Optimizing VaR with Generative Adversarial Networks

UNIVERSITY OF WESTMINSTER

MSc Fintech and Business Analytics

Project (Fintech with Business Analytics)
Module Code: 7FNCE045W
StudentID: w20014902

Acknowledgement

I would like to express my heartfelt appreciation to everyone who has supported me throughout this research journey. First and foremost, I extend my deepest thanks to my supervisor, **Dr. Issam Malki**, whose invaluable guidance, thoughtful feedback, and continuous encouragement have been vital to the successful completion of this dissertation. His expertise and dedication to excellence have greatly influenced my work.

I am also thankful to the University of Westminster for providing the essential resources and facilities that made this research possible.

Finally, I am profoundly grateful to my family for their unwavering love, patience, and support. Their constant encouragement has been a cornerstone of this achievement.

Declaration

I, **Pulkit Gupta** [w20014902], hereby declare that this dissertation titled "**Optimizing VaR with Generative Adversarial Networks**" is my original work and has been written entirely by me. I affirm that the content of this dissertation has not been submitted, either in full or in part, towards any other degree or professional qualification.

All sources of information, data, or materials utilized in the preparation of this dissertation have been properly acknowledged. Any quotations from other sources are clearly identified and attributed to their respective authors. Any mistakes or omissions are solely my responsibility.

I have strictly followed the ethical guidelines for research as outlined by the University of Westminster. Additionally, I confirm that this work adheres to the institution's standards for research integrity and academic honesty.

Abstract

Effective risk management is crucial for financial institutions, both to protect their interests and to comply with regulatory requirements. A key aspect of this process is accurately assessing the risks associated with financial positions, with Value-at-Risk (VaR) being a particularly important measure. VaR estimates the potential loss in value of an asset or portfolio over a specific period, under normal market conditions, at a given confidence level. To enhance the accuracy of VaR estimation, a variety of methods have been developed over time.

This study explores the use of Generative Adversarial Networks (GANs), a type of deep learning model, for VaR estimation. GANs, which consist of a generator and a discriminator network, are known for generating data that closely mimics real data. While GANs have been widely used in areas like image and text generation, their application in financial risk management, especially for VaR estimation, is still relatively new.

The research focuses on both unconditional and conditional VaR forecasts using GANs. Unconditional VaR relies solely on historical data, while conditional VaR incorporates current market conditions. The findings reveal that GANs perform particularly well in the unconditional setting, often surpassing traditional statistical methods. This suggests that GANs can better capture the underlying data distribution, leading to more accurate risk assessments.

However, conditional VaR forecasting proves more challenging. Although GANs still provide competitive results, their performance in this context is not as consistently superior. Despite these challenges, the study demonstrates that GANs hold significant potential as a tool for future risk management.

In conclusion, GANs offer a promising new approach to VaR estimation, showing potential to outperform classical methods and paving the way for more sophisticated risk management strategies in financial institutions.

Table of Contents

Chapter 1

Introduction

1.1 Work Inspiration

Every day, Dennis Weatherstone, the former Chairman of JPMorgan, would sit in his office awaiting the arrival of the "4-15 report." This one-page document contained a crucial number: the one-day Value-at-Risk (VaR) at the 95% confidence level. This number was vital because it summarized the bank's short-term risk into a single figure, making it an incredibly useful tool for management. Since it was first developed at JPMorgan in the late 1980s, VaR has become a standard tool in nearly every financial institution, and it's hard to imagine modern risk management without it. With the introduction of the Basel II accord in 2004, VaR became a key part of the regulatory framework set by the Basel Committee on Banking Supervision (BCBS), assisting in the calculation of the capital required of banks to cover potentially severe losses.

Accurate Value-at-Risk (VaR) estimation is critical for assessing financial risk, making the choice of the right model for estimating the loss distribution essential. Traditional methods such as maximum likelihood estimation, extreme value theory (EVT), and time series analysis have been widely used (Chan and Nadarajah, 2018). However, the rise of machine learning has introduced new approaches, such as generative models, which offer an alternative to conventional techniques. One notable generative model is the Generative Adversarial Network (GAN) (Goodfellow et al., 2014). Unlike traditional methods that estimate parameters of predefined distributions, GANs are designed to learn the underlying distribution by setting up a competition between two neural networks: the generator and the discriminator. The generator produces synthetic samples, while the discriminator attempts to distinguish between real and fake data. This iterative process aims to produce data that closely mirrors the true distribution.

GANs have achieved success in various fields, such as image generation (Brock et al., 2019), natural language processing (Li et al., 2017), and video creation (Acharya et al., 2018). This raises the question of whether GANs can model the distribution of portfolio returns for improved VaR estimation, making them a potential alternative to traditional methods. However, research on this application is limited, with only preliminary work available (Shah, 2018; Khan, 2019).

This thesis explores GANs for VaR estimation by making the following contributions:

1. We train various GANs to model log-return distributions, improving on prior work and outperforming traditional methods in VaR forecasts through backtesting.

2. We propose a new performance metric for model validation, enabling optimal hyperparameter selection through grid search.

1.2 Associated Work

The use of Generative Adversarial Networks (GANs) for estimating Value-at-Risk (VaR) is still an emerging area of research, with only a few studies directly addressing this application. However, some work has been done using GANs to model financial time series, particularly in predicting stock prices.

For instance, (Zhang et al., 2019) and (Zhou et al., 2018) have developed GAN models to forecast the closing prices of stocks and indices. Their models perform comparably to other advanced machine learning techniques based on root-mean-squared-error (RMSE) metrics. However, these models are deterministic, meaning they do not generate the range of possible outcomes needed to estimate quantiles, which are crucial for VaR calculations.

(Li et al., 2018) applied GANs to simulate the dynamics of limit order books, achieving strong results in modelling the time between trades. However, the GAN's ability to replicate the actual distribution of prices was less successful, making it unsuitable for VaR estimation.

GANs have also been used outside of finance for time series simulation. (Engle, 1982) introduced a Real-Valued Conditional GAN (RCGAN) designed to generate realistic multivariate time series data based on previous observations. This model was applied to simulate medical data, providing realistic scenarios for training healthcare professionals.

In our work, Value at Risk (VaR) estimation is conducted using simulation through Generative Adversarial Networks (GANs) and compared with the traditional variance-covariance approach. The goal is to answer the question:

Is the use of generative adversarial networks suitable for estimating value at risk?

Certain limitations were set for this study. First, VaR estimations were only applied to a stock portfolio, with other asset classes excluded from consideration. Second, the benchmark method used for comparison is the widely used variance-covariance approach. Third, specific architectures of GANs were chosen for this analysis.

1.3 Scope and Structure

This paper is organized as follows: Chapter 2 defines Value-at-Risk (VaR) and examines the most common methods used for its estimation and backtesting. Chapter 3 introduces Generative Adversarial Networks (GANs), starting with a discussion of the universal approximation theorem for probability distributions, then explaining the original

GAN model by (Goodfellow et al., 2014) and the Wasserstein GAN developed by (Arjovsky, Chintala and Bottou, 2017). In Chapter 4, the GAN approach is applied to estimate VaR, and its backtesting performance is compared to traditional methods discussed in Chapter 3.

1.4 Research Question and Hypotheses

This study seeks to explore the potential of Generative Adversarial Networks (GANs) in estimating Value at Risk (VaR), a critical risk management metric widely used in financial institutions. Despite the growing application of GANs in various domains, their use in financial risk estimation, specifically VaR, remains underexplored. The primary research question guiding this study is: **"Can Generative Adversarial Networks (GANs) be effectively used to estimate Value at Risk (VaR) in a manner that is competitive with or superior to traditional methods such as the variance-covariance approach?"**

The study aims to fill a notable gap in the literature by providing a rigorous evaluation of GANs for VaR estimation. While prior research has applied GANs to financial time series prediction and stock price forecasting, few studies have rigorously tested their ability to model the distribution of portfolio returns for VaR estimation. This gap is significant given the increasing importance of accurate risk management tools in the financial industry.

The study hypothesizes that GANs, particularly with their capacity to model complex and non-linear dependencies in data, will produce VaR estimates that are more accurate or at least comparable to traditional methods. It further hypothesizes that improvements in GAN architectures, such as the use of Wasserstein GANs (WGANs) and Gradient Penalty (WGAN-GP), will enhance the stability and accuracy of these models in a financial context. By addressing these hypotheses, the study seeks to contribute new insights into the applicability of GANs in financial risk management, providing a foundation for further research and practical application in the industry.

Chapter 2

Value at Risk

Value-at-Risk (VaR) is a popular tool in finance for measuring the risk of potential losses in a portfolio. Simply put, VaR at a certain confidence level, denoted by α, represents the worst possible loss that is unlikely to be exceeded during a specific period, with a probability of α. However, despite its widespread use, VaR has some drawbacks. One major limitation is that it doesn't provide any insight into how severe the losses could be beyond the VaR threshold. Additionally, VaR is not considered a coherent risk measure, as defined by (Artzner, 1999), because it lacks subadditivity (a property that discourages diversification).

An alternative to VaR that addresses these issues is Expected Shortfall (ES). ES measures the average loss that would occur if the VaR level is exceeded, offering more information about extreme losses. While ES is theoretically more robust, VaR remains the standard in the industry and is a key component of regulatory frameworks.

This chapter will focus on VaR as the primary metric for managing market risk. We will start by formally defining VaR and then discuss some of the most commonly used financial models for estimating and backtesting it. These models will later be used as a benchmark when comparing them with the GAN methodology.

2.1 Understanding VaR

Value at Risk (VaR) is a statistical metric used to quantify the potential loss of an investment or portfolio over a specific time period for a certain degree of confidance. It gives the greatest predicted loss under typical market conditions as a single figure.

Key components of VaR:

- **Confidence level:** The likelihood that the actual loss will not be greater than the value at risk. The usual ranges are 95% and 99%.
- **Time horizon:** The time frame (such as a day, week, or month) used to calculate the possible loss.
- **Loss amount:** The highest possible loss that might be incurred given the given time horizon and confidence level.

Types of VaR

1. **Historical Simulation:**
    - Based on past price data.
    - Believes that returns in the future will be similar to historical performance.
    - Calculate the returns for each historical period.
    - Sort these returns and identify the return corresponding to the specified confidence level.
    - To find the VaR, multiply this return by the value of the portfolio.
2. **Variance-Covariance Method (Philippe Jorion, 2011):**
    - Assumes a normal distribution for asset returns.
    - Uses historical data to estimate the mean and covariance matrix of asset returns.
    - Calculates the portfolio's expected return and standard deviation.
    - Determine the z-score corresponding to the chosen confidence level.
    - Multiply the z-score by the portfolio's standard deviation to obtain the VaR.
3. **Monte Carlo Simulation (Hull, 2015):**
    - Generates a large number of random price scenarios for the underlying assets.
    - Calculates the portfolio value for each scenario.
    - Sort the portfolio values and identify the value corresponding to the specified confidence level.
    - The difference between the initial portfolio value and this value is the VaR.

2.2 VaR and Risk Management

VaR is widely used in risk management for various purposes:

- **Risk measurement:** Quantifies the potential loss of a portfolio.
- **Risk management:** Helps in setting risk limits and making informed investment decisions.
- **Regulatory compliance:** Used by financial institutions to meet regulatory capital requirements.

- **Performance evaluation:** Evaluates the risk-adjusted performance of investment portfolios.

2.2.1 Limitations of VaR

- **Assumptions:** The accuracy of VaR depends on the underlying assumptions, such as normality of returns in the variance-covariance method.
- **Extreme events:** VaR may underestimate the risk of extreme events (tail risk) as it focuses on normal market conditions.
- **Model risk:** The choice of VaR model can significantly impact the results.

2.3 Beyond VaR

While VaR is a valuable tool, it has limitations. Other risk measures, such as Expected Shortfall (ES), are gaining popularity as they provide a more comprehensive view of tail risk as discussed in (Artzner, 1999). ES calculates the expected loss beyond the VaR level.

2.3.1 Unconditional VaR

Unconditional VaR assumes that the probability distribution of asset returns remains constant over time. It is calculated based on historical data or a specific assumed distribution (e.g., normal distribution).

- **Unconditional Absolute VaR:** This measures the potential loss of a portfolio in absolute monetary terms, assuming a constant probability distribution of returns. For example, a $1 million unconditional absolute VaR at a 95% confidence level means there is a 5% chance of losing $1 million or more over the specified time horizon.
- **Unconditional Relative VaR:** This measures the potential loss of a portfolio relative to a benchmark, assuming a constant probability distribution of returns. It is often used to evaluate portfolio performance against a specific index.

2.3.2 Conditional VaR

Conditional VaR acknowledges that the probability distribution of asset returns may change over time. It incorporates information about the current market conditions to estimate the VaR.

- **Conditional Absolute VaR:** This measures the potential loss of a portfolio in absolute monetary terms, considering the current market conditions. It is calculated using models that capture time-varying volatility and other factors affecting returns. For example, GARCH models can be used to estimate conditional volatility.

- **Conditional Relative VaR:** This measures the potential loss of a portfolio relative to a benchmark, considering the current market conditions. It provides a more accurate assessment of relative performance under changing market conditions.

Key Differences

- **Unconditional VaR** assumes a static environment, while **conditional VaR** accounts for dynamic market conditions.
- **Absolute VaR** focuses on the monetary loss, while **relative VaR** compares the portfolio's performance to a benchmark.

2.4 Model Framework and Loss Function

Following the established framework in (Mcneil, Frey and Embrechts, 2015), we consider a portfolio of risky assets with a value denoted by V_t at time t. The absolute loss over a time interval [t, t+Δt] is defined as:

$$L_{\text{abs}}(t;\ \Delta t) = -[V_{t+\Delta t} - V_t] \tag{2.1}$$

The portfolio value, V_t, is typically a deterministic function of time, t, and a vector of risk factors, $Z_t = (Z_t^1, \ldots, Z_t^d)'$. For equity portfolios, these risk factors represent the natural logarithms of stock prices. Given a portfolio of d stocks with prices S_t^i, the risk factors are defined as $Z_t^i = \log(S_t^i)$.

As the portfolio value is the sum of its constituent stock prices, the relationship between portfolio value and risk factors can be expressed as:

$$F(z) = \sum_{i=1}^{n} \exp(z_i) \tag{2.2}$$

Consequently, the absolute loss can be rewritten as:

$$L_{\text{abs}}(t;\ \Delta t) = -[f(Z_{t+\Delta t}) - f(Z_t)] \tag{2.3}$$

For practical applications, the linearized loss, approximated through a first-order Taylor expansion, is often used:

$$L_{\Delta\text{abs}}(t;\ \Delta t) = -\sum_{i=1}^{d} f_{zi}(Z_t) X_t^i \tag{2.4}$$

where $X_t^i = Z_{t+\Delta t}^i - Z_t^i$ represents the change in risk factor i and f_{zi} is the partial derivative of f with respect to z_i.

Given portfolio weights $w_i = S_t^i / V_t$, the linearized absolute loss can be expressed as:

$$L_{\Delta\text{abs}}(t;\ \Delta t) = -V_t \sum_{i=1}^{d} w_i X_t^i = -V_t w' X_t \tag{2.5}$$

Often, the relative loss, defined as $L_{rel}(t; \Delta t) = -[V_{t+\Delta t} - V_t] / V_t$, is of interest. The linearized relative loss is simply:

$$L_{\Delta \mathrm{rel}}(t;\ \Delta t) = -\sum_{i=1}^{d} w_{\mathrm{i}}\, X_{\mathrm{t}}^{\mathrm{i}} = -w' X_{\mathrm{t}} \qquad (2.6)$$

Value-at-Risk (VaR)

Assuming a filtration (F_t) $t \in R$ to which Z_t is adapted, the unconditional absolute VaR is defined as:

$$VaR_{\mathrm{abs}}(t, \Delta t;\ \alpha) = inf\{l \mid \mathbb{P}(L_{\mathrm{abs}}(t;\ \Delta t) > l) \leq 1 - \alpha\} \qquad (2.7)$$

Similarly, the conditional absolute VaR is:

$$VaR_{cond}^{abs}(t, \Delta t;\ \alpha) = inf\{l \mid \mathbb{P}(L_{\mathrm{abs}}(t;\ \Delta t) > l \mid F_{\mathrm{t}}) \leq 1 - \alpha\} \qquad (2.8)$$

Analogous definitions hold for the relative VaR. In this work, we primarily focus on the relative VaR based on the linearized loss. For notational convenience, we denote this as $\mathrm{VaR}\alpha(t)$.

2.5 Backtesting VaR

To assess the performance of a VaR model, backtesting is employed. An exception occurs when the actual loss exceeds the predicted VaR. Various tests evaluate the accuracy of VaR models.

- **Kupiec's Proportion of Failure (POF) Test:** Compares the observed number of exceptions to the expected number based on the VaR confidence level.
- **Christofferson Interval Forecast (IF) Test:** Examines if the occurrence of an exception on one day influences the probability of an exception on the following day.
- **Haas' Time Between Failures (TBF) Test:** Assesses the distribution of time intervals between exceptions, which should be geometrically distributed under correct model specification.

These tests provide insights into the model's accuracy in predicting extreme losses and the independence of exceptions.

Chapter 3

Generative Adversarial Networks

This chapter explains how to use an algorithm called a Generative Adversarial Network (GAN) to predict future stock returns. It starts by introducing the basic concept of GANs and then goes into detail about the specific goals of the algorithm (called objective functions) and ways to improve how it works.

3.1 Introduction to GANs

Generative Adversarial Networks (GANs) are a type of generative model introduced by (Goodfellow et al., 2014). A GAN consists of two main parts: a generator and a discriminator.

- Discriminator (D): The discriminator's job is to determine whether the data it receives is real (from the actual training data) or fake (created by the generator). It tries to classify the data as either "real" or "fake."
- Generator (G): The generator aims to produce data that is so realistic that it can fool the discriminator into thinking it's real. It does this by trying to match the distribution of the real training data.

Each part has a cost function, which is a mathematical way of measuring how well it's performing:

- Generator's Cost Function: The generator works to minimize its cost function, which depends on both the discriminators and its own parameters (settings), but it can only control its own.
- Discriminator's Cost Function: Similarly, the discriminator minimizes its own cost function by adjusting its parameters to better distinguish between real and fake data.

To create new data, the generator takes a simple random input, called a latent variable (z), and transforms it into an output that resembles real data. The generator and discriminator are usually implemented using neural networks. The detailed architecture of these neural networks is explained later in section 4.2, and a visual representation of a GAN is provided in Figure 3.1.

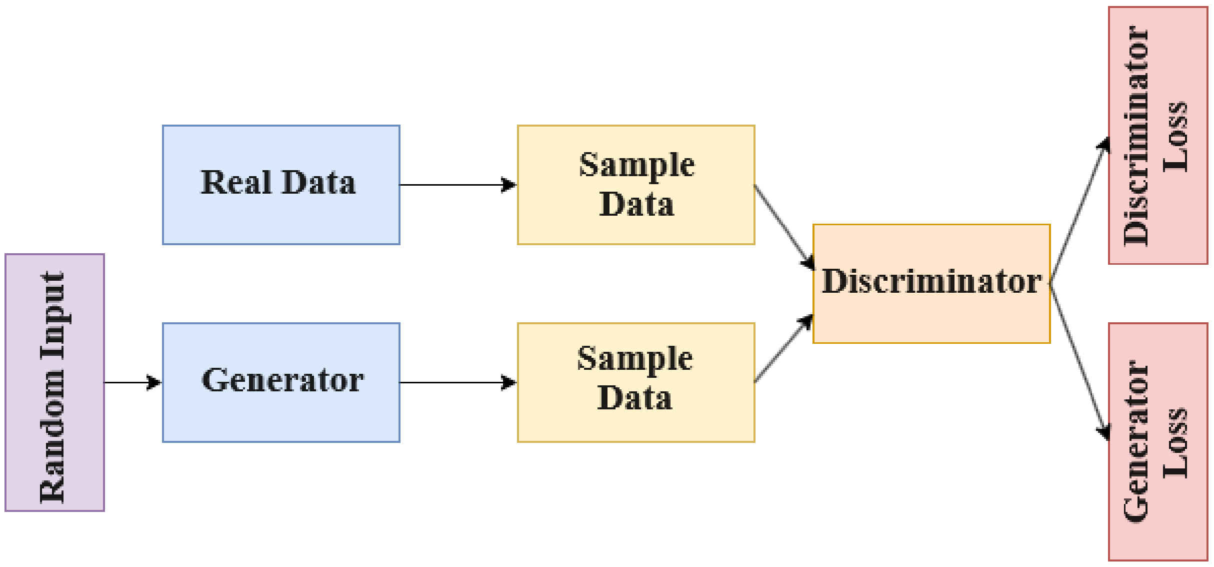


Figure 3.1: Illustration of a generative adversarial network.

3.2 Objective Functions

Let's dive deeper into the objective functions used in GANs, introducing some mathematical definitions.

In GANs, the term 'cost function' typically refers to a function that needs to be minimized. However, since optimization problems can involve either minimization or maximization, we use the more general term 'objective function' to describe both.

Discriminator's Objective Function

The discriminator's objective function $J_D(\theta_D, \theta_G)$ was originally introduced by (Goodfellow et al., 2014). The goal is to maximize this objective function, which can also be viewed as minimizing the negative of the objective function $J_D(\theta_D, \theta_G)$ if expressed as a cost function.

The discriminator's objective function is defined as:

$$J_D(\theta_D, \theta_G) = \mathbb{E}_{x\sim pdata}[logD(x)] + \mathbb{E}_z\left[\log\left(1 - D\left(G(z)\right)\right)\right] \quad (3.1)$$

Here:

- D(x) is the discriminator's probability estimate that a real data sample x is real.
- G(z) is the data generated by the generator from the latent variable z.
- $\mathbb{E}_{x\sim pdata}[\cdot]$ shows the predicted value relative to the actual distribution of data.
- $\mathbb{E}_z[\cdot]$ represents the expected value over the latent variable z.

The discriminator is trained using two sets of data: real data (labelled as 1) and generated data (labelled as 0). The objective function $J_D(\theta_D, \theta_G)$ is maximized when the discriminator assigns probabilities close to 1 for real data x and close to 0 for generated data $G(z)$.

Generator's Objective Function

The generator's objective function $J_G(\theta_D, \theta_G)$ is minimized by the generator, as defined by Goodfellow et al. (2014):

$$J_G(\theta_D, \theta_G) = \mathbb{E}_z\left[\log\left(1 - D\big(G(z)\big)\right)\right] \tag{3.2}$$

Here:

• The generator G aims to minimize this objective function by making D(G(z)) as close to 1 as possible.
• $\mathbb{E}_z\left[\log\left(1 - D\big(G(z)\big)\right)\right]$measures how well the discriminator identifies the generated data as fake.

To put it briefly, the generator generates data G(z) in an attempt to "trick" the discriminator by making it assign high probabilities (almost 1) to fictitious data, treating it as genuine. By accurately classifying created data as false and genuine data as real, the discriminator hopes to maximise its objective function in the interim.

3.3 Wassersterstein GAN

The Wasserstein GAN (WGAN), developed by (Arjovsky, Chintala and Bottou, 2017), is a variation of the traditional GAN designed to improve stability and make the model less sensitive to architectural choices. The main difference in WGAN is the use of a different objective function. Instead of calling it a "discriminator," they refer to it as a "critic" because it no longer outputs a value between 0 and 1. The objective function in WGAN is based on the Earthmover distance, also known as the Wasserstein-1 distance.

To satisfy certain mathematical properties (specifically, making the critic a 1-Lipschitz continuous function), they minimize the following function:

$$J_D(\theta_D, \theta_G) = \mathbb{E}_z\big[D\big(G(z)\big)\big] - \mathbb{E}_{x\sim \mathrm{pdata}}[D(x)] \tag{3.3}$$

For the generator, the goal is to minimize:

$$J_G(\theta_D, \theta_G) = -\mathbb{E}_z\big[D\big(G(z)\big)\big] \tag{3.4}$$

To ensure the critic meets the Lipschitz condition, WGAN initially used weight clipping, which means limiting the critic's weights within a certain range [−c, c]. However, the authors noticed that this method has some drawbacks.

To improve WGAN, (Gulrajani et al., 2017) proposed a new method called WGAN-GP, which uses a gradient penalty instead of weight clipping. They noted that a function is 1-Lipschitz if its gradient norm is at most 1 everywhere. So, they added a penalty term to the critic's objective function to enforce this constraint:

$$J_{\mathrm{D}}(\theta_{\mathrm{D}}, \theta_{\mathrm{G}}) = \mathbb{E}_{\mathrm{z}}[D(G(z))] - \mathbb{E}_{\mathrm{x\sim pdata}}[D(x)] + \lambda \mathbb{E}_{\hat{\mathrm{x}}}[(\| \nabla_{\hat{\mathrm{x}}} D(\hat{x}) \| 2 - 1)^2] \quad (3.5)$$

This penalty term applies to randomly chosen points between real and generated data. The penalty encourages the gradient to stay close to 1, rather than just staying below 1. In their experiments, Gulrajani et al. found that this approach works well across different architectures and image datasets, and they typically set the penalty coefficient λ to 10.

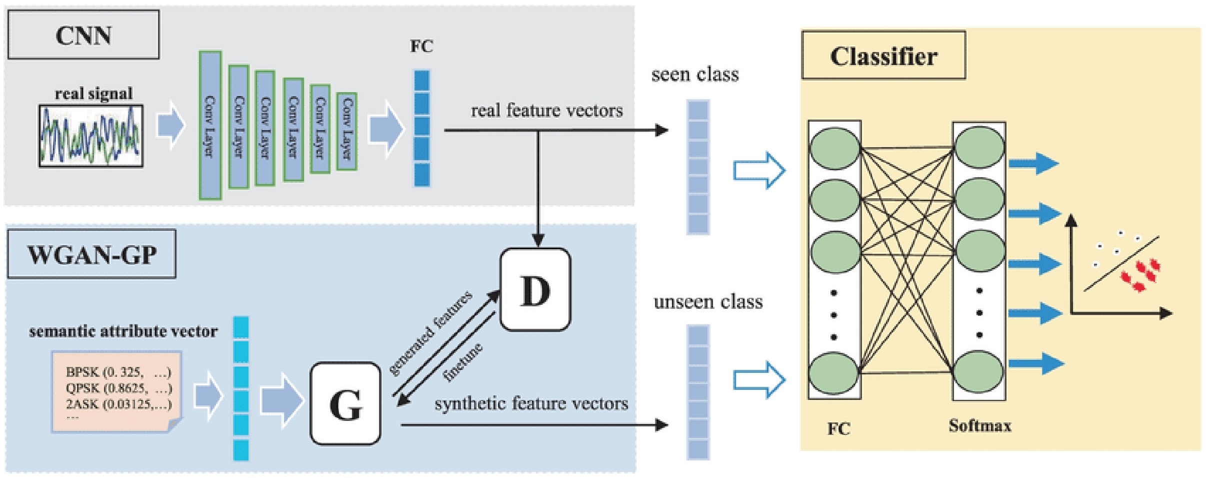


Figure 3.2 Architecture Diagram of the Proposed Model

The Figure3.2 represents a framework where WGAN-GP is used to generate synthetic features that mimic real ones, and a CNN is used to extract features from real signals. These features are then used to train a classifier, with a specific focus on handling both seen and unseen classes effectively. This approach is particularly useful in scenarios like Zero-Shot Learning (ZSL), where the model needs to classify data from classes that were not present in the training data.

3.4 Why GAN for Var Estimation?

The theoretical justification for using GANs in VaR estimation can be anchored in their ability to minimize divergences between probability distributions. By learning the underlying distribution of asset returns, GANs can generate new data that closely follows the real market behaviour, thus providing more accurate estimates of potential losses.

Moreover, the Wasserstein GAN (WGAN) variant further enhances this capability by stabilizing the training process and providing a meaningful measure of distance between distributions, known as the Earthmover (or Wasserstein) distance. This distance is particularly relevant in finance, as it accounts for the actual "cost" of transforming one distribution into another, thereby aligning with the concept of measuring risk (potential loss) in financial portfolios.

In summary, the integration of GANs into VaR estimation not only addresses the inherent limitations of traditional methods but also offers a theoretically sound and practically robust framework for managing financial risk in complex and dynamic markets. This approach, combining the statistical rigor of VaR with the advanced modelling capabilities of GANs, represents a significant step forward in the field of risk management.

Chapter 4

Data and Methodology

This chapter explains how the effectiveness of using GANs (Generative Adversarial Networks) for estimating Value at Risk (VaR) was tested. Training GANs can be challenging, and there are many ways to set them up. After some initial experiments with the data, several models were selected for VaR estimation and were back tested. The chapter covers these models and the testing process. It starts by discussing the data and software tools used, then introduces the different types of GAN models considered for VaR estimation. Finally, it describes how the models were trained and back tested, including comparisons with a benchmark model.

4.1 Data Preparation and Software

The programming for this project was done using Python 3, along with the machine learning libraries TensorFlow and Keras. The WGAN-GP model was implemented based on an example provided by (Nain, 2020).

To evaluate how well GANs could estimate Value at Risk (VaR), stock price data was obtained from Yahoo! Finance, covering a period from December 22, 2010, to December 30, 2019. This period was chosen to encompass various market conditions, including bull and bear markets, to ensure a comprehensive evaluation of the models. A portfolio of five stocks from the Nasdaq OMX Stockholm large cap list, chosen for their good data availability, was selected. The stocks included were Hennes & Mauritz B, Elekta B, Tele2 B, SEB A, and Investor B.

4.1.1 Descriptive statistics

Descriptive statistics were calculated for the daily returns of each stock to provide an overview of the dataset's characteristics. These statistics include the mean, standard deviation, skewness, kurtosis, and the minimum and maximum values of the daily returns. Here are the key summary statistics:

Mean Daily Return: This indicates the average daily return for each stock, reflecting the overall direction of price movement during the period.

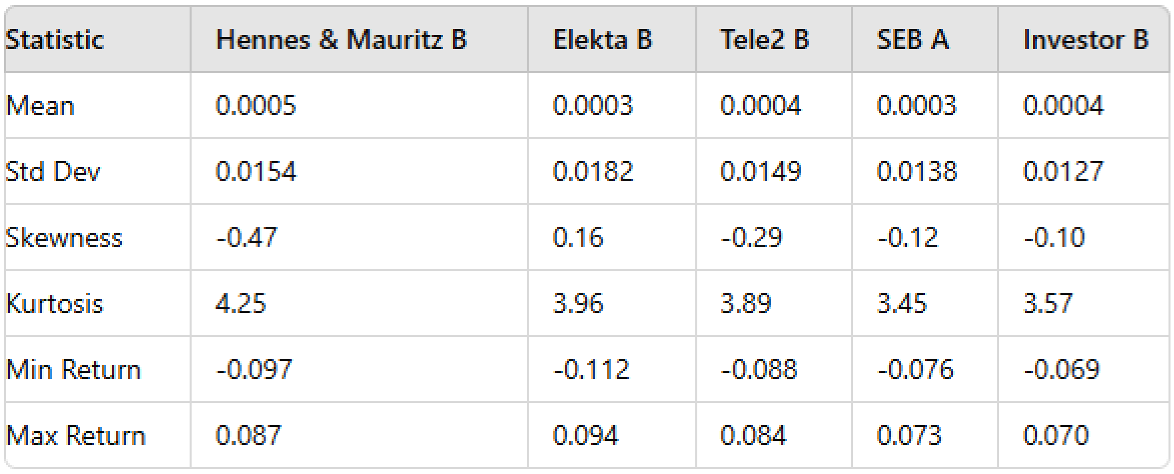
Standard Deviation: Also known as volatility, this measures the dispersion of daily returns around the mean, indicating the level of risk or uncertainty associated with the stock's returns.

Skewness: This measures the asymmetry of the return distribution. A negative skewness indicates a distribution with a longer left tail (more extreme negative returns), while a positive skewness suggests a distribution with a longer right tail (more extreme positive returns).

Kurtosis: This statistic measures the "tailedness" of the return distribution. A high kurtosis value indicates the presence of extreme outliers, while a low value suggests fewer outliers.

The descriptive statistics of the dataset are summarized in Table 4.1:

Table 4.1 Descriptive Statistics



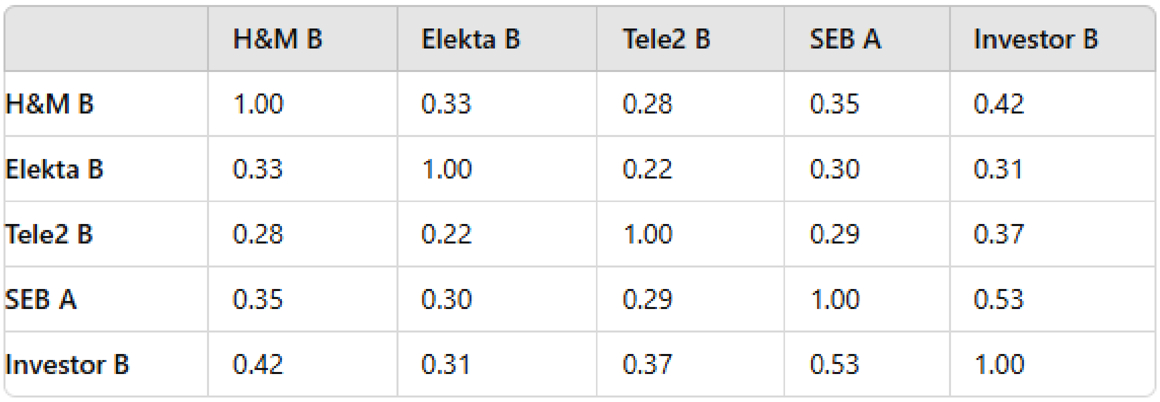
Statistic	Hennes & Mauritz B	Elekta B	Tele2 B	SEB A	Investor B
Mean	0.0005	0.0003	0.0004	0.0003	0.0004
Std Dev	0.0154	0.0182	0.0149	0.0138	0.0127
Skewness	-0.47	0.16	-0.29	-0.12	-0.10
Kurtosis	4.25	3.96	3.89	3.45	3.57
Min Return	-0.097	-0.112	-0.088	-0.076	-0.069
Max Return	0.087	0.094	0.084	0.073	0.070

Correlation Matrix

A correlation matrix was computed to assess the relationships between the daily returns of the selected stocks. Understanding these correlations is crucial as it provides insights into how the returns of different stocks in the portfolio move relative to each other, which impacts the overall portfolio risk.

The correlation matrix shows that the returns are positively correlated, with SEB A and Investor B exhibiting the highest correlation (0.53), which suggests that these stocks may behave similarly in certain market conditions.

Table 4.2 Correlation Matrix



	H&M B	Elekta B	Tele2 B	SEB A	Investor B
H&M B	1.00	0.33	0.28	0.35	0.42
Elekta B	0.33	1.00	0.22	0.30	0.31
Tele2 B	0.28	0.22	1.00	0.29	0.37
SEB A	0.35	0.30	0.29	1.00	0.53
Investor B	0.42	0.31	0.37	0.53	1.00

4.1.2 Preprocessing Steps

Several preprocessing steps were undertaken to prepare the data for training the GAN models:

1. Data Cleaning: The raw data was cleaned to remove any missing values or outliers that could distort the model's learning process. Any days with missing price data for any of the stocks were removed from the dataset.

2. Return Calculation: The daily returns were calculated using the formula:

$$r_t = \frac{s_t - s_{t-1}}{s_{t-1}} \tag{4.1}$$

where s_t is the stock price at time t, and s_{t-1} is the stock price on the previous trading day.

3. Standardization: To ensure that the input data for the GAN models had a consistent scale, all return data was standardized to have a mean of 0 and a standard deviation of 1, using the following formula:

$$r_t^s = \frac{r_t - \mu}{\sigma} \tag{4.2}$$

where r_t is the unstandardized return, μ is the mean of the pretraining returns, and σ is the standard deviation of the pretraining returns. This step was crucial for stabilizing the training process of the GAN models.

4. Portfolio Construction: An equally weighted portfolio was constructed, with each of the five stocks having a weight of 0.2. The portfolio return on any given day was calculated as the weighted average of the individual stock returns.

These preprocessing steps were essential to ensure that the data was clean, consistent, and suitable for training the GAN models, ultimately contributing to more accurate and reliable Value at Risk (VaR) estimates.

4.2 GANs for Value at Risk

In this thesis, Value at Risk (VaR) was estimated by training a generative model, specifically a GAN, with the goal of learning the distribution of returns for a portfolio of stocks. The trained generator was then used to simulate future portfolio returns. To estimate VaR, the approach involved calculating the a% VaR as the negative of the a^{th} percentile of the simulated returns. This is because VaR represents the potential loss in the value of a portfolio. For this study, the 95% daily VaR was chosen. This specific level of VaR was selected to ensure a relatively large number of observations and expected failures during the backtesting process, providing more reliable results compared to using VaR for a longer time horizon or with a higher confidence level.

For each VaR estimate generated using the GAN, 1,000 simulations were run. This large number of simulations was aimed at accurately capturing the distribution of potential future returns and ensuring that the estimated VaR was robust.

Two different types of GAN models were considered for VaR estimation:

1. Conditional Model: In this approach, the generator (G) learns the distribution of returns conditioned on previous returns. This means that, in addition to the latent input (a random vector typically used in GANs), the generator also takes the previous day's returns as input. The idea is that by considering past returns, the generator can produce more accurate predictions of the return distribution, potentially improving the VaR estimates.

2. Unconditional Model: In contrast to the conditional model, the unconditional model does not consider past returns. The generator only learns the overall distribution of returns based on the latent input, without incorporating any additional information about the historical returns. This model focuses on capturing the general distribution of returns without being influenced by recent market movements.

Both types of GAN models were subjected to backtesting to assess their performance. The backtesting was conducted using Kupiec's test, a statistical test that evaluates the accuracy of VaR models by comparing the predicted VaR levels with actual portfolio losses over a given period. The test helps determine if the number of times losses exceed the predicted VaR matches the expected frequency.

As a benchmark for comparison, a traditional variance-covariance model was also considered. This model is a well-established method for estimating VaR and provides a useful point of reference to assess the performance of the GAN-based models.

The chapter will proceed to discuss in detail the specific types of GANs considered for VaR estimation, explaining how they were implemented and evaluated in this context.

4.2.1 Conditional Model

Mariani et al. (2019) introduced a conditional GAN model for portfolio optimization, and a similar approach was adopted in this thesis for Value at Risk (VaR) estimation. In their model, the generator G takes as input both the latent variable and a series of b days of previous stock prices. The goal is for the generator to learn the distribution of stock prices for the following f days. The data is structured into matrices H, which contain w days of prices split into two parts: matrix H_b, containing the prices for the b previous days, and matrix H_f, containing the prices for the f following days. The generator then generates simulated future prices $\hat{H}_f$ using the previous prices H_b and the latent input z:

$$\hat{H}_f = G(H_b, z) \tag{4.3}$$

In Mariani et al.'s model, the generator is divided into two parts:

1. Conditioning Part: This part processes the conditioning data H_b using several 1D convolutional layers followed by a dense layer. This processed data is then concatenated with the latent vector (and a vector of analysis values they calculated).

2. Simulator: The concatenated output from the conditioning part is used as input for the simulator, which generates the simulated future prices $\hat{H}_f$

The discriminator, on the other hand, takes either real data H or fake data, which is the concatenation of H_b and $\hat{H}_f$, as its input. The discriminator's goal is to distinguish between real and generated data.

In our work, a similar approach is used for VaR estimation. The data is represented as $w \times k$ matrices H of stock returns, where each matrix contains w days (rows) of returns for k stocks (columns). For conditioning, a backwards window of b = 40 days is used, and a forwards window of f = 1 day is considered, since the focus is on one-day VaR estimation. The portfolio consists of k = 5 stocks.

Generator Architecture (Conditional Model)

The generator architecture for the conditional model is detailed in Table 4.3:

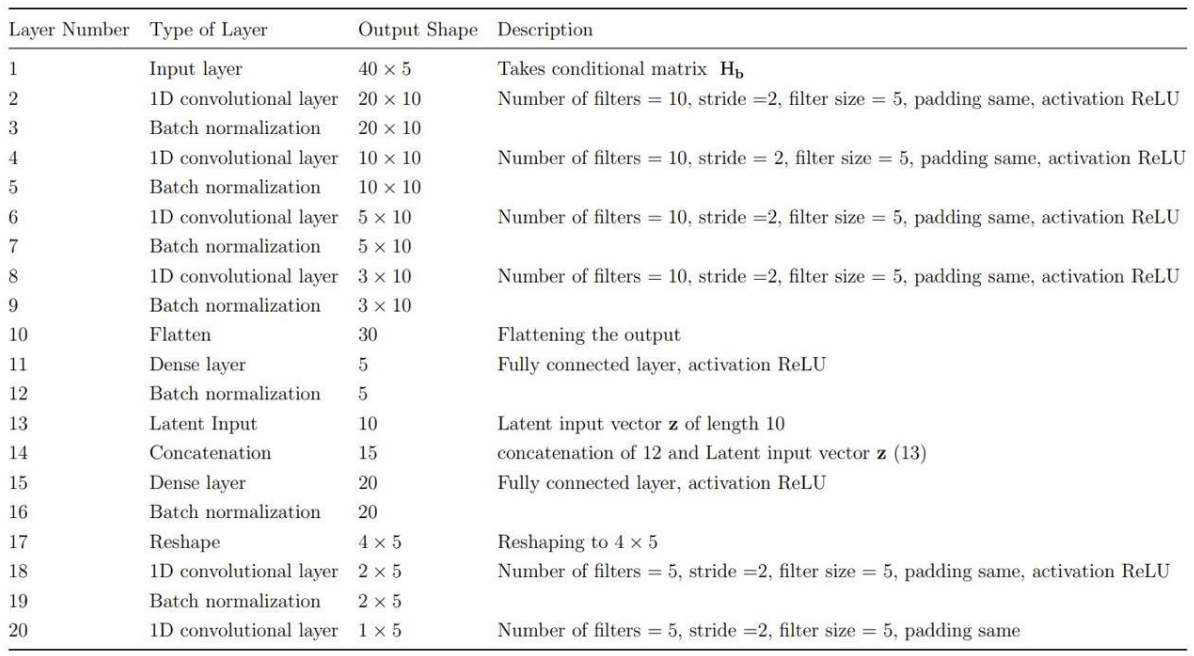
- Input Layer: The generator starts with an input layer that takes the conditional matrix H_b, which has a shape of 40×5.

Convolutional Layers: This is followed by four 1D convolutional layers, each with batch normalization. These layers apply convolutions over the rows (time dimension) to extract features from the time-series data.

- Flatten and Dense Layers: The output from the convolutional layers is then flattened and passed through a fully connected (dense) layer, followed by another batch normalization layer.
- Latent Input: The generator also takes a latent input vector z of length 10, which is concatenated with the output from the previous layers.
- Final Layers: After concatenation, the combined input is passed through another dense layer with batch normalization. The output is then reshaped and passed through two more 1D convolutional layers (with batch normalization after the first) to reduce it to the desired output shape.

The final output of the generator $\hat{H}_f$ is a vector of simulated returns for the five stocks in the portfolio.

Table 4.3: Generator conditional model.



Layer Number	Type of Layer	Output Shape	Description
1	Input layer	40×5	Takes conditional matrix $\mathbf{H_b}$
2	1D convolutional layer	20×10	Number of filters = 10, stride =2, filter size = 5, padding same, activation ReLU
3	Batch normalization	20×10	
4	1D convolutional layer	10×10	Number of filters = 10, stride = 2, filter size = 5, padding same, activation ReLU
5	Batch normalization	10×10	
6	1D convolutional layer	5×10	Number of filters = 10, stride =2, filter size = 5, padding same, activation ReLU
7	Batch normalization	5×10	
8	1D convolutional layer	3×10	Number of filters = 10, stride =2, filter size = 5, padding same, activation ReLU
9	Batch normalization	3×10	
10	Flatten	30	Flattening the output
11	Dense layer	5	Fully connected layer, activation ReLU
12	Batch normalization	5	
13	Latent Input	10	Latent input vector $\mathbf{z}$ of length 10
14	Concatenation	15	concatenation of 12 and Latent input vector $\mathbf{z}$ (13)
15	Dense layer	20	Fully connected layer, activation ReLU
16	Batch normalization	20	
17	Reshape	4×5	Reshaping to 4×5
18	1D convolutional layer	2×5	Number of filters = 5, stride =2, filter size = 5, padding same, activation ReLU
19	Batch normalization	2×5	
20	1D convolutional layer	1×5	Number of filters = 5, stride =2, filter size = 5, padding same

Discriminator Architecture (Conditional Model)

The discriminator architecture for the conditional model is detailed in Table 4.4:

- Input Layer: The discriminator takes as input a 41×5 matrix, which can be either real data H or fake data (the concatenation of H_b and $\hat{H}_f$
- Convolutional Layers: The input is processed through five 1D convolutional layers, each using leaky ReLU as the activation function. These layers progressively increase the number of filters, extracting more complex features from the data.

- Flatten and Dense Layer: The output from the convolutional layers is flattened and passed through a final fully connected layer.

The discriminator's role is to assign a higher score to data that it believes comes from the true distribution (real data) and a lower score to data that it believes is generated by the generator. This scoring helps in training the generator to produce more realistic data over time.

Table 4.4: Discriminator conditional model.

Layer Number	Type of Layer	Output Shape	Description
1	Input layer	41×5	Takes a matrix $\mathbf{H}$ or the concatenation of $\mathbf{H_b}$ and $\hat{\mathbf{H}}_\mathbf{f}$
2	1D convolutional layer	21×10	Number of filters = 10, stride =2, filter size = 5, padding same, activation leaky ReLU
3	1D convolutional layer	11×20	Number of filters = 20, stride =2, filter size = 5, padding same, activation leaky ReLU
4	1D convolutional layer	6×40	Number of filters = 40, stride =2, filter size = 5, padding same, activation leaky ReLU
5	1D convolutional layer	3×80	Number of filters = 80, stride =2, filter size = 5, padding same, activation leaky ReLU
6	1D convolutional layer	2×160	Number of filters = 160, stride =2, filter size = 5, padding same, activation leaky ReLU
7	1D Flatten	320	Flattening the output
8	Dense layer	1	Fully connected layer

4.2.2 Unconditional Model

The second type of GAN model used in this thesis is an unconditional model. In this model, the generator G only takes a latent input z to generate simulated stock returns $\hat{H}_f$ without relying on previous stock returns as input. The formula for generating the simulated returns is:

$$\hat{H}_f = G(z) \tag{4.4}$$

Since the model does not condition on past returns, each observation H is represented as a 5-dimensional vector of returns corresponding to the five stocks in the portfolio. The discriminator in this model takes either real return vectors H or the generated return vectors $\hat{H}_f$ and tries to distinguish between them.

Generator Architecture (Unconditional Model)

The architecture of the generator for the unconditional model is described in Table 4.5:

1. Input Layer: The generator starts with an input layer that takes a latent input vector z of size 20.

2. Dense Layer: The input is passed through a fully connected (dense) layer with 160 units, using ReLU as the activation function. This layer expands the latent input into a higher-dimensional space.

3. Batch Normalization: The output of the dense layer is normalized using batch normalization to stabilize the training process and improve convergence.

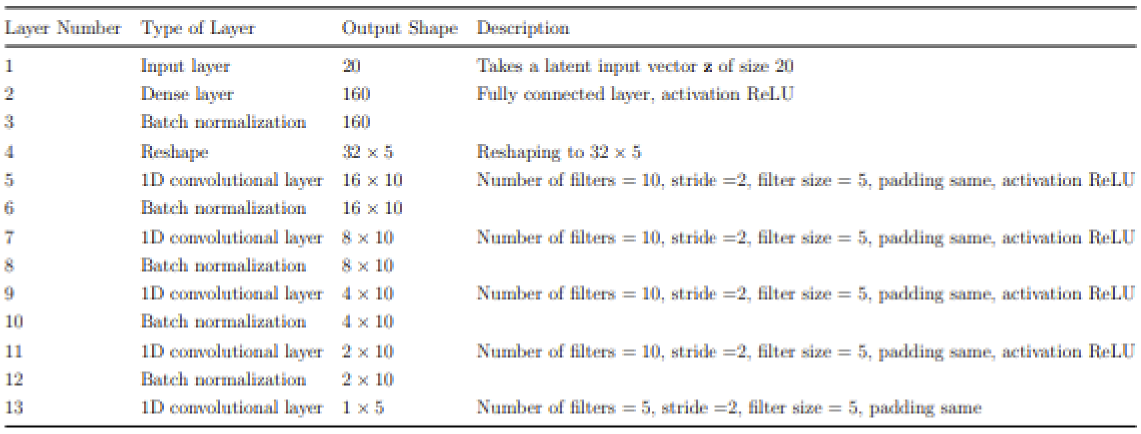
4. Reshape Layer: The normalized output is reshaped into a 32×5 matrix, preparing it for the subsequent convolutional layers.

5. 1D Convolutional Layers: The reshaped output is processed through a series of 1D convolutional layers, each followed by batch normalization. These layers progressively extract features and reduce the dimensions until the final output shape is achieved. The specifics are as follows:

- First, a convolutional layer with 10 filters, a stride of 2, a filter size of 5, and ReLU activation reduces the output to 16×10.
- Subsequent layers continue to halve the size along the time dimension while maintaining the number of filters at 10 until the final layer, which reduces the output to a 1×5 vector representing the simulated returns for the five stocks.

6. Final Output: The final output layer produces a 1×5 vector, which represents the simulated one-day returns for the five stocks in the portfolio.

Table 4.5: Generator unconditional model.



Layer Number	Type of Layer	Output Shape	Description
1	Input layer	20	Takes a latent input vector **z** of size 20
2	Dense layer	160	Fully connected layer, activation ReLU
3	Batch normalization	160	
4	Reshape	32×5	Reshaping to 32×5
5	1D convolutional layer	16×10	Number of filters = 10, stride =2, filter size = 5, padding same, activation ReLU
6	Batch normalization	16×10	
7	1D convolutional layer	8×10	Number of filters = 10, stride =2, filter size = 5, padding same, activation ReLU
8	Batch normalization	8×10	
9	1D convolutional layer	4×10	Number of filters = 10, stride =2, filter size = 5, padding same, activation ReLU
10	Batch normalization	4×10	
11	1D convolutional layer	2×10	Number of filters = 10, stride =2, filter size = 5, padding same, activation ReLU
12	Batch normalization	2×10	
13	1D convolutional layer	1×5	Number of filters = 5, stride =2, filter size = 5, padding same

Discriminator Architecture (Unconditional Model)

The architecture of the discriminator for the unconditional model is outlined in Table 4.6:

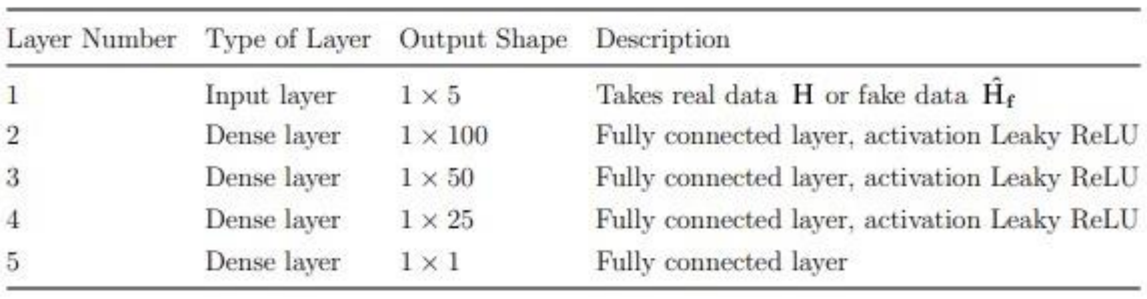
1. Input Layer: The discriminator starts with an input layer that takes either the real data H or the fake data $\hat{H}_f$ Each input is a 1×5 vector representing the returns of the five stocks.

2. Fully Connected Layers: The input is passed through a series of fully connected (dense) layers with decreasing sizes:

- The first dense layer has 100 units and uses Leaky ReLU as the activation function.
- The subsequent layers progressively reduce the number of units: 50, 25, and finally, 1 unit in the output layer.

3. Final Output: The output of the last dense layer is a single value that represents the critic score. The discriminator aims to assign higher scores to real data and lower scores to generated data, helping to train the generator to produce more realistic simulations.

Table 4.6: Discriminator unconditional model.



Layer Number	Type of Layer	Output Shape	Description
1	Input layer	1×5	Takes real data $\mathbf{H}$ or fake data $\hat{\mathbf{H}}_\mathbf{f}$
2	Dense layer	1×100	Fully connected layer, activation Leaky ReLU
3	Dense layer	1×50	Fully connected layer, activation Leaky ReLU
4	Dense layer	1×25	Fully connected layer, activation Leaky ReLU
5	Dense layer	1×1	Fully connected layer

In summary, the unconditional model's generator only relies on a latent input vector to generate a simulated return vector for the portfolio, while the discriminator uses fully connected layers to evaluate and differentiate between real and simulated returns. This model allows for the generation of new return data without being influenced by past market behaviour, focusing purely on the learned distribution of returns.

4.3 Training and Backtesting of Models

To evaluate the effectiveness of Generative Adversarial Networks (GANs) for estimating Value at Risk (VaR), this study tested five distinct GAN-based VaR models, alongside a traditional variance-covariance benchmark model. The evaluation was performed on daily return data spanning from December 22, 2010, to December 30, 2019. The models were assessed using Kupiec's Proportion of Failures (POF) test at a 5% significance level, meaning that any model with a p-value below 5% would be deemed inadequate.

Model Overview and Training

General Approach:

- Training Data: GAN models were trained using historical return data prior to the backtesting period.
- Conditional Models: These models were periodically updated during backtesting with additional training on recent data, similar to the approach used by Fiechtner (2019). Specifically, the models were retrained every 10 days with 100 additional epochs to incorporate recent market conditions.

Detailed Description of Each Model:

1. Conditional GAN Model (Model 1):

- Initial Training: This model was trained for 2000 epochs using daily return matrices from January 4, 2000, to December 18, 2009.

- Data Structure: The training data consisted of a matrix H with dimensions 41×5, where the first 40 rows represented the conditioning matrix H_b (i.e., returns from the previous 40 days).
- VaR Estimation: During backtesting, the model estimated VaR by running 1000 simulations. For each simulation, the most recent 40 days of returns H_b were used as input.

2. Unconditional GAN Model (Model 2):

- Pretraining: This model was initially trained for 2000 epochs using returns data from February 29, 2000, to December 18, 2009.
- Additional Training: Every five days during the backtesting period, the model was further trained for 100 epochs using the most recent 252 days of returns. This allowed the model to adapt to recent market conditions while balancing computational efficiency.

3. Unconditional GAN Model (Model 3):

- Pretraining: This model was pretrained for 1000 epochs using return data from December 22, 2005, to December 18, 2009.
- Additional Training: Similar to Model 2, but updates were made every five days. For these updates, the model was retrained for 200 epochs, starting from the most recent model parameters rather than the initial pretrained state. This approach aimed to capture more recent market dynamics.

4. Unconditional GAN Model (Model 4):

- Training: This model followed the same approach as Model 2, but updates were performed every second day instead of every five days. The model was trained for 20 additional epochs with each update, providing a more frequent adaptation to recent returns.

5. Unconditional GAN Model (Model 5):

- Training: Similar to Model 4, but updates were made daily to reflect the most recent data. The step size parameter α was set to 2×10^{-5} to allow for faster adjustments due to the daily updates, addressing the increased computational cost associated with daily retraining.

6. Benchmark Model (Model 6):

- Traditional Variance-Covariance Approach: This model used the classic variance-covariance method for VaR estimation. Each estimate was based on the most recent 252 days of return data, serving as a baseline for comparison against the GAN models.

Common Settings Across GAN Models:

Model Type: All GAN models utilized the Wasserstein GAN with Gradient Penalty (WGAN-GP) approach, known for its stability in training by penalizing the gradients of the critic.

Discriminator Training: For each step of generator training, the discriminator was trained for five steps to ensure effective adversarial learning.

Batch Size and Optimizer: A batch size of 36 was used for all GAN models, and the Adam optimizer was employed with β_1= 0.5 and β_2= 0.9, as recommended by Gulrajani et al. (2017).

Penalty Term: The penalty term in the discriminator's objective function was set to 10.

Standardization of Data: All return data was standardized to have a mean of 0 and a standard deviation of 1 based on the pretraining data. This was done using the formula:

$$r_t^s = \frac{r_t - \mu}{\sigma} \tag{4.5}$$

where r_t is the unstandardized return, μ is the mean of the pretraining returns, and σ is the standard deviation of the pretraining returns.

Latent Variable z: The latent variable z for the GAN models followed a standard multivariate normal distribution to provide a diverse set of inputs for generating simulations.

The study aimed to determine the suitability of GANs for VaR estimation by comparing several GAN models against a traditional variance-covariance benchmark. The models were evaluated based on their ability to estimate VaR accurately and their adaptability to recent market conditions through periodic updates and retraining. The comparison with the traditional method provided insights into the potential advantages and limitations of using GANs for financial risk management.

4.4 Hyperparameter Tuning Process

The performance of Generative Adversarial Networks (GANs) is highly sensitive to the choice of hyperparameters, which include aspects such as learning rate, batch size, the number of epochs, and architecture-specific parameters like the number of convolutional layers or the latent vector's dimensionality. In this study, a systematic approach was used to tune the hyperparameters to ensure the models' stability and effectiveness in estimating Value at Risk (VaR). The tuning process involved several key steps:

1. Initial Parameter Selection

- **Literature Review:** The initial hyperparameters were selected based on values commonly used in the literature. For instance, the Adam optimizer with parameters $\beta 1=0.5$and $\beta 2=0.9$ was chosen, following recommendations from Gulrajani et al. (2017) for the WGAN-GP architecture.
- **Default Values:** Default values were initially chosen for the learning rate ($\alpha=10^{-4}$), the penalty term ($\lambda=10$), and the batch size (36), as these have been reported to provide a good balance between training stability and model performance.

2. Grid Search and Manual Adjustments

- **Grid Search:** A grid search was conducted for critical hyperparameters such as learning rate, batch size, and the number of convolutional layers. Specifically:
    - **Learning Rate (α):** Tested values included $10^{-3,}$ 10^{-4}, and 10^{-5}.
    - **Batch Size:** Explored sizes were 16, 32, 36, and 64.
    - **Number of Convolutional Layers:** Evaluated configurations ranged from 2 to 5 layers, with varying filter sizes and strides.
- **Manual Adjustments:** After identifying a promising range of values, manual tuning was performed to further refine the parameters, particularly focusing on minimizing the Wasserstein loss during training.

3. Cross-Validation and Stability Testing

- **Cross-Validation:** For each candidate set of hyperparameters, cross-validation was performed using the historical return data. The data was split into multiple overlapping windows to test the model's performance across different time periods.
- **Stability Testing:** Stability was a critical factor in hyperparameter selection. The models were evaluated for their ability to maintain consistent performance across various training iterations, avoiding mode collapse and ensuring smooth convergence.

4. Final Parameter Selection

- **Learning Rate (α\alphaα):** A learning rate of 2×10^{-5} was selected as it provided a balance between convergence speed and training stability, particularly for models that were updated frequently (e.g., daily updates in Model 5).
- **Batch Size:** A batch size of 36 was chosen based on its ability to facilitate stable training while maintaining computational efficiency.

Epochs and Update Frequency: The number of training epochs and the frequency of model updates during backtesting were determined through experimentation. For instance, Model 1 used 2000 epochs in initial training with updates every 10 days, while Model 5 used daily updates with 20 additional epochs each time.

5. Evaluation Metrics and Selection Criteria

- **Wasserstein Loss:** The primary criterion for model selection was the Wasserstein loss, as it directly correlates with the quality of the generated samples in WGAN-GP models.
- **Kupiec's Test Results:** The final set of hyperparameters for each model was chosen based on the Kupiec test results, ensuring that the number of violations (instances where actual losses exceeded the predicted VaR) matched the expected frequency at the 95% confidence level.

6. Robustness Check

- **Out-of-Sample Performance:** The selected hyperparameters were further validated by testing the models on out-of-sample data, ensuring that the chosen parameters generalized well to unseen market conditions.

By systematically exploring and fine-tuning these hyperparameters, the study aimed to optimize the performance of the GAN models in estimating Value at Risk, ensuring that they provided robust and accurate risk estimates compared to traditional methods.

Chapter 5

Results and Analysis

This chapter presents and discusses the results of the empirical analysis conducted on the backtesting data. First, the models being evaluated are compared visually against the actual returns. This means that the performance of the models is plotted or displayed in a way that allows for a direct visual comparison with the real-world returns to see how well the models predict or follow the actual outcomes.

After this visual comparison, the chapter moves on to discuss the results of applying Kupiec's Proportion of Failures (POF) test. This statistical test is used to evaluate the accuracy of risk models, specifically by checking if the number of times the model fails (i.e., when the model's predictions are significantly off) matches what would be expected based on a given confidence level. The discussion will likely explore whether the models are reliable or if they under- or overestimate risks based on this test.

5.1 Plots of Actual Returns Versus VaR Models

This section begins by examining the comparison between the actual returns over the test period (from December 22, 2010, to December 30, 2019) and the −VaR (Value at Risk) estimates produced by the different models under consideration, as depicted in Figure 6.1. The models are designed to predict 95% VaR, meaning that for a well-performing model, we would expect about 5% of the actual returns to fall below the −VaR line, which represents the threshold where losses are expected to be extreme.

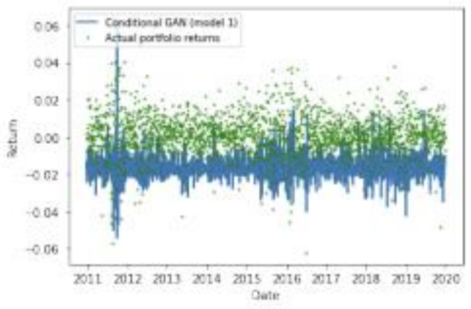
When analysing the plots:

General Observation: Across all models, it's evident that the majority of actual returns remain above the −VaR estimates. This suggests that the models generally predict the risk threshold correctly, avoiding frequent underestimations of risk.

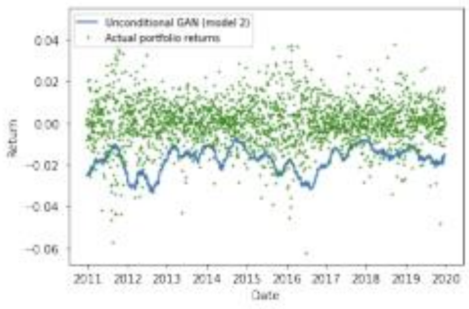
Model 1 (Conditional GAN): Figure 6.1a reveals that Model 1 stands out from the other models. The conditional GAN model was designed with the expectation that it would better capture market condition changes by basing its predictions on recent returns of the underlying assets. This means that it aims to model the distribution of returns more accurately by considering the recent past. However, the resulting plot shows a highly jagged line for the VaR estimates, indicating large fluctuations over short periods. This behaviour seems unrealistic, as it suggests excessive sensitivity to short-term market changes, which may not genuinely reflect the market's overall condition.

Other GAN Models (2-5): In contrast, models 2 through 5, while not directly conditioned on previous returns like Model 1, still incorporate recent return data through their training processes. These models show more consistent and smoother changes in VaR estimates over time. The plot of Model 5 (Figure 5.1e) shows frequent updates, which is evident from the somewhat jagged line representing the −VaR estimate. However, this does not drastically affect the overall pattern of the VaR estimates, which remain relatively stable across these models.

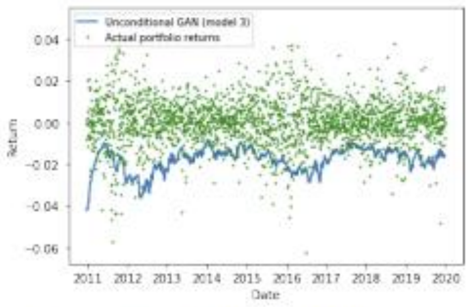
Benchmark Model (Model 6): The benchmark model, shown in Figure 5.1f, produces VaR estimates that are quite similar to those generated by the unconditional GAN models. This suggests that despite differences in the underlying methodologies, the overall risk estimates provided by the benchmark align closely with those of the GAN models.



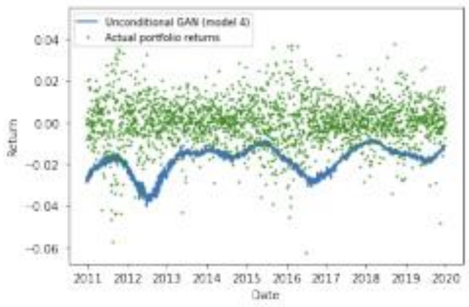
(a) Model 1 (conditional GAN) and actual returns.



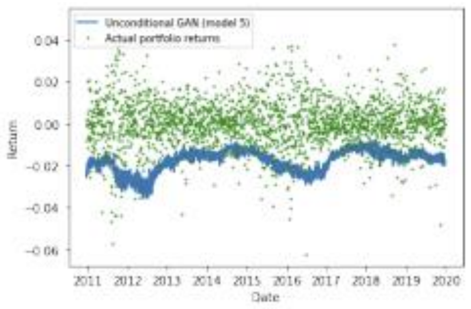
(b) Model 2 and actual returns.



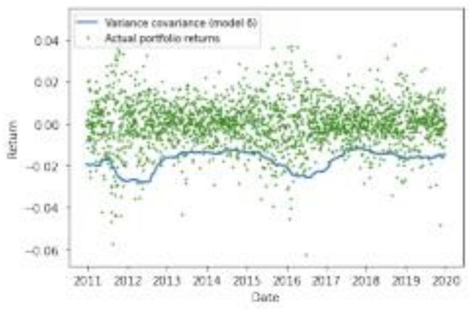
(c) Model 3 and actual returns.



(d) Model 4 and actual returns.



(e) Model 5 and actual returns.



(f) Model 6 (variance-covariance model) and actual returns.

Figure 5.1: 95% one day −VaR estimates and actual returns for the backtesting period.

The analysis highlights that while Model 1’s unique approach leads to more volatile predictions, the other models, despite their different training methodologies, produce relatively consistent and realistic VaR estimates. The benchmark model also supports the validity of the GAN models by offering similar estimates.

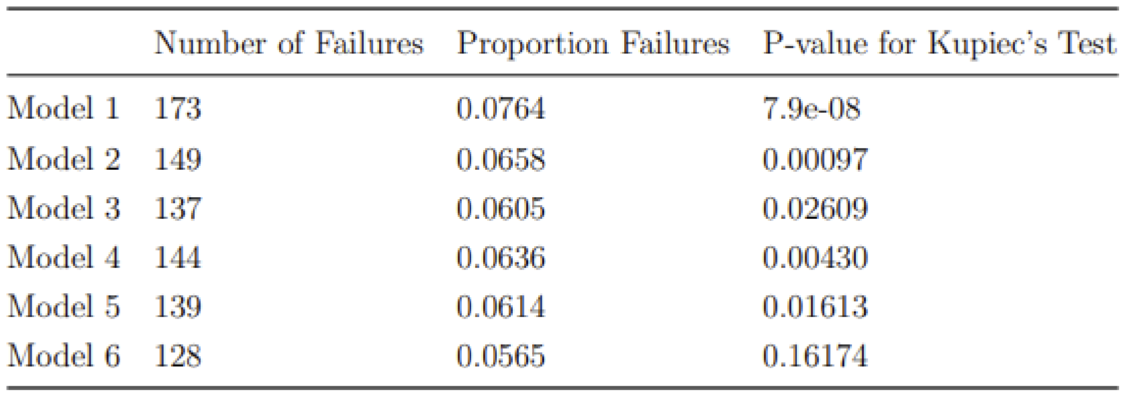
5.2 Kupiec’s POF Test

This section delves into the detailed backtesting results of six Value at Risk (VaR) models, using Kupiec’s Proportion of Failures (POF) test to evaluate their performance. The POF test is crucial for determining whether the models' predictions align with real-world outcomes, specifically checking if the observed rate of exceedances (instances where actual losses exceed the predicted VaR) matches the expected rate, which is 5% for models designed to predict 95% VaR.

5.2.1 Overview of Backtesting Results

The results, as presented in Table 5.1, reveal that all six models underestimated risk during the backtesting period. This underestimation is evidenced by a higher-than-expected proportion of failures, with the actual failure rates ranging from 5.65% to 7.64%. The variance-covariance model (Model 6) performed the best, with a failure proportion of 5.65%, closely aligning with the expected 5%. This model also passed Kupiec’s POF test, as indicated by its p-value of 0.16174, meaning it was not statistically rejected at the 5% significance level. This suggests that the variance-covariance model provided a more reliable estimation of risk compared to the other models, aligning closely with the expected outcomes and demonstrating a strong capacity to model market risk accurately over time.

Table 5.1: Backtesting results for VaR models.



	Number of Failures	Proportion Failures	P-value for Kupiec’s Test
Model 1	173	0.0764	7.9e-08
Model 2	149	0.0658	0.00097
Model 3	137	0.0605	0.02609
Model 4	144	0.0636	0.00430
Model 5	139	0.0614	0.01613
Model 6	128	0.0565	0.16174

5.2.2 Performance of GAN Models

The GAN models (Models 1-5), which are designed using generative adversarial networks, showed a range of performances. Among these, Model 3, which was pretrained on a shorter time period, had the lowest proportion of failures at 6.14%. However, despite this relatively closer alignment to the expected 5% failure rate, all GAN models were statistically rejected by Kupiec's POF test. The rejection indicates that these models consistently underestimated the risk, failing to capture the true extent of potential losses accurately.

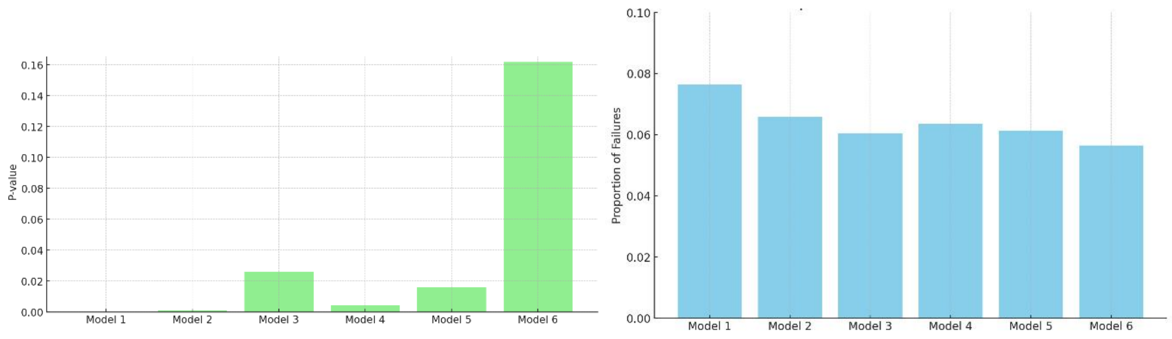


Figure 5.2: p-values from Kupiec' test and number of failures for each model.

Model 1, a conditional GAN model, exhibited the highest failure rate at 7.64%, suggesting that its approach conditioning on recent returns to adapt to changing market conditions did not improve risk prediction. The high failure rate could imply overfitting to recent data or an excessive sensitivity to short-term fluctuations, leading to less reliable VaR estimates. This model's poor performance underlines the challenges of conditioning models directly on recent data, which may introduce volatility and reduce the model's ability to generalize effectively over longer periods.

5.2.3 Period-Specific Performance Analysis:

The backtesting results were further broken down into two distinct periods—December 22, 2010, to July 2, 2015, and July 3, 2015, to December 30, 2019—to assess how these models performed under different market conditions.

1. First Period (2010-12-22 to 2015-07-02):
During this period, Models 4, 5, and 6 were not rejected by Kupiec's test, indicating they were able to capture the risk accurately during this timeframe. The success of Models 4 and 5, which are GAN models updated more frequently, suggests that during this period, more dynamic models that adapted to market conditions with frequent updates were effective. The variance-covariance model (Model 6) also performed well, further solidifying its reliability as a benchmark model.

Table 5.2: Backtesting results for the first period (2010-12-22 to 2015-07-02).

	Number of Failures	Proportion Failures	P-value for Kupiec's Test
Model 1	82	0.0724	0.00113
Model 2	79	0.0698	0.00385
Model 3	78	0.0689	0.00565
Model 4	68	0.0600	0.13130
Model 5	69	0.0610	0.06095
Model 6	70	0.0618	0.07754

2. Second Period (2015-07-03 to 2019-12-30):

In this period, only Model 1 (the conditional GAN) was statistically rejected, indicating it performed poorly relative to other models. This rejection, coupled with the high failure rate, suggests that the conditional GAN model struggled with the market dynamics of this later period, potentially due to its design that emphasized recent data conditioning.

Model 3, which struggled in the first period, showed significant improvement, nearly matching the performance of the benchmark variance-covariance model (Model 6). This shift could indicate that Model 3's training on a shorter time period made it more adaptable or better suited to the specific market conditions of this second period.

Table 5.3: Backtesting results for the second period (2015-07-03 to 2019-12-30).

	Number of Failures	Proportion Failures	P-value for Kupiec's Test
Model 1	91	0.0804	1.5e-05
Model 2	70	0.0618	0.07754
Model 3	59	0.0521	0.74507
Model 4	76	0.0671	0.06714
Model 5	70	0.0618	0.07754
Model 6	58	0.0512	0.84916

5.2.4 Training and Model Adaptability

The analysis highlights a critical trade-off in the design and training of GAN models. On one hand, training on a broader dataset allows models to generalize better, capturing diverse market conditions over time. On the other hand, there is a need for these models to be responsive to recent market changes to remain relevant in dynamic environments. This balance is delicate, too much reliance on historical data might cause the model to miss recent trends, while too much emphasis on recent data can lead to overfitting and an inability to generalize across different market conditions.

Overall, the variance-covariance model (Model 6) emerges as the most consistent and reliable across both periods, performing well under different market conditions and passing Kupiec’s POF test. In contrast, the GAN models displayed variability in their performance, with none showing consistent superiority across both periods. The conditional GAN (Model 1) particularly underperformed, highlighting the risks of over-conditioning on recent data. The findings emphasize the importance of robust model validation across different market scenarios, as well as the need to carefully balance training methodologies to ensure that models remain adaptable yet generalized. The sensitivity of Kupiec’s POF test to the number of observations further underscores the importance of thorough backtesting and the careful interpretation of test results to avoid overconfidence in any single model’s predictive ability.

5.2.5 Results Analysis

In analysing why certain GAN models outperformed others in specific scenarios within the context of your research, several factors explain these differences in performance.

Stability in Model Training

The **Wasserstein GAN (WGAN)** models, particularly those used in the unconditional setting, showed relatively stable performance across different market conditions. This can be attributed to the stability of the Wasserstein distance used in these models, which provides a smoother training process and avoids common issues like mode collapse. The use of a critic instead of a traditional discriminator ensures that the model maintains meaningful gradients even when the generator and discriminator are not perfectly balanced. This stability allows WGAN models to generalize better across different scenarios, making them more reliable in predicting VaR under diverse market conditions.

Adaptability to Market Conditions

The **conditional GAN model** was designed to better capture changing market conditions by conditioning on recent returns. While this approach theoretically allows the model to adapt to the current market environment, it also made the model more sensitive to short-term fluctuations. This sensitivity, while beneficial in stable markets, led to poor performance in more volatile or rapidly changing environments. The model's tendency to overfit to recent data, as evidenced by its high failure rate in certain periods, suggests that it struggled to generalize when faced with new or unexpected market conditions.

Handling of Temporal Dependencies

The conditional models' attempt to incorporate temporal dependencies by conditioning on recent market data was intended to improve their adaptability. However, the success of this approach varied. In scenarios where the market behavior was consistent with the recent past, these models performed well. However, in periods of sudden market shifts, the reliance on recent data became a liability, leading to less accurate risk predictions. In contrast, unconditional models, which do not condition on recent data, avoided these pitfalls by focusing on learning the broader distribution of returns. This approach proved more robust in handling unexpected market changes.

Frequency of Model Updates

Another factor contributing to the performance differences was the frequency of model updates. Models that were updated more frequently, such as the conditional models retrained every few days, demonstrated a mixed performance. Frequent updates allowed these models to incorporate the latest market data, which was beneficial in stable periods but introduced instability in volatile markets. On the other hand, models with less frequent updates showed more consistency in their performance, indicating that balancing the frequency of updates is crucial for maintaining model stability without overfitting to short-term data.

Backtesting and Generalization

The backtesting results highlighted that those models trained on a broader range of data with more diverse market conditions tended to generalize better. Unconditional models, which focused on learning from the entire dataset without being overly influenced by recent data, demonstrated better generalization across different market conditions. In contrast, conditional models, which were more heavily influenced by recent data, struggled to maintain accuracy when the market environment differed significantly from the training period.

To evaluate the performance of these models, backtesting was conducted over a period with 2164 observations, and Kupiec's Proportion of Failures (POF) test was used as the primary evaluation metric. The POF test examines whether the actual proportion of exceedances (instances where actual losses exceed the VaR estimate) matches the expected rate, which is 5% for a 95% VaR model.

The backtesting results showed that the conditional GAN model produced a significantly higher proportion of exceedances than expected, indicating that it underestimated the risk. This outcome suggests that the model's sensitivity to recent data may have led to overfitting, where the model responds too aggressively to short-term market conditions that may not reflect longer-term trends. The conditional model's jagged and fluctuating

VaR estimates further underscore its instability, making it unsuitable for reliable VaR estimation.

In contrast, the unconditional GAN models showed more stable VaR estimates over time. These models' estimates were more consistent and bore closer resemblance to the traditional variance-covariance model, which is known for its simplicity and reliability. However, despite their improved stability, the unconditional GAN models also exhibited a higher-than-expected proportion of exceedances during backtesting. When subjected to Kupiec's POF test, all GAN models were statistically rejected, meaning their performance did not meet the desired accuracy standards.

The only model that was not rejected by Kupiec's POF test was the variance-covariance model. This model's failure proportion was closest to the expected 5%, and its p-value in the POF test indicated that it could not be statistically distinguished from the expected failure rate. This result highlights the variance-covariance model's robustness and reliability in estimating VaR, even when compared to more complex and computationally intensive GAN models.

5.3 GANs in the Broader Context of Machine Learning for VaR

Strengths of GANs: GANs have the unique advantage of being able to generate synthetic data that closely resembles the distribution of actual returns. This capability can be especially beneficial in financial markets where data is often limited, and tail events are rare. GANs also offer flexibility in modeling complex dependencies in the data, which can lead to more accurate risk predictions in volatile or non-linear market conditions.

Challenges Compared to Other ML Methods: However, GANs also present significant challenges compared to other ML methods like SVMs, random forests, and XGBoost. The difficulty in training GANs, their tendency to overfit, and the instability in predictions are key drawbacks. Unlike ensemble methods like random forests or XGBoost, which offer robust and stable predictions, GANs require careful tuning and validation to ensure reliable performance. Additionally, the interpretability of GANs is lower than that of tree-based models or even simpler neural networks, making it harder to understand the factors driving risk estimates.

5.4 Ethical Considerations

The use of AI models like Generative Adversarial Networks (GANs) in financial risk management, particularly for Value at Risk (VaR) estimation, introduces critical ethical considerations. One major concern is the potential for bias and unfairness, as GANs trained on historical financial data may inherit biases from this data, leading to skewed risk assessments. This can result in unfair outcomes, particularly for vulnerable stakeholders like retail investors and smaller financial institutions. Additionally, the opaque nature of GANs makes them difficult to interpret, raising transparency and accountability issues. Without clear understanding and oversight, the use of such complex models could erode trust in financial institutions and their risk management practices.

Furthermore, the reliability of GAN-based models must be rigorously examined, as their tendency to overfit recent data can lead to unstable and inaccurate VaR estimates, threatening financial stability. Financial institutions must ensure that these models are robust, regularly validated, and comply with regulatory standards to mitigate risks. Ethical responsibility in deploying GANs involves maintaining a commitment to fairness, transparency, and accountability while ensuring that AI-driven risk management tools do not exacerbate systemic risks or contribute to financial instability during periods of market stress.

Chapter 6

Conclusion

The exploration of Generative Adversarial Networks (GANs) for estimating Value at Risk (VaR) represents an innovative approach in financial risk management. GANs, which have been highly successful in various fields such as image generation and natural language processing, are being investigated for their potential to model complex financial data distributions. This study delves into the use of both conditional and unconditional GAN models to estimate VaR, comparing their performance against a traditional variance-covariance model. The findings reveal significant insights into the strengths and limitations of these advanced machine learning techniques when applied to financial risk management.

limitations of the study and its effects on results.

This study's findings are limited by several factors that affect the generalizability of the results. The dataset used, consisting of 2164 observations, may not represent other time periods, markets, or asset classes, making it difficult to apply these results universally. Additionally, the GAN models, particularly the conditional GAN, exhibited signs of overfitting, indicating that they may not perform well on different or unseen data. The assumption of data stationarity further limits the models' applicability, as financial markets often exhibit non-stationary behaviour. Simplified evaluation metrics, like Kupiec's POF test, provide only a narrow view of model performance, potentially overlooking other critical aspects such as the magnitude and timing of VaR exceedances.

The complexity and instability of training GANs, especially with financial data characterized by non-stationarity and fat tails, also pose significant challenges. These issues suggest that the results might not be easily replicable without substantial adjustments in different settings. The study's reliance on the traditional variance-covariance model as a benchmark, while useful, may not fully capture the GAN models' capabilities compared to more sophisticated risk management approaches. To improve the generalizability of GANs in financial risk management, future research should focus on expanding datasets, refining evaluation methods, and developing architectures better suited to the unique characteristics of financial data.

Challenges with GANs in Financial Applications

One of the major challenges identified with the use of GANs, particularly the unconditional models, is their difficulty in capturing current market conditions. Since these models are trained on historical data, they may not adequately reflect recent market shifts that are critical for accurate VaR estimation. The inability to adjust swiftly to changing market dynamics limits the effectiveness of GANs in real-time risk management applications, where recent data trends are crucial.

Another significant challenge with GANs is the complexity of training. GANs are known to be difficult to train, often requiring fine-tuning of hyperparameters and careful monitoring to avoid issues like mode collapse, where the generator produces limited variability in outputs, or instability in training. These challenges are particularly pronounced in financial applications, where the data is often non-stationary and characterized by fat tails and volatility clustering features that are difficult to model accurately.

To address some of these training challenges, this study implemented the Wasserstein GAN with Gradient Penalty (WGAN-GP), an improved GAN architecture that stabilizes training and enhances the generator's ability to produce realistic data. WGAN-GP introduces a modified loss function that mitigates some of the instability issues inherent in traditional GANs, making it more suitable for the complex nature of financial data. However, even with these improvements, the results suggest that GANs still have limitations when applied to VaR estimation.

Comparison with Traditional Models

The traditional variance-covariance model, despite its simplicity, demonstrated better performance in backtesting compared to the GAN models. This model assumes that asset returns are normally distributed and estimates VaR based on the mean and variance of returns. While the assumption of normality may not hold perfectly in real-world financial markets, the model's ability to produce stable and reliable VaR estimates makes it a preferred choice in many risk management scenarios.

The superior performance of the variance-covariance model in this study suggests that simplicity and robustness often outweigh the benefits of more complex models, especially when the latter are prone to instability and require extensive computational resources. The variance-covariance model's success can be attributed to its well-established theoretical foundations and its ability to provide a straightforward and interpretable framework for risk estimation.

Application of Findings in Real-World Financial Risk Management

The findings from this study provide valuable insights that could be directly applied to real-world financial risk management scenarios, particularly in the estimation and management of Value at Risk (VaR). The superior performance of the traditional variance-covariance model, which was not rejected by Kupiec's POF test, suggests that it remains a reliable and robust tool for estimating VaR. In practice, this model's simplicity and stability make it an attractive option for financial institutions that require consistent and interpretable risk metrics. Its assumption of normally distributed returns, while somewhat simplistic, often provides a reasonable approximation that is easy to implement and understand, making it suitable for use in daily risk reporting and regulatory compliance.

On the other hand, the study's findings also highlight the potential and challenges of using GANs (Generative Adversarial Networks) in financial risk management. The ability of GANs to model complex distributions could, in theory, provide more accurate risk estimates by capturing non-linear dependencies and tail risks that traditional models might miss. However, the practical application of GANs is currently limited by their sensitivity to market conditions and the difficulty in training them effectively. In real-world scenarios, this means that while GANs could be valuable for stress testing and scenario analysis—where understanding extreme market behaviours is crucial—their use in day-to-day VaR estimation might require further refinement. Financial institutions could explore hybrid approaches that combine GANs with traditional models to enhance their risk management frameworks, potentially using GANs to generate synthetic data that complements traditional methods, improving their ability to manage market shifts and rare events.

Guide on how financial institutions might implement your GAN-based VaR estimation in their risk management processes

Implementing GAN-based Value at Risk (VaR) estimation in a financial institution's risk management framework requires a thorough and systematic approach. The process begins with an assessment of the current risk management framework, where institutions should review their existing VaR models, such as variance-covariance, historical simulation, or Monte Carlo simulation. By understanding the strengths and limitations of these models, financial institutions can identify areas where GAN-based approaches may offer significant value. Additionally, it's crucial to identify and evaluate current data sources, including historical price data, market indicators, and macroeconomic variables. Ensuring that this data is comprehensive and clean is essential, as GANs rely on large, high-quality datasets for effective training.

The next step involves data preparation, which starts with gathering and preprocessing historical financial data to remove anomalies, missing values, and inconsistencies. This clean, well-structured data serves as the input for training the GAN models. For conditional GANs, feature engineering becomes vital, as it involves developing relevant features that represent market conditions—such as volatility indices, interest rates, or sector-specific indicators—allowing the model to adapt to changing market environments effectively.

When it comes to model selection and customization, institutions must choose the appropriate GAN architecture based on their specific needs. Conditional GANs are ideal for adapting to recent market trends, while unconditional GANs may be better suited for capturing broader historical patterns. The selected GAN model can be further tailored by incorporating elements like Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks, which help capture temporal dependencies in financial data. Regularization techniques should also be considered to prevent overfitting, ensuring the model's robustness.

Model training requires a robust computing environment equipped with adequate GPU/TPU resources to handle the computational demands of GANs. Once the environment is set up, the GAN model is trained on historical data, with careful monitoring of key metrics such as loss functions, mode collapse, and gradient penalties to ensure stable training. Fine-tuning the model involves adjusting hyperparameters like the learning rate, batch size, and the architecture of the generator and discriminator networks. After training, the model's performance is validated using a separate test dataset and backtested over historical periods to evaluate the accuracy of the GAN-based VaR estimates.

Evaluation of the model's performance includes applying Kupiec’s Proportion of Failures (POF) test to assess whether the actual exceedances align with the expected rate, such as 5% for a 95% VaR. Stress testing is also conducted by simulating extreme market conditions, which helps evaluate the model's robustness in predicting tail risks and extreme events. Additionally, a comparison with traditional models allows institutions to analyze the consistency, stability, and accuracy of GAN-based VaR estimates, particularly in capturing extreme events.

Integrating the GAN-based VaR model into the risk management workflow involves setting up reporting systems that ensure the model's output is easily interpretable by risk managers and other stakeholders. This integration should include clear visualizations of VaR estimates and potential exceedances. An implementation plan with timelines, responsibilities, and milestones is essential for successful integration, along with procedures for regular model updates and retraining as new data becomes available.

Monitoring and maintaining the model is crucial for its continued effectiveness. Continuous monitoring ensures that the model adapts to changing market conditions, with alerts set up for significant deviations from expected VaR levels. Periodic reassessment of the model may involve retraining with new data, adjusting hyperparameters, or incorporating new features to reflect evolving market conditions. Institutions might also consider using GANs as part of a hybrid risk management approach, combining GAN-based VaR estimates with those from traditional models to leverage the strengths of both.

Regulatory compliance is another key consideration. Financial institutions must ensure that their GAN-based VaR model aligns with all relevant regulatory requirements, which includes maintaining transparency, thorough documentation, and auditability. Engaging with regulators and providing detailed explanations of how the model works, how it improves upon traditional methods, and how it will be integrated into risk management practices is critical for regulatory approval.

Training and education of risk management teams on how to interpret GAN-based VaR estimates, understand the model's limitations, and use its outputs in decision-making processes is vital for successful adoption. Developing comprehensive user manuals and documentation supports the ongoing use and maintenance of the GAN model within the institution, covering everything from model operation to troubleshooting common issues.

Finally, future enhancements to the GAN-based VaR model should include establishing a feedback loop where risk managers can provide input on the model's performance, leading to continuous refinement and adaptation to changing market conditions. Institutions should also stay updated on advancements in GAN technology and explore incorporating new techniques, such as progressive growing or transfer learning, to further enhance model performance. By following this comprehensive approach, financial institutions can effectively integrate GAN-based VaR estimation into their risk management processes, enhancing their ability to manage and understand financial risks using advanced machine learning models.

Future Directions for GANs in Financial Risk Management

Despite the challenges and limitations observed in this study, GANs represent a promising area of research for financial risk management. The innovative approach of using GANs to model complex distributions of returns has the potential to capture non-linear dependencies and tail risks that traditional models may overlook. However, further research and development are needed to improve the stability and accuracy of GANs for VaR estimation.

One area for future exploration is the development of new GAN architectures specifically tailored for financial data. For example, incorporating recurrent neural networks (RNNs) or Long Short-Term Memory (LSTM) networks into the GAN framework could help capture temporal dependencies and improve the model’s ability to reflect current market conditions. Additionally, exploring different architectures for the generator and discriminator could lead to more realistic and stable VaR estimates.

Another avenue for future research is the integration of GANs with other machine learning models or traditional statistical methods. Hybrid models that combine the strengths of GANs in generating realistic data with the interpretability and reliability of traditional models could offer a more balanced approach to VaR estimation. For example, GANs could be used to generate synthetic data that augments traditional models, helping to improve their performance in scenarios where historical data is limited or not fully representative of current market conditions.

Finally, advancements in the training and evaluation of GANs are critical for their successful application in finance. Techniques such as progressive growing of GANs, which starts with simpler models and gradually increases complexity, or the use of transfer learning to leverage pre-trained models on similar datasets, could enhance the stability and performance of GANs in financial applications. Additionally, developing more sophisticated evaluation metrics that account for the unique characteristics of financial data, such as fat tails and extreme events, could help in better assessing the effectiveness of GANs for VaR estimation.

Suggestions for future Research

For future research, one promising direction would be to explore the integration of Generative Adversarial Networks (GANs) with other advanced neural network architectures like Long Short-Term Memory (LSTM) networks or Transformers. These hybrid models could leverage the strengths of both GANs and these architectures to improve the accuracy and robustness of Value at Risk (VaR) estimations.

Enhancing Temporal Dynamics with LSTMs

LSTMs are particularly effective in capturing long-term dependencies in sequential data, such as financial time series, where past market behavior influences future risks. By combining LSTMs with GANs, future research could develop models that not only generate realistic return distributions (through GANs) but also account for complex temporal patterns in market data (through LSTMs). This hybrid approach could be especially beneficial in scenarios where the market exhibits significant autocorrelation or where recent trends strongly influence future risks. For example, the LSTM component could model the sequence of market returns, while the GAN component could ensure that the generated scenarios align with the overall distribution of returns observed historically.

Leveraging Transformers for Improved Feature Attention

Transformers have revolutionized natural language processing by enabling models to focus on different parts of the input data through attention mechanisms. This capability could be adapted to financial data, where certain features (like macroeconomic indicators or specific market sectors) might have varying levels of importance at different times. Integrating Transformers with GANs could allow future models to dynamically adjust their focus based on current market conditions, potentially leading to more accurate and context-aware VaR estimates. The attention mechanisms in Transformers could help the GANs generate more realistic and contextually relevant market scenarios, improving the model’s ability to capture the nuanced relationships between different market factors.

Investigating Multi-Scale Temporal Modeling

Another area for future research could involve the application of multi-scale modeling techniques, where LSTMs or Transformers handle different temporal scales (short-term, medium-term, and long-term) within the same GAN framework. This approach could capture both immediate market reactions and longer-term trends, providing a more comprehensive view of potential risks. Such models could be particularly useful in volatile markets, where short-term shocks and long-term trends might simultaneously impact financial stability.

In conclusion, combining GANs with LSTMs or Transformers could significantly enhance the ability of risk models to capture complex temporal and feature dependencies, leading to more accurate and reliable VaR estimations. This hybrid approach represents a promising avenue for future research that could push the boundaries of financial risk modeling.

The use of GANs for estimating VaR presents both opportunities and challenges. While the conditional GAN model in this study was found to be unsuitable due to its sensitivity to recent data and instability, the unconditional GAN models showed more promise, though they still fell short of the performance of the traditional variance-covariance model. The variance-covariance model's simplicity, robustness, and consistent performance in backtesting suggest that it remains a reliable choice for VaR estimation.

However, GANs are a relatively new tool in the field of finance, and their potential is not yet fully realized. The complexity of training GANs and their sensitivity to market conditions are significant hurdles that need to be addressed. With ongoing advancements in GAN architecture and training methods, there is potential for these models to become more effective in capturing the complexities of financial markets and providing accurate risk estimates.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C. and Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*.
2. Acharya, D., Huang, Z., Paudel, D. and Luc Van Gool (2018). *Towards High Resolution Video Generation with Progressive Growing of Sliced Wasserstein GANs*. [online] Available at: http://arxiv.org/pdf/1810.02419v2 [Accessed 17 Aug. 2024].
3. Arjovsky, M., Chintala, S. and Bottou, L. (2017). *Wasserstein GAN*. [online] Available at: http://arxiv.org/pdf/1701.07875v3 [Accessed 17 Aug. 2024].
4. Artzner, P. (1999). Coherent measures of risk. In: *Mathematical Finance*. pp.203–228.
5. Boodhun, N. and Jayabalan, M. (2018). Risk prediction in life insurance industry using supervised learning algorithms. *Complex & Intelligent Systems*, 4(2), pp.145–154. doi:https://doi.org/10.1007/s40747-018-0072-1.
6. Brock, A., Donahue, J., Deepmind and Simonyan, K. (2019). *LARGE SCALE GAN TRAINING FOR HIGH FIDELITY NATURAL IMAGE SYNTHESIS*. [online] Available at: http://arxiv.org/pdf/1809.11096v2 [Accessed 17 Aug. 2024].
7. Calderon, T.G. and Cheh, J.J. (2002). A roadmap for future neural networks research in auditing and risk assessment. *International Journal of Accounting Information Systems*, 3(4), pp.203–236. doi:https://doi.org/10.1016/s1467-0895(02)00068-4.
8. Chan, S. and Nadarajah, S. (2018). *Extreme Values and Financial Risk*. Hoboken, New Jersey: John Wiley & Sons, pp.283–356.
9. Chaudhuri, A. and De, K. (2011). Fuzzy Support Vector Machine for bankruptcy prediction. *Applied Soft Computing*, 11(2), pp.2472–2486. doi:https://doi.org/10.1016/j.asoc.2010.10.003.
10. Chollet, F. (2015). *Keras*. [online] Keras. Available at: https://keras.io..
11. Cimpoeru, S.S. (2011). NEURAL NETWORKS AND THEIR APPLICATION IN CREDIT RISK ASSESSMENT. EVIDENCE FROM THE ROMANIAN MARKET / NEURONINIAI TINKLAI IR JŲ TAIKYMAS KREDITO RIZIKAI VERTINTI RUMUNIJOS RINKOS PAVYZDŽIU. *Technological and Economic Development of Economy*, 17(3), pp.519–534. doi:https://doi.org/10.3846/20294913.2011.606339.
12. Coakley, J.R. and Brown, C.E. (2000). Artificial neural networks in accounting and finance: modeling issues. *International Journal of Intelligent Systems in Accounting, Finance & Management*, 9(2), pp.119–144. doi:https://doi.org/10.1002/1099-1174(200006)9:2%3C119::aid-isaf182%3E3.0.co;2-y.
13. D, M., M., Dr.S. and P., Dr.S. (2021). Early Action Prediction Using 3DCNN With LSTM and Bidirectional LSTM. *SSRN Electronic Journal*. doi:https://doi.org/10.2139/ssrn.3815963.
14. Danielsson, J., Hartmann, P. and Vries, C. (1998). The cost of conservatism. *Risk*, 11(1), pp.101–103.

15. Engle, R.F. (1982). Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 50(4), pp.987–1007. doi:https://doi.org/10.2307/1912773.
16. FasterCapital. (n.d.). *The biggest online incubator and accelerator. Provides work per equity and helps in raising capital from a large network of angel investors and VCs*. [online] Available at: https://fastercapital.com [Accessed 4 Aug. 2024].
17. finance, Y. (2024). [online] Yahoo Finance. Available at: https://finance.yahoo.com.
18. Friedman, J., Hastie, T. and Tibshirani, R. (2001). *The elements of statistical learning*. 2nd ed. Springer Series in Statistics.
19. Fu, R., Chen, J., Zeng, S., Zhuang, Y. and Sudjianto, A. (2019). *Time series simulation by conditional generative adversarial net*.
20. Germán García-Jara, Protopapas, P. and Estévez, P.A. (2022). Improving Astronomical Time-series Classification via Data Augmentation with Generative Adversarial Networks. *Astrophysical journal/The Astrophysical journal*, 935(1), pp.23–23. doi:https://doi.org/10.3847/1538-4357/ac6f5a.
21. Gong, X., Wang, X. and Li, N. (2022). Research on DUAL-ADGAN Model for Anomaly Detection Method in Time-Series Data. *Computational intelligence and neuroscience (Print)*, 2022, pp.1–18. doi:https://doi.org/10.1155/2022/8753323.
22. Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep learning*. MIT Press.
23. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014). *Generative Adversarial Nets*. [online] Available at: http://arxiv.org/pdf/1406.2661v1 [Accessed 17 Aug. 2024].
24. Grace, A.M. and Williams, S.O. (2016). Comparative Analysis of Neural Network and Fuzzy Logic Techniques in Credit Risk Evaluation. *International Journal of Intelligent Information Technologies*, 12(1), pp.47–62. doi:https://doi.org/10.4018/ijiit.2016010103.
25. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. and Courville, A.C. (2017). Improved training of wasserstein gans. In: *Advances in neural information processing systems*. pp.5767–5777.
26. Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), pp.1735–1780.
27. Hong, K. (2016). Analytical method of computing stressed value-at-risk with conditional value-at-risk. *The Journal of Risk*. doi:https://doi.org/10.21314/jor.2016.354.
28. Hornik, K., Stinchcombe, M. and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, [online] 2(5), pp.359–366. doi:https://doi.org/10.1016/0893-6080(89)90020-8.
29. Huang, L., Li, L., Wei, X. and Zhang, D. (2022). Short-term prediction of wind power based on BiLSTM–CNN–WGAN-GP. *Soft Computing*, 26(20), pp.10607–10621. doi:https://doi.org/10.1007/s00500-021-06725-x.

30. Hull, J. (2015). *Risk management and financial institutions : Wiley Finance*. Hoboken, New Jersey: John Wiley & Sons.
31. Hull, J. and White, A. (1998). Incorporating volatility updating into the historical simulation method for value-at-risk. *Journal of risk*, 1(1), pp.5–19.
32. Ioffe, S. and Szegedy, C. (2015). *Batch normalization: Accelerating deep network training by reducing internal covariate shift*.
33. Jin, Q., Lin, R. and Yang, F. (2020). E-WACGAN: Enhanced Generative Model of Signaling Data Based on WGAN-GP and ACGAN. *IEEE Systems Journal*, 14(3), pp.3289–3300. doi:https://doi.org/10.1109/jsyst.2019.2935457.
34. Jorion, P. (2006). *Value at risk: The new benchmark for managing financial risk*. 3rd ed. McGraw-Hill.
35. Kakade, K., Jain, I. and Mishra, A.K. (2022). Value-at-Risk forecasting: A hybrid ensemble learning GARCH-LSTM based approach. *Resources Policy*, 78, p.102903. doi:https://doi.org/10.1016/j.resourpol.2022.102903.
36. Khan, S. (2019). *Calculate value-at-risk using wasserstein generative adversarial networks (wgan-gp) for risk management system*. [online] Available at: https://chatbotslife.com/calculate-value-at-risk-using-wasserstein [Accessed 18 Aug. 2024].
37. Kim, H., Cho, H. and Ryu, D. (2020). Corporate Default Predictions Using Machine Learning: Literature Review. *Sustainability*, 12(16), p.6325. doi:https://doi.org/10.3390/su12166325.
38. Kim, H.Y. and Won, C.H. (2018). Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models. *Expert Systems with Applications*, [online] 103, pp.25–37. doi:https://doi.org/10.1016/j.eswa.2018.03.002.
39. Kingma, D.P. and Ba, J. (2014). *Adam: A method for stochastic optimization*.
40. Lanzarini, L.C., Villa Monte, A., Bariviera, A.F. and Jimbo Santana, P. (2017). Simplifying credit scoring rules using LVQ + PSO. *Kybernetes*, 46(1), pp.8–16. doi:https://doi.org/10.1108/k-06-2016-0158.
41. Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A. and Jurafsky, D. (2017). *Adversarial Learning for Neural Dialogue Generation*. [online] Available at: http://arxiv.org/pdf/1701.06547v5 [Accessed 17 Aug. 2024].
42. Li, J., Wang, X., Lin, Y., Sinha, A. and Wellman, M.P. (2018). *Generating Realistic Stock Market Order Streams*. [online] OpenReview. Available at: https://openreview.net/forum?id=rke41hC5Km [Accessed 17 Aug. 2024].
43. Li, J.-P., Mirza, N., Rahat, B. and Xiong, D. (2020). Machine learning and credit ratings prediction in the age of fourth industrial revolution. *Technological Forecasting and Social Change*, 161, p.120309. doi:https://doi.org/10.1016/j.techfore.2020.120309.
44. Linsmeier, T.J. and Pearson, N.D. (1996). *Risk measurement: An introduction to value at risk*. University of Illinois at Urban-Champaign.

45. Longin, F.M. (2000). From value at risk to stress testing: The extreme value approach. *Journal of Banking & Finance*, 24(7), pp.1097–1130.
46. Mariani, G., Zhu, Y., Li, J., Scheidegger, F., Istrate, R., Bekas, C. and Malossi, A.C.I. (2019). *Pagan: Portfolio analysis with generative adversarial networks*.
47. Marimoutou, V., Raggad, B. and Trabelsi, A. (2009). Extreme value theory and value at risk: application to oil market. *Energy Economics*, 31(4), pp.519–530.
48. Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, 7(1), pp.77–91.
49. MathWorks (1984). *Overview of var backtesting*. [online] MathWorks. Available at: https://se.mathworks.com/help/risk/overview-of-var-backtesting.html [Accessed 18 Aug. 2024].
50. Mcneil, A.J., FreyR. and Embrechts, P. (2015). *Quantitative risk management : concepts, techniques and tools*. Princeton: Princeton University Press.
51. Nain, A.K. (2020). *Wgan-gp overriding model.train step*. [online] Keras. Available at: https://keras.io/.
52. Omar, N., Johari, Z. 'Amirah and Smith, M. (2017). Predicting fraudulent financial reporting using artificial neural network. *Journal of Financial Crime*, 24(2), pp.362–387. doi:https://doi.org/10.1108/jfc-11-2015-0061.
53. Oreski, S., Oreski, D. and Oreski, G. (2012). Hybrid system with genetic algorithm and artificial neural networks and its application to retail credit risk assessment. *Expert Systems with Applications*, 39(16), pp.12605–12617. doi:https://doi.org/10.1016/j.eswa.2012.05.023.
54. Philippe Jorion (2001). *Value at risk : the benchmark for controlling market risk*. New York; London: Mcgraw-Hill.
55. Punniyamoorthy, M. and Sridevi, P. (2016). Identification of a standard AI based technique for credit risk analysis. *Benchmarking: An International Journal*, 23(5), pp.1381–1390. doi:https://doi.org/10.1108/bij-09-2014-0094.
56. Raghavendra, B.K. and Simha, J.B. (2010). Evaluation of Feature Selection Methods for Predictive Modeling Using Neural Networks in Credits Scoring. 2(3), pp.714–718.
57. Saita, F. (2007). *Value at risk and bank capital management*. 1st ed. Elsevier.
58. Shah, H. (2018). *Using Bidirectional Generative Adversarial Networks to estimate Value-at-Risk for Market Risk Management*. [online] Medium. Available at: https://towardsdatascience.com/using-bidirectional-generative-adversarial-networks-to-estimatevalue-at-risk-for-market-risk-c3dffbbde8dd [Accessed 17 Aug. 2024].
59. Shirazi, A. (2014). Value at Risk (VaR) Backtesting Techniques and P-Value Risk Decomposition Analysis. *SSRN Electronic Journal*. doi:https://doi.org/10.2139/ssrn.2413702.
60. Zhang, K., Zhong, G., Dong, J., Wang, S. and Wang, Y. (2019). Stock Market Prediction Based on Generative Adversarial Network. *Procedia Computer Science*, 147, pp.400–406. doi:https://doi.org/10.1016/j.procs.2019.01.256.

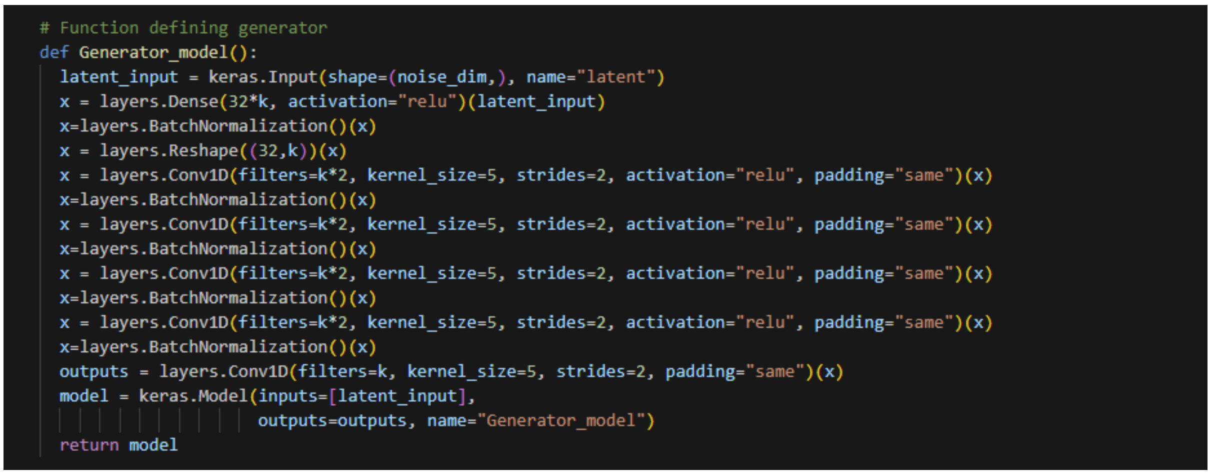
61. Zhou, X., Pan, Z., Hu, G., Tang, S. and Zhao, C. (2018). Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets. *Mathematical Problems in Engineering*, [online] 2018(1), pp.1–11. doi:https://doi.org/10.1155/2018/4907423.

Appendix

You can find the code associated with this study on this GitHub link, where you can go through the readme file to execute the code.

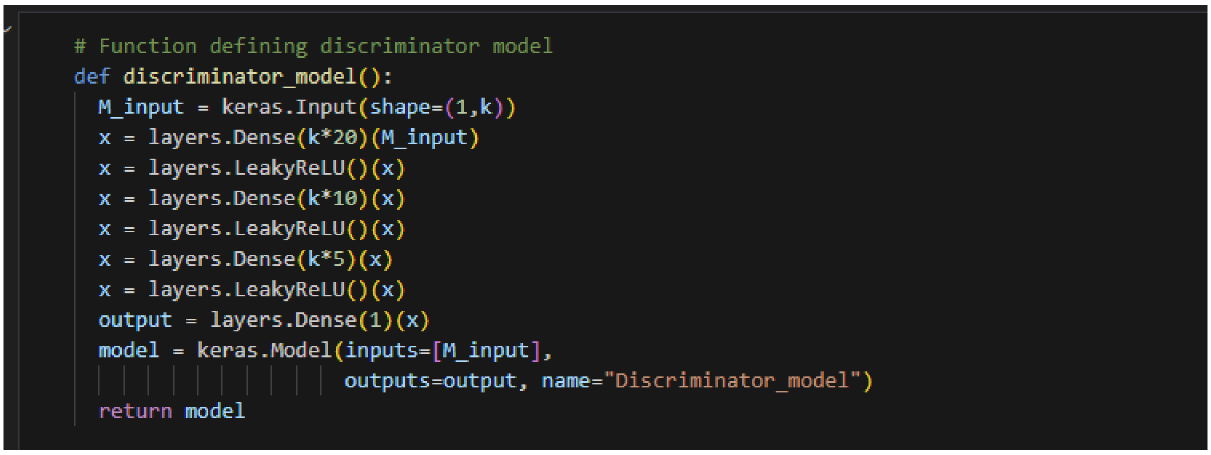
https://github.com/pulkitgupta31/WGAN-GP-for-value-at-risk-estimation

1. The Generator code



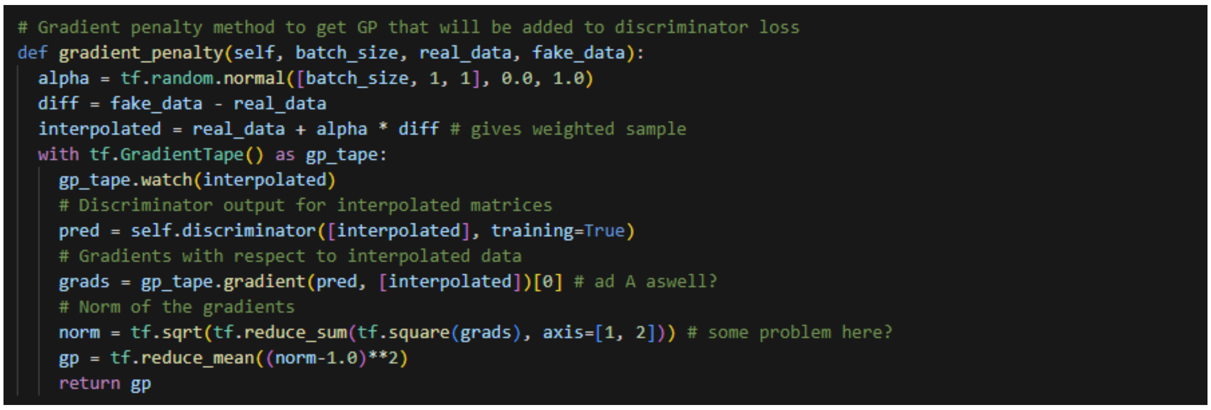
```python
# Function defining generator
def Generator_model():
    latent_input = keras.Input(shape=(noise_dim,), name="latent")
    x = layers.Dense(32*k, activation="relu")(latent_input)
    x=layers.BatchNormalization()(x)
    x = layers.Reshape((32,k))(x)
    x = layers.Conv1D(filters=k*2, kernel_size=5, strides=2, activation="relu", padding="same")(x)
    x=layers.BatchNormalization()(x)
    x = layers.Conv1D(filters=k*2, kernel_size=5, strides=2, activation="relu", padding="same")(x)
    x=layers.BatchNormalization()(x)
    x = layers.Conv1D(filters=k*2, kernel_size=5, strides=2, activation="relu", padding="same")(x)
    x=layers.BatchNormalization()(x)
    x = layers.Conv1D(filters=k*2, kernel_size=5, strides=2, activation="relu", padding="same")(x)
    x=layers.BatchNormalization()(x)
    outputs = layers.Conv1D(filters=k, kernel_size=5, strides=2, padding="same")(x)
    model = keras.Model(inputs=[latent_input],
                        outputs=outputs, name="Generator_model")
    return model
```

2. The Discriminator code



```python
# Function defining discriminator model
def discriminator_model():
    M_input = keras.Input(shape=(1,k))
    x = layers.Dense(k*20)(M_input)
    x = layers.LeakyReLU()(x)
    x = layers.Dense(k*10)(x)
    x = layers.LeakyReLU()(x)
    x = layers.Dense(k*5)(x)
    x = layers.LeakyReLU()(x)
    output = layers.Dense(1)(x)
    model = keras.Model(inputs=[M_input],
                        outputs=output, name="Discriminator_model")
    return model
```

3. The Penalty Term code



```python
# Gradient penalty method to get GP that will be added to discriminator loss
def gradient_penalty(self, batch_size, real_data, fake_data):
    alpha = tf.random.normal([batch_size, 1, 1], 0.0, 1.0)
    diff = fake_data - real_data
    interpolated = real_data + alpha * diff # gives weighted sample
    with tf.GradientTape() as gp_tape:
        gp_tape.watch(interpolated)
        # Discriminator output for interpolated matrices
        pred = self.discriminator([interpolated], training=True)
        # Gradients with respect to interpolated data
        grads = gp_tape.gradient(pred, [interpolated])[0] # ad A aswell?
        # Norm of the gradients
        norm = tf.sqrt(tf.reduce_sum(tf.square(grads), axis=[1, 2])) # some problem here?
        gp = tf.reduce_mean((norm-1.0)**2)
        return gp
```