (i.e. the closing prices of the assets) to interact.

The Discriminator

The discriminator consists of five consecutive convolutional layers, again with stride 2 and kernel size 5. For the first 3 layers I apply valid padding and for the last 2 layers I apply same padding. The convolutional layers are followed by a Flatten layer which downsizes the output of the last convolutional layer to concatenate it with the excess values. After concatenation the discriminator produces a single output, the critic value, using the dense layer with linear activation. Except for layer No. 9 (the final dense

layer), the layers deploy the leaky ReLU activation function. Through the convolutions, the number of channels is increased whereas the temporal direction gets downsized. The latter is necessary as the discriminator outputs a single value for each batch rather than a time series.

Table 7. *The generator network*

Layer no.	Layer type	Output shape	Comments
1	Input layer	(batches, 40, 7)	Normalized closing prices
2	1D Conv	(batches, 18, 7)	
3	1D Conv	(batches, 7, 7)	
4	1D Conv	(batches, 2, 7)	
5	1D Conv	(batches, 1, 7)	
6	Flatten	(batches, 7)	
7	Dense	(batches, 7)	
8	Input layer	(batches, 7)	Asset excess values
9	Input layer	(batches, 14)	Random values: $\boldsymbol{z} \sim U(0,1)$
10	Dense layer	(batches, 20)	
11	Reshape	(batches, 20, 28)	
12	1D Conv Transpose	(batches, 20, 14)	
13	1D Conv Transpose	(batches, 20, 7)	

Table 8. *The discriminator network*

Layer no.	Layer type	Output shape	Comments
1	Input layer	(batches, 60, 7)	Real or fake data
2	1D Conv	(batches, 28, 7)	
3	1D Conv	(batches, 12, 14)	
4	1D Conv	(batches, 4, 28)	
5	1D Conv	(batches, 2, 56)	
6	1D Conv	(batches, 1, 112)	
7	Flatten	(batches, 112)	
8	Input layer	(batches, 7)	Asset excess values
9	Dense layer	(batches, 1)	

D Forecasting Performance

This Appendix shows the forecasting performance of WGAN for four different horizons. From Table 9, you can see that there is no horizon outperforming across all constituents. The 10-day horizon is outperforming for ARM, AMS, KIO, ASR, and ALSI. However, for ANG and GFI the 20-day horizon performs best. For the constituents of the Russian portfolio, the forecasting performance of WGAN is not consistently superior for one horizon either (Table 10). For GAZP, ROSN, SNGS and MCFTR, the 10-day horizon is superior. For LKOH and NVTK the 5-day horizon performs best, and for WGAN shows the best performance for the 20-day horizon for TATN.

	ARM		AMS		ANG		GFI		KIO		ASR		ALSI	
1 day ahead														
% correct	72.2		72.9		69.9		67.2		75.7		72.4		66.5	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	15,861	16,041	18,751	11,469	26,266	15,204	36,452	16,563	32,021	14,049	33,681	16,920	27,783	18,087
down	12,389	61,459	14,999	59,281	15,484	47,546	16,798	35,437	10,729	48,951	10,569	43,830	14,717	43,413
5 days ahead														
% correct	75.9		83.3		78.0		75.6		67.9		72.4		74.3	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	4,953	2,076	4,494	1,132	4,622	1,817	7,065	1,509	5,807	3,279	6,949	2,541	7,341	3,289
down	2,547	10,424	2,256	12,368	2,628	11,183	3,185	8,241	1,193	9,971	3,051	7,709	1,659	7,711
10 days ahead														
% correct	77.7		83.7		79.0		70.9		79.8		79.6		81.3	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	1,970	1,501	2,015	685	2,471	377	3,283	513	3,284	1,107	3,127	1,468	4,311	1,224
down	780	5,999	985	6,565	1,779	5,623	2,467	3,987	716	4,893	373	5,032	689	4,026
20 days ahead														
% correct	66.3		61.7		81.2		77.4		71.3		74.6		44.8	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	1,316	835	1,250	508	1,588	323	1,465	403	1,918	675	1,783	614	1,500	2,150
down	684	2,165	1,250	1,992	662	2,677	535	2,597	832	1,825	717	2,136	750	850

Table 9. *Characteristics and performance of the WGAN per constituent of the South Africa portfolio for four different time horizons.*

	GAZP		LKOH		NVTK		ROSN		SNGS		TATN		MCFTR	
1 day ahead														
% correct	68.1		76.1		72.0		75.4		63.4		77.7		65.2	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	22,260	16,264	6,279	15,128	12,202	17,716	34,006	11,690	42,140	15,684	19,429	9,854	7,488	20,098
down	16,240	50,736	8,221	75,372	11,298	65,034	11,244	46,810	21,860	25,816	12,821	63,896	14,012	62,402
5 days ahead														
% correct	78.5		76.5		73.9		75.4		69.5		75.0		66.8	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	5,468	2,581	1,269	2,521	3,276	3,569	7,121	3,350	8,377	4,059	3,455	3,262	2,632	4,601
down	1,782	10,419	1,981	14,229	1,474	11,681	1,629	8,150	2,123	5,691	1,795	11,738	1,868	10,899
10 days ahead														
% correct	79.3		73.8		67.7		79.2		82.9		80.2		72.5	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	2,808	1,181	1,129	561	1,912	1,721	3,844	1,480	4,717	1,220	2,211	1,243	2,370	1,438
down	942	5,319	1,871	6,439	1,588	5,029	656	4,270	533	3,780	789	6,007	1,380	5,062
20 days ahead														
% correct	68.1		52.6		68.5		66.0		60.9		83.5		61.2	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	1,349	275	278	19	845	249	1,444	479	2,004	1,059	1,169	36	802	87
down	901	2,225	1772	2481	1,405	2,751	1,306	2,021	996	1,191	831	3,214	1,698	2,413

Table 10. *Characteristics and performance of the WGAN per constituent of the Russian portfolio for four different time horizons.*

E Robustness

This Appendix shows the forecasting performance of WGAN in case the closing prices are slightly modified with respect to the original closing prices. By means of forecasting based on the adjusted closing prices and evaluating this performance, I assess the robustness of WGAN. If the performance across slight modifications match, I can conclude that WGAN is robust.

From Table 11, Table 12, and Table 9 (Appendix D), you can see that for the South African market, the WGAN performance is quite stable with percentages varying up to 5.1%. For example, for AMS, the 20-day results is 66.8% for the +10% adjusted closing prices, 61.9% for the −10% adjusted closing prices, and 61.7% for the original closing prices. For KIO, the 10-day results is 82.8% for the +10% adjusted closing prices, 84.2% for the −10% adjusted closing prices, and 79.6% for the original closing prices. For the other constituents, the results are even less variable.

From Table 13, Table 14, and Table 10 (Appendix D), you can see that for the Russian market WGAN also performs steadily across varying closing prices. The largest performance difference you can see for ROSN 20-day, with 65.7% for adjusted closing prices +10%, 66.3 for adjusted closing prices −10%, and 66.0% for the unadjusted closing prices.

Overall, I conclude that WGAN performs steadily for various closing price inputs.

	ARM		AMS		ANG		GFI		KIO		ASR		ALSI	
1 day ahead														
% correct	72.3		73.2		69.3		66.9		75.8		72.3		66.5	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	15,894	16,002	18,877	11,512	26,347	15,155	36,428	16,574	31,847	13,982	33,737	16,878	27,708	18,056
down	12,356	61,498	15,123	59,488	15,153	47,845	17,322	35,176	10,903	49,268	11,013	43,622	14,792	43,444
5 days ahead														
% correct	76.0		83.5		79.8		75.5		77.9		72.7		74.0	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	4,972	2,083	4,504	1,095	4,913	1,503	7,038	1,498	5,784	3,253	6,974	2,509	7,309	3,320
down	2,528	10,417	2,246	12,405	2,587	11,247	3,212	8,252	1,216	9,997	3,026	7,741	1,691	7,680
10 days ahead														
% correct	78.1		84.1		79.4		71.3		82.8		79.3		81.0	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	2,004	1,502	2,052	679	2,518	379	3,300	494	3,287	1,054	3,149	1,520	4,300	1,251
down	746	5,998	948	6,571	1,732	5,621	2,450	4,006	713	5,196	601	4,980	700	3,999
20 days ahead														
% correct	66.3		66.8		81.6		82.2		71.5		74.2		44.8	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	1,308	827	1,250	492	1,586	304	1,459	392	1,911	657	1,773	626	1,499	2,147
down	692	2,173	1,250	2,258	664	2,696	541	2,858	839	1,843	727	2,124	751	853

Table 11. *Characteristics and performance of the WGAN per constituent of the South Africa portfolio for four different time horizons, for closing prices adjusted* +10%.

	ARM		AMS		ANG		GFI		KIO		ASR		ALSI	
1 day ahead														
% correct	72.2		73.3		69.2		67.0		75.9		72.2		66.9	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	15,899	16,081	18,883	11,492	26,324	15,251	36,472	16,494	31,997	14,046	33,723	16,922	27,866	18,012
down	12,351	61,419	15,117	59,508	15,176	47,749	17,278	35,256	10,753	49,204	11,027	43,578	14,634	43,488
5 days ahead														
% correct	76.1		83.5		79.7		75.6		77.9		72.9		74.1	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	4,976	2,069	4,502	1,094	4,891	1,496	7,061	1,495	5,791	3,269	7,009	2,502	7,334	3,324
down	2,524	10,431	2,248	12,406	2,609	11,254	3,189	8,255	1,209	9,981	2,991	7,748	1,666	7,676
10 days ahead														
% correct	78.1		84.2		79.5		71.3		84.2		79.3		81.2	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	1,985	1,476	2,058	680	2,514	366	3,309	500	3,282	1,083	3,141	1,515	4,310	1,235
down	765	6,024	942	6,570	1,736	5,634	2,441	4,000	718	5,167	609	4,985	609	4,015
20 days ahead														
% correct	66.6		61.9		81.7		82.0		71.2		74.7		45.1	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	1,312	814	1,251	505	1,598	311	1,468	412	1,904	664	1,806	635	1,500	2,132
down	688	2,186	1,249	2,245	652	2,689	532	2,838	846	1,836	694	2,115	750	868

Table 12. *Characteristics and performance of the WGAN per constituent of the South Africa portfolio for four different time horizons, for closing prices adjusted* -10%.

	GAZP		LKOH		NVTK		ROSN		SNGS		TATN		MCFTR	
1 day ahead														
% correct	68.3		76.1		71.8		75.6		63.4		77.5		65.4	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	22,286	16,333	6,264	15,155	12,211	17,729	34,044	11,936	42,118	15,898	19,175	9,846	7,724	20,093
down	16,214	50,917	8,236	75,345	11,039	64,771	11,206	47,064	21,882	25,852	12,825	63,904	14,026	62,407
5 days ahead														
% correct	78.4		76.1		75.4		75.3		69.3		75.4		67.2	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	5,464	2,585	1,237	2,580	3,310	3,539	7,087	3,336	8,360	4,084	3,471	3,203	2,658	4,543
down	1,786	10,415	2,013	14,170	1,440	11,961	1,663	8,164	2,140	5,666	1,779	11,797	1,842	10,957
10 days ahead														
% correct	79.5		73.8		68.1		78.5		82.5		79.3		72.4	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	2,839	1,190	1,146	586	1,995	1,769	3,839	1,543	4,713	1,260	2,213	1,332	2,395	1,469
down	911	5,310	1,854	6,414	1,505	4,981	661	4,207	537	3,740	787	5,918	1,355	5,031
20 days ahead														
% correct	72.4		52.6		68.6		65.7		65.6		83.6		61.0	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	1,332	282	278	14	870	268	1,429	479	2,002	1,058	1,178	37	805	102
down	918	2,468	1,722	2,486	1,380	2,732	1,321	2,021	748	1,442	822	3,213	1,695	2,398

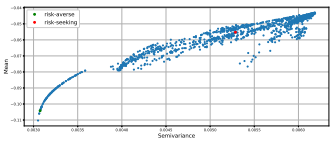
Table 13. *Characteristics and performance of the WGAN per constituent of the Russian portfolio for four different time horizons, for closing prices adjusted* +10%.

	GAZP		LKOH		NVTK		ROSN		SNGS		TATN		MCFTR	
1 day ahead														
% correct	68.3		76.1		71.7		75.7		63.4		77.5		65.5	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	22,246	16,279	6,254	15,156	12,207	17,847	34,003	11,848	42,108	15,852	19,196	9,873	7,773	20,066
down	16,254	50,971	8,246	75,344	11,043	64,653	11,247	47,152	21,892	25,898	12,804	63,877	13,977	62,434
5 days ahead														
% correct	78.4		76.4		75.0		75.4		69.3		75.5		67.3	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	5,463	2,587	1,268	2,543	3,306	3,619	7,098	3,324	8,346	4,067	3,465	3,185	2,642	4,515
down	1,787	10,413	1,982	14,207	1,444	11,881	1,652	8,176	2,154	5,683	1,785	11,815	1,858	10,985
10 days ahead														
% correct	79.7		74.3		68.7		78.5		82.5		80.3		73.5	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	2,810	1,136	1,136	520	1,953	1,662	3,816	1,517	4,704	1,244	2,216	1,233	2,411	1,380
down	940	5,364	1,864	6,480	1,547	5,088	684	4,233	546	3,756	784	6,017	1,339	5,120
20 days ahead														
% correct	72.5		52.5		68.9		66.3		65.8		83.6		61.6	
predicted/realized	up	down	up	down	up	down	up	down	up	down	up	down	up	down
up	1,335	278	278	20	871	256	1,450	470	2,005	1,049	1,172	35	823	89
down	915	2,472	1,722	2,480	1,379	2,744	1,300	2,030	745	1,451	828	3,215	1,677	2,411

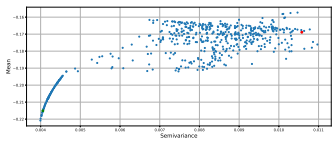
Table 14. *Characteristics and performance of the WGAN per constituent of the Russian portfolio for four different time horizons, for closing prices adjusted* -10%.

F Frontiers from NSGA-II and MC

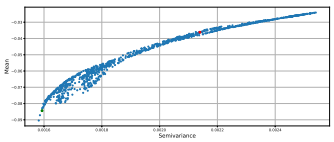
This Appendix shows the Pareto fronts for time intervals of 20-days as the portfolios are rebalanced every 20 days according to the investor's risk-return preferences. This means that we require a new allocation every 20 days, yielding 21 fronts for the investing period that covers 420 days, i.e. from 2017-09 to 2019-04. In Figure 9 and Figure 10, you can find the final frontiers for all 20 time periods.



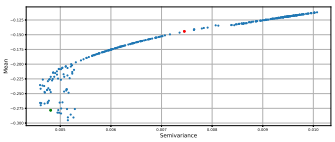
(a) Batch 1: 2017-09-18 to 2017-10-16



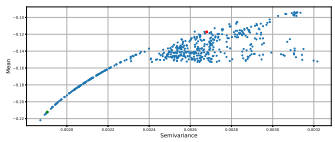
(b) Batch 2: 2017-10-16 to 2017-11-13



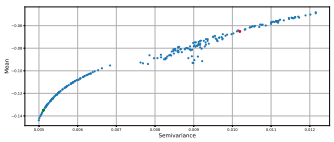
(c) Batch 3: 2017-11-13 to 2017-12-11



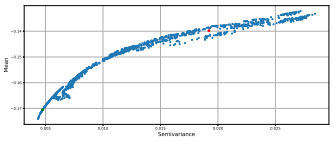
(d) Batch 4: 2017-12-11 to 2018-01-08



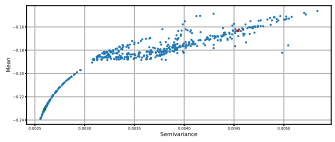
(e) Batch 5: 2018-01-08 to 2018-02-05



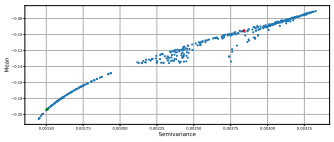
(f) Batch 6: 2018-02-05 to 2018-03-05



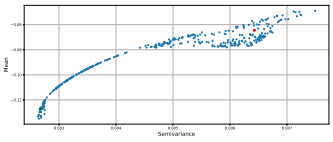
(g) Batch 7: 2018-03-05 to 2018-04-02



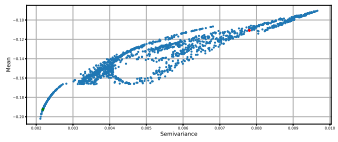
(h) Batch 8: 2018-04-02 to 2018-05-28



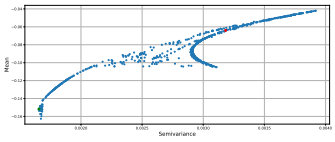
(i) Batch 9: 2018-05-28 to 2018-06-25



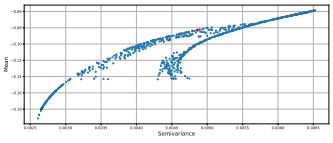
(j) Batch 10: 2018-06-25 to 2018-07-23



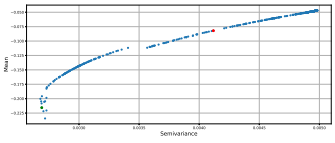
(k) Batch 11: 2018-07-23 to 2018-08-20



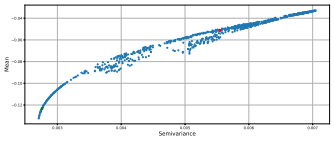
(l) Batch 12: 2018-08-20 to 2018-09-17



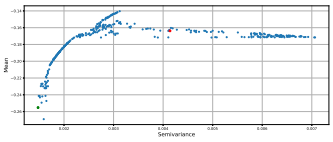
(m) Batch 13: 2018-09-17 to 2018-10-15



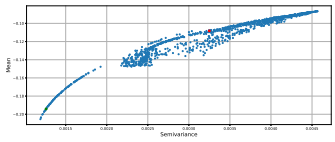
(n) Batch 14: 2018-10-15 to 2018-11-12



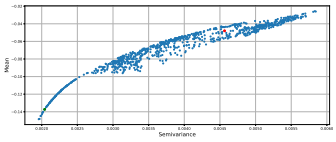
(o) Batch 15: 2018-11-12 to 2018-12-10



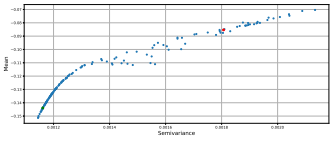
(p) Batch 16: 2018-12-10 to 2019-01-04



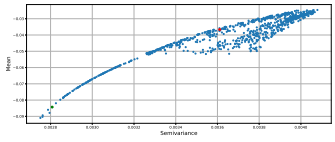
(q) Batch 17: 2019-01-04 to 2019-02-01



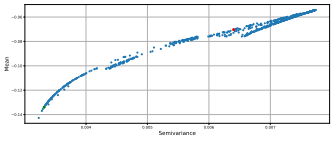
(r) Batch 18: 2019-02-01 to 2019-03-01



(s) Batch 19: 2019-03-01 to 2019-03-29

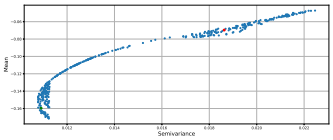


(t) Batch 20: 2019-03-29 to 2019-04-26

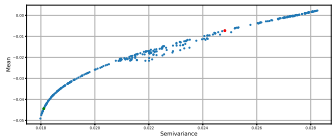


(u) Batch 21: 2019-04-26 - 2019-05-20

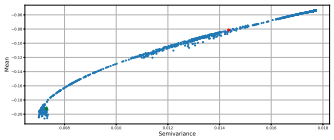
Figure 9. Mean-semivariance frontiers: relevant portfolios extracted from the NSGA-II simulations as well as the MC simulations. The red portfolio represents the risk-seeking investor. The green portfolio represents the risk-averse investor.



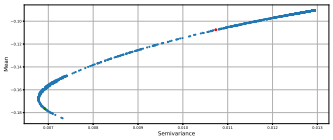
(a) Batch 1: 2017-09-18 to 2017-10-16



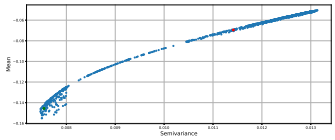
(b) Batch 2: 2017-10-16 to 2017-11-13



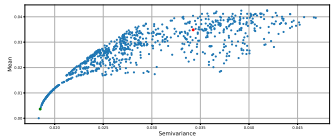
(c) Batch 3: 2017-11-13 to 2017-12-11



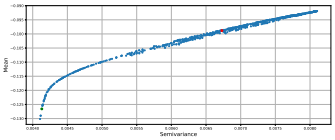
(d) Batch 4: 2017-12-11 to 2018-01-08



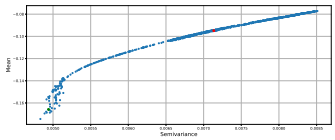
(e) Batch 5: 2018-01-08 to 2018-02-05



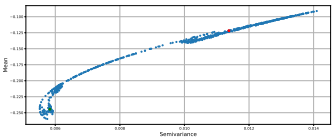
(f) Batch 6: 2018-02-05 to 2018-03-05



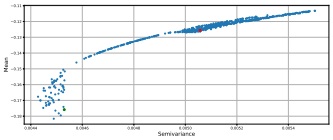
(g) Batch 7: 2018-03-05 to 2018-04-02



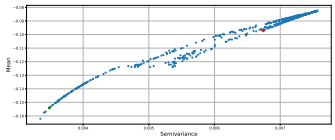
(h) Batch 8: 2018-04-02 to 2018-05-28



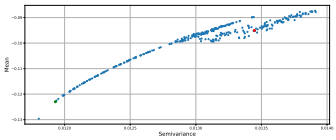
(i) Batch 9: 2018-05-28 to 2018-06-25



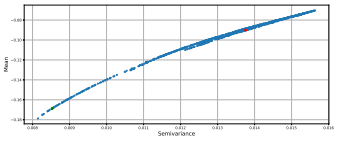
(j) Batch 10: 2018-06-25 to 2018-07-23



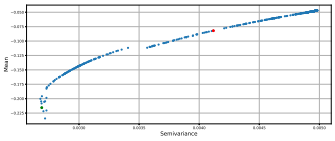
(k) Batch 11: 2018-07-23 to 2018-08-20



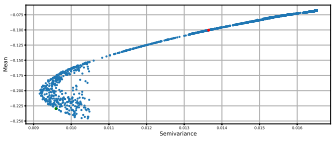
(l) Batch 12: 2018-08-20 to 2018-09-17



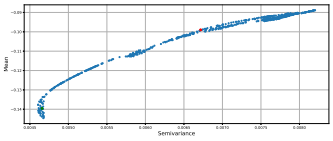
(m) Batch 13: 2018-09-17 to 2018-10-15



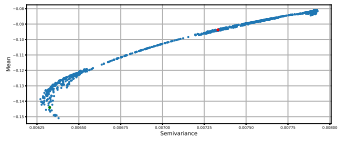
(n) Batch 14: 2018-10-15 to 2018-11-12



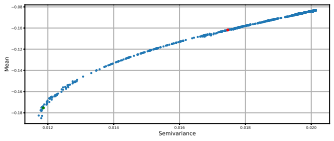
(o) Batch 15: 2018-11-12 to 2018-12-10



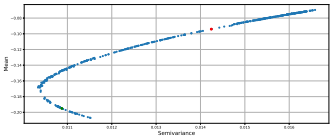
(p) Batch 16: 2018-12-10 to 2019-01-04



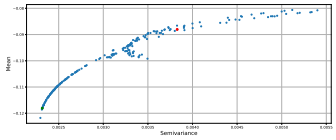
(q) Batch 17: 2019-01-04 to 2019-02-01



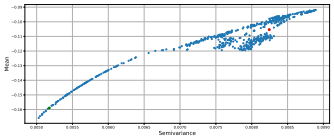
(r) Batch 18: 2019-02-01 to 2019-03-01



(s) Batch 19: 2019-03-01 to 2019-03-29



(t) Batch 20: 2019-03-29 to 2019-04-26



(u) Batch 21: 2019-04-26 - 2019-05-20

Figure 10. Mean-semivariance frontiers for the Russian stocks: relevant portfolios extracted from the NSGA-II simulations as well as the MC simulations. The red portfolio represents the risk-seeking investor. The green portfolio represents the risk-averse investor.