Multi-Agent Deep Reinforcement Learning and GAN-Based Market Simulation for Derivatives Pricing and Dynamic Hedging

By

Samson Qian

B.S. Data Science
University of California San Diego, 2021

Submitted to the MIT Sloan School of Management in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF FINANCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2023

Authored by: Samson Qian
MIT Sloan School of Management
January 20, 2022

Certified by: Leonid Kogan
Nippon Telegraph and Telephone Professor of Management
Thesis Supervisor

Accepted by: Urmi Samadar
Assistant Dean, MIT Sloan Master of Finance Program
MIT Sloan School of Management

Multi-Agent Deep Reinforcement Learning and GAN-Based Market Simulation for Derivatives Pricing and Dynamic Hedging

by

Samson Qian

Submitted to the MIT Sloan School of Management
on January 20, 2023, in partial fulfillment of the
requirements for the degree of
Master of Finance

Abstract

Advancements in computing capabilities have enabled machine learning algorithms to learn directly from large amounts of data. Deep reinforcement learning is a particularly powerful method that uses agents to learn by interacting with an environment of data. Although many traders and investment managers rely on traditional statistical and stochastic methods to price assets and develop trading and hedging strategies, deep reinforcement learning has proven to be an effective method to learn optimal policies for pricing and hedging. Machine learning removes the need for various parametric assumptions about underlying market dynamics by learning directly from data. This research examines the use of machine learning methods to develop a data-driven method of derivatives pricing and dynamic hedging. Nevertheless, machine learning methods like reinforcement learning require an abundance of data to learn. We explore the implementation of a generative adversarial network-based approach to generate realistic market data from past historical data. This data is used to train the reinforcement learning framework and evaluate its robustness. The results demonstrate the efficacy of deep reinforcement learning methods to price derivatives and hedge positions in the proposed systematic GAN-based market simulation framework.

Thesis Supervisor: Leonid Kogan
Title: Nippon Telegraph and Telephone Professor of Management

Acknowledgments

I am honored and grateful to have Professor Leonid Kogan as my research advisor. Thank you, Professor Kogan, for all the guidance throughout the research process and the great advice on the direction and focus of the study. Thank you for advising both the theoretical and practical applications of my work.

I would like to thank my parents and friends for supporting me and fostering my intellectual curiosity and passion to complete my Master of Finance degree and to try to understand more about the complex financial markets. Thank you for always pushing and motivating me to achieve more.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background of Derivatives Pricing Methods

Financial derivatives pricing has been a prominent and important field in modern finance as investment managers seek to understand the factors that impact derivative contract prices and devise trading strategies on those insights. Derivatives have also been used as a tool for hedging to reduce risk in an investor's positions over time. Options, one type of financial derivative, are one of the most popular instruments for investors, trading an average daily of 39 million contracts. Previous work done on options pricing rely heavily on assumptions about the market and underlying stock movement to solve for partial differential equations and derive pricing models. For example, many models rely on the assumption that stock prices move according to a geometric Brownian motion stochastic process. The most popular framework for pricing options is the Black-Scholes Merton (BSM) model [1], derived based on those market assumptions to measure the value of derivatives based on factors including the underlying asset volatility, strike price, interest rate, and expiration time. Another method to price options is to run Monte Carlo Simulations on stock price movement, assuming stock price follows geometric Brownian motion paths, and derive option value based on the results of the simulations. These assumptions about the market and underlying, however, do not always hold in the empirical world, and stock movement patterns are heavily influenced by many factors apart from randomness.

1.2 Research Motivation

1.2.1 Market Assumptions

Despite the elegance of the modern theoretically-sound options pricing models such as Black-Scholes, these models depend on many market condition assumptions that are unrealistic in the real world. For example, in the real world, markets are incomplete and also have many frictions. These market frictions can have significant impacts on option prices and interfere with many trading and hedging strategies. For example, transaction costs resulting from market bid-ask spreads are an important factor to consider, but not accounted for by the Black-Scholes model. Furthermore, interest rates and stock prices are affected by a large number of factors that these stochastic pricing models do not account for. These factors reside within the empirical market data as seen by actual stock price movements and market orders, influenced by trading transactions between various participants in the market. In reality, the market microstructure is very complex and is affected by a large number of factors. As a result, the Black-Scholes model and other traditional options pricing methods do not yield exact option prices quoted on the market.

Nevertheless, the model serves as a good framework for analyzing which factors come into play for option price movement and can be a useful baseline method. Attributes of an option including strike price and maturity as well as the underlying spot price, volatility, and interest rates are all important factors contributing to the option's price. The Black-Scholes model is given by the following partial differential equation and the resulting closed-form solution for the price of a call option with strike price K, maturity t, underlying spot price S, underlying volatility σ, and risk-free rate r:

$$\begin{aligned}\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC = 0 \\ C = N(d)S - N(d - \sigma\sqrt{t})Ke^{-rt} \\ d = \frac{\ln\frac{S}{K} + (r + \frac{\sigma^2}{2})t}{\sigma\sqrt{t}}\end{aligned} \tag{1.1}$$

This study explores how to adapt these conventional options pricing frameworks to develop a model-free deep reinforcement learning-based options pricing and hedging algorithm based on empirical market data without relying on market assumptions.

1.2.2 Limitations of Historical Financial Data

Historical financial data is very useful, as it represents real data observed in the market, but incredibly scarce, especially for applications of machine learning methods. If there was an abundance of historical stock price data that is representative of diverse market conditions, it would be possible to build and train machine learning models that are more robust when deployed live. However, even with daily-frequency market data, assuming there are 252 trading days in a year translates to only 2,520 data points for a 10-year period. This presents a problem with only using historical data to build models. To effectively build models and test for an algorithm's generalizability, it is essential to have a large number of samples to train and evaluate a model on.

Backtesting Limitations

Often traders evaluate and test models and trading strategies by backtesting on historical market data. Although backtesting is a popular methodology used for testing trading strategy performances, it has many fundamental flaws that may bias the results and allow traders to believe the particular model or strategy will work once deployed, when in fact it may not.

Backtesting simply uses historical market data to evaluate a strategy's performance but does not account for the effect of the strategy's execution on the response from other market participants. This flaw is fundamental in the backtesting methodology because it uses the same data for evaluation, regardless of the trading strategy being evaluated. However, in real-world markets, the execution of different trading strategies, varying by factors such as order size, may have a significant impact on other market orders placed in response. Consequently, backtesting on historical data may bias the evaluation result and is not representative of how a trading algorithm will actually perform once deployed. Furthermore, the factors that affect the markets are constantly evolving and changing over time. Shifts in economic trends may make backtesting on historical data unfavorable because of the inflexibility of testing conditions. Alternatively, a framework that learns from historical market data and consistently generates new market data with similar properties is more favorable.

Generating Market Data

A traditional method for generating market data comes through the use of Monte Carlo simulations, in which a large number of data samples are generated through probabilistic simulations. These simulations are parameter-based and rely on the careful selection of parameters to produce accurate and representative data. However, not only are markets always dynamic and have constantly changing conditions, it is difficult to determine which parameters are most important to include in the simulation. Many derivatives pricing models rely on these assumptions about the distribution of the underlying asset price movement. For example, a popular framework for using Monte Carlo simulations to generate stock price paths assumes stock prices follow geometric Brownian motion processes and price levels are log-normally distributed. The Black-Scholes model assumes the underlying stock follows a geometric Brownian motion process (with expected return μ and volatility σ), defined by the following stochastic process:

$$\begin{aligned} \frac{dS}{S} &= \mu dt + \sigma dW \\ S_T &= S_0 * e^{(r-\frac{1}{2}\sigma^2)t+\sigma\sqrt{t}*N(0,1)} \end{aligned} \tag{1.2}$$

Using this process, Monte Carlo simulations generate a large number of stock price paths based on the specified parameters return volatility σ and some randomness. The advantage of this approach is the unbounded ability to generate large amounts of data representative of the vast amount of different possible paths that the stock can take. The disadvantage, however, comes with the parametric assumptions about underlying stock price movement which may not be completely market-realistic. In reality, stock price movements depend on a large number of factors and contain complex properties. Monte Carlo simulations are generally useful for simulating data with a known and constant probability distribution. However, in the case of financial data, market dynamics and conditions are constantly changing, and it may not be feasible to consistently know the true underlying probability distribution for stock returns.

To overcome some of the limitations of traditional methods like backtesting and Monte Carlo simulations, a possible solution is to create a generative market simulation system that can produce synthetic market data based on historical market data. Adversarial machine learning has the ability to generate fake data that closely resembles real data. In this study, we explore the use of generative adversarial networks (GANs) to learn the market microstructure and systematically generate realistic synthetic market data based on historical market data. This approach is an unsupervised, non-parametric approach as opposed to the use of Monte Carlo simulations. Ultimately, we adapt this framework to train and evaluate the efficacy of our proposed deep reinforcement learning-based pricing and hedging methods.

1.3 Objective

Previous work done on derivatives pricing rely heavily on assumptions about the market and underlying stock movement to be consistent with theoretical standards for solving partial differential equations and deriving pricing models. For example, many simulations rely on the assumption that stock prices move according to a geometric Brownian motion stochastic process. This assumption, however, does not always hold in the empirical world, and underlying price movement patterns are heavily influenced by many factors apart from randomness. The most common and famous framework to compare Monte Carlo pricing simulation results to is the Black-Scholes Merton (BSM) model. This framework is derived from a partial differential equation that also relies on market assumptions.

This study extends previous work and hopes to implement and apply deep reinforcement learning methods to price derivatives on assets in a diverse set of markets with more realistic conditions. This research aims to extend previous work by implementing deep reinforcement learning methods to price financial derivatives and perform dynamic hedging in a diverse set of markets with more realistic market conditions reflected through empirical data. The objective of the deep reinforcement learning approach is to perform better than traditional models on a variety of financial derivatives for multiple asset classes. Then, better hedging strategies can be implemented to achieve higher expected returns and lower volatilities. Multiple state-of-the-art reinforcement learning models will be compared against traditional baseline models for the financial derivatives considered. We explore the implementation of a GAN-based synthetic market to train and evaluate deep hedging agents.

1.4 Related Works

There have been previous works done using deep reinforcement learning for options pricing and hedging. Hans Buehler et al. (2019) researched and developed "Deep Hedging" [2] methods using deep reinforcement learning to devise optimal hedging strategies while accounting for market frictions and other factors that impact derivatives prices. In "Deep Hedging", the authors use a synthetic market based on the Heston Model [3] to generate data with a parametric approach. Furthermore, Igor Halperin implemented a "QLBS" (Q-Learning Black Scholes) [4] model to price and hedge options in a Black-Scholes world based on Q-learning methods. James Hutchinson et al. propose a non-parametric options pricing and hedging approach using traditional machine learning and learning network algorithms [5]. Gordon Ritter et al. propose a framework for using reinforcement learning to hedge derivatives that can be accurately priced [6]. This study extends previous works of deep hedging by training and evaluating the algorithm in a GAN-based market that generates realistic market data representative of and exhibiting similar properties to real market dynamics.

Chapter 2

Generative Adversarial Networks for Market Data Generation

2.1 Overview of Generative Adversarial Networks

2.1.1 Vanilla GAN (VGAN)

Generative adversarial networks (GANs) are one type of powerful generative AI that serve as the backbone of many applications in synthetic data generation and recent Deepfake technology. GANs were first introduced as an innovative unsupervised framework to generate synthetic data that represents real data by simultaneously training a generator and discriminator to learn the underlying distribution of the real data. Ian J. Goodfellow et al. (2014) demonstrated the efficacy of GANs in generating synthetic data that resemble the properties of real data [7]. The generator takes real input data x and adds Gaussian-distributed random noise z to output synthetic data, while the discriminator tries to distinguish the output from the generator and the real data. As the discriminator performs better in distinguishing real and synthetic data, the generator learns more about the underlying distribution and becomes better at generating new data. Vanilla GANs (VGAN) have been very successful in generating fake images and videos. During the training process, the discriminator acts as a classifier and outputs a probability that the generator-produced sample is fake. The standard machine learning process of using gradient descent is performed to continuously improve both the generator and discriminator simultaneously. The objective function of the generator G and discriminator D used to update the GAN's weights, computed at each training iteration, is given by the following value function V in which the generator tries to minimize and the discriminator tries to maximize.

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

In the context of generating market data, real historical time-series data of equity markets are used as input instead of images. The end goal is still the same, which is to produce representative synthetic data that exhibit similar properties as the real data. The structure of a vanilla GAN is displayed in Figure 2-1.

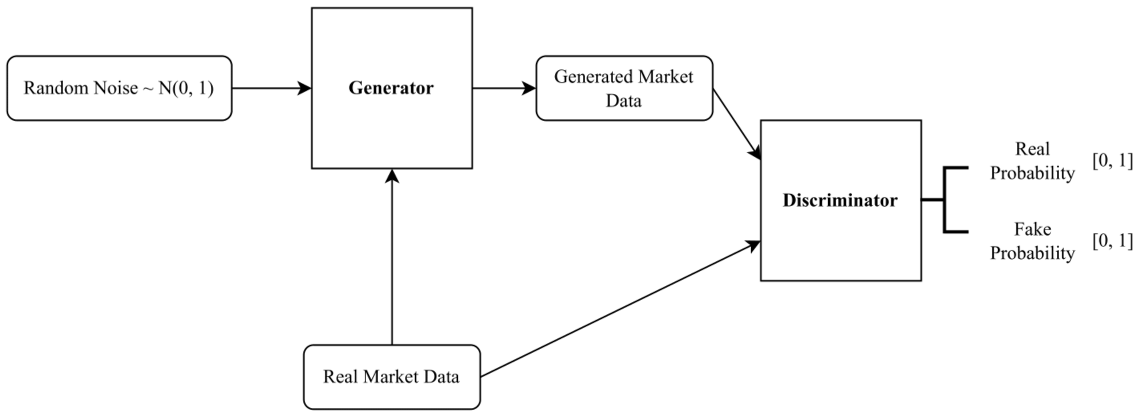

Figure 2-1: Structure of a Vanilla GAN

The generator and discriminator compete in a zero-sum game with the objective of training the generator to outperform the discriminator. At each training iteration, the loss is computed given the discriminator's performance and the errors are backpropagated through the generator and discriminator to adjust weights. As the discriminator becomes better at distinguishing real and fake data, the generator has a higher loss and is forced to adjust its weights to generate data that follow the distribution of real data more closely. At the end of the training process, the generator is able to produce new data that resembles the original data, since it has learned the distribution of the real data during the training process.

Applications of GANs

GANs have mostly been used for image processing tasks such as generating fake images or performing adversarial attacks on neural networks. However, the framework of GANs can be extended to work with other types of data as well, including tabular and time-series data. This has great implications for working with financial data, as the majority of data in finance consists of tabular and time-series data, such as asset price movement and limit order books. Similar to the GANs used in Deepfake technology, GANs also have the potential to learn the underlying distributions and data-generating processes of price evolution and order books. This allows GANs to learn the market microstructure and price evolution process directly from data instead of relying on various parametric assumptions about market dynamics.

2.1.2 Wasserstein GAN (WGAN)

Nevertheless, vanilla GANs often suffer from a common problem known as mode collapse, where the model hits a local minimum during the training process and is unable to continue learning from the data. Very often, this issue arises with the structure of the GAN due to the non-convexity of the discriminator's loss function during training. Mode collapse ultimately results in the generator producing the same data with the discriminator unable to learn to distinguish this fake. For example, if the generator learns to produce one possible path of simulated market data that is indistinguishable from real market data to the discriminator, then the generator will continue producing this single instance of market data. This can be caused by a variety of factors, with vanishing gradients in the discriminator as one possibility. In order to implement a useful market simulator, the generator component of the GAN must be versatile and learn to generate a large variety of synthetic data that is representative of empirical data. To achieve this versatility, it is essential that the discriminator can perform well during the training process to force the generator to learn to generate a diverse set of useful representations of market data.

Many alternative forms of GANs have been proposed to tackle this issue. One method to address the issue of mode collapse in vanilla GANs is to adjust the output of the discriminator so that it returns a score instead of a probability. Rather than being strictly a classifier of real and fake images, it is possible to turn the discriminator into a critic of generated data instead. This creates the benefit of allowing the discriminator output to exceed the bound of 0 to 1, which can alleviate the effects of vanishing gradients and mode collapse. Arjovsky, Chintala, and Bottou (2017) explore the use of Wasserstein GANs (WGAN) [8] to implement this idea by using the Wasserstein loss metric as the output of the discriminator. We will not delve into the derivation of gradient descent, but instead, focus on the implementation with financial data. With a set of 1-Lipschitz functions D_f, the Wasserstein GAN value function is defined by:

$$\min_{G} \max_{D} V_W(D, G) = E_{x \sim p_{data}(x)}[D_f(x)] + E_{z \sim p_z(z)}[D_f(G(z))] \tag{2.2}$$

Instead of making classifications like the discriminator of a VGAN does, the discriminator of a WGAN computes the Wasserstein distance and uses this loss to update weights. The other components of the model remain relatively similar. The structure of a Wasserstein GAN is displayed in Figure 2-2.

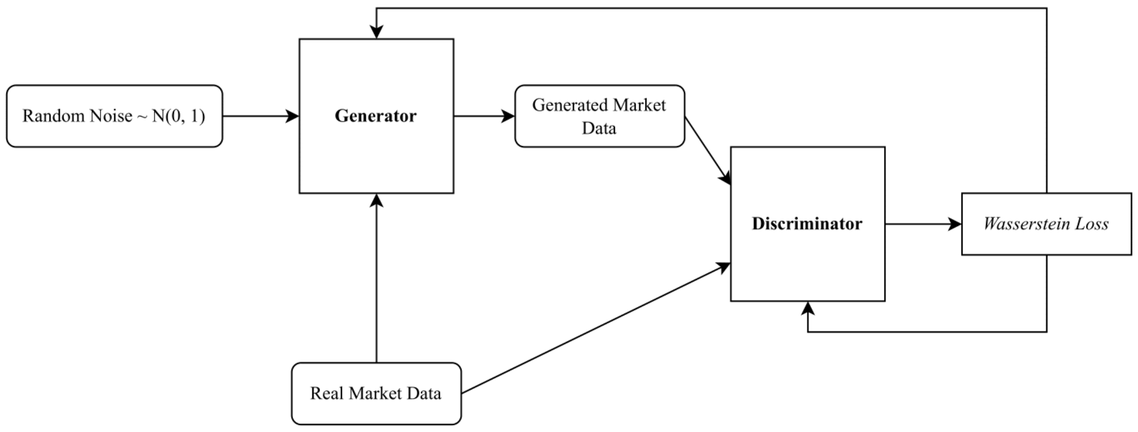

Figure 2-2: Structure of a Wasserstein GAN

For implementing a useful synthetic market environment, the generator must be robust in creating market data from a wide range of possible market conditions and possibilities. Gulrajani et al. (2017) propose an additional improvement to the original WGAN framework by introducing a critic gradient penalty [9] instead of the standard weight clipping approach in the training process, which will further help with mitigating mode collapse and promoting stability in data generation. The gradient penalty, combined with the use of Wasserstein distance, greatly improves the GAN's ability to learn robust representations of real market data. This model can now be trained using historical financial data, and the trained generator can be used to construct a representative synthetic market environment which may be useful for a variety of applications, particularly for machine learning methods.

2.1.3 Other Forms of GANs

Additional studies have been done in devising new methods to alter the GAN's structure to improve performance in generating better data. Many other forms of GANs have recently been developed, many of which serve a specific task in data generation. For example, many GANs have been tailored to work particularly on time-series data, which exhibit properties that differ from image data. As time-series data often has high complexities, some variations of GANs are implemented using convolutional and recurrent neural networks in the generators and discriminators to learn sequential data. Olof Mogren (2016) describes Continuous Recurrent Neural Network GANs (C-RNN-GAN) [10], a variation of GANs, to generate continuous sequential data. Nevertheless, the proposed framework, unlike WGANs, relies on the classification framework and may suffer from mode collapse issues when applied to financial data.

Time-Series GANs

One specific feature of time-series data that differs from other types of data is that time-series exhibit various temporal dynamics, which should be preserved when using a GAN to generate time-series data. However, the original framework of GANs does not provide any specific means of learning and maintaining temporal correlations on time series. A notable extension of the GAN framework to work with time-series data is the Time-Series GAN. Jinsung Yoon, Daniel Jarrett, and Mihaela van der

Schaar (2019) discuss solutions to this problem and propose a modification of GANs to generate more-realistic time-series data that maintain statistical and temporal properties [11]. The Time-Series GAN (TimeGAN) preserves the temporal dynamics of time-series data by introducing a step-wise supervised loss to allow the generator and discriminator to learn the conditional distribution of the data, as well as an embedding network to reduce the dimensionality of the feature space during training. TimeGAN combines the adversarial learning framework with autoencoders to simultaneously learn and optimize the supervised and unsupervised losses, as shown below.

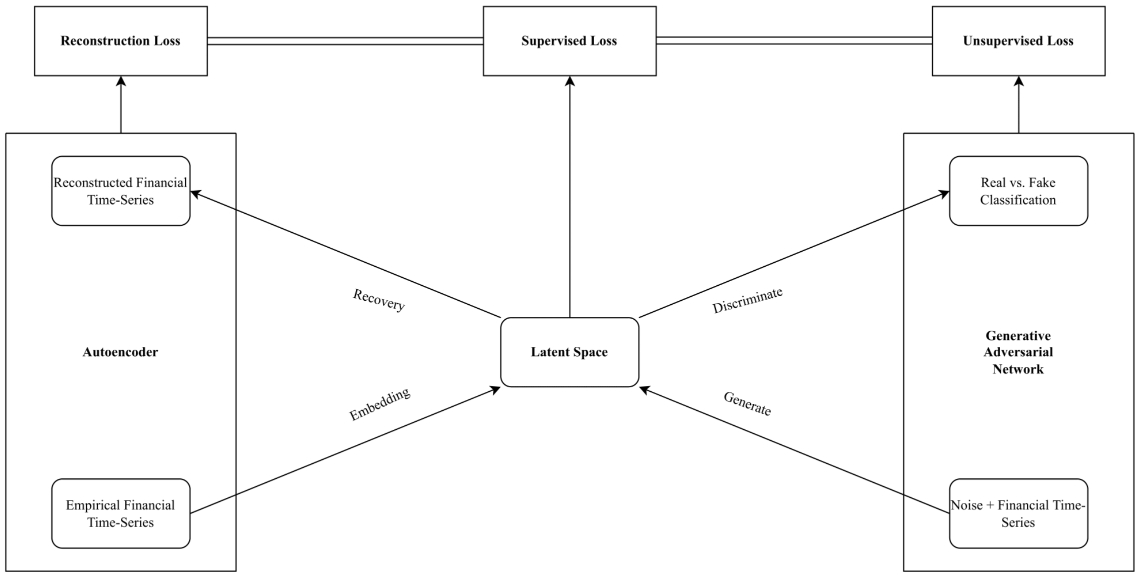
Reconstruction Loss
Supervised Loss
Unsupervised Loss
Reconstructed Financial Time-Series
Real vs. Fake Classification
Recovery
Discriminate
Autoencoder
Latent Space
Generative Adversarial Network
Embedding
Generate
Empirical Financial Time-Series
Noise + Financial Time-Series

Figure 2-3: Structure of a Time-Series GAN

2.2 Using GANs for Synthetic Data Generation

Although state-of-the-art applications of GANs involve image processing tasks, this framework can be adapted to work with data other than images, including tabular and time-series data. The goal of using GANs as an alternative approach to generating and simulating data is to remove the need for parameterization and assumptions about market dynamics. Since GANs are an unsupervised learning and non-parametric framework, they learn the distribution directly from real data instead of relying on the careful selection of parameters.

The trained generator portion of the GAN can be used to generate new data given a starting point and random noise input. During the synthetic data generation process, the discriminator portion of the GAN is not required, as it served its primary purpose of improving the generator and helping it learn the underlying data-generating process during training. As discussed before, the objective of the GAN is to generate a diverse set of possible market scenarios representative of the true patterns and dynamics observed in empirical market data. In the context of simulating financial time-series data of spot prices in the market, this translates into using the generator to generate spot price paths that have relatively similar distributional statistics and movement patterns to real spot price data [12].

Markets can be bullish, bearish, or stagnant at most points in time, but true market dynamics can be incredibly complex and noisy which presents a huge challenge for working with financial data. Ideally, the generator will be able to simulate many possible market scenarios that are representative of the complexity and noisiness present in empirical data. The possibilities and randomness of market dynamics are represented as random noise inputted into the generator when generating data.

2.2.1 Simulating Financial Market Data with GANs

Although the model can be adapted to work with any asset that has historical data available, we will focus on using the trained model for a specific stock, such as Apple (AAPL). AAPL has an abundance of historical price data available, dating back to the 1980s, with over 40 years of data. The generator requires a starting point in the financial time series to then input random noise and generate new data. Of course, at any point in time during the life of the company, there is a current price level and an unpredictable and uncertain future trend. Being able to simulate many realistically possible paths of data can be useful for training and evaluating algorithms that require an abundance of data. With the trained GAN, a large number of data points can be generated by simply producing random noise as input. The generator uses a starting point and combines random noise with the underlying distribution it has learned during training. Using the trained generator to simulate one path of spot prices for AAPL yields the following results displayed below.

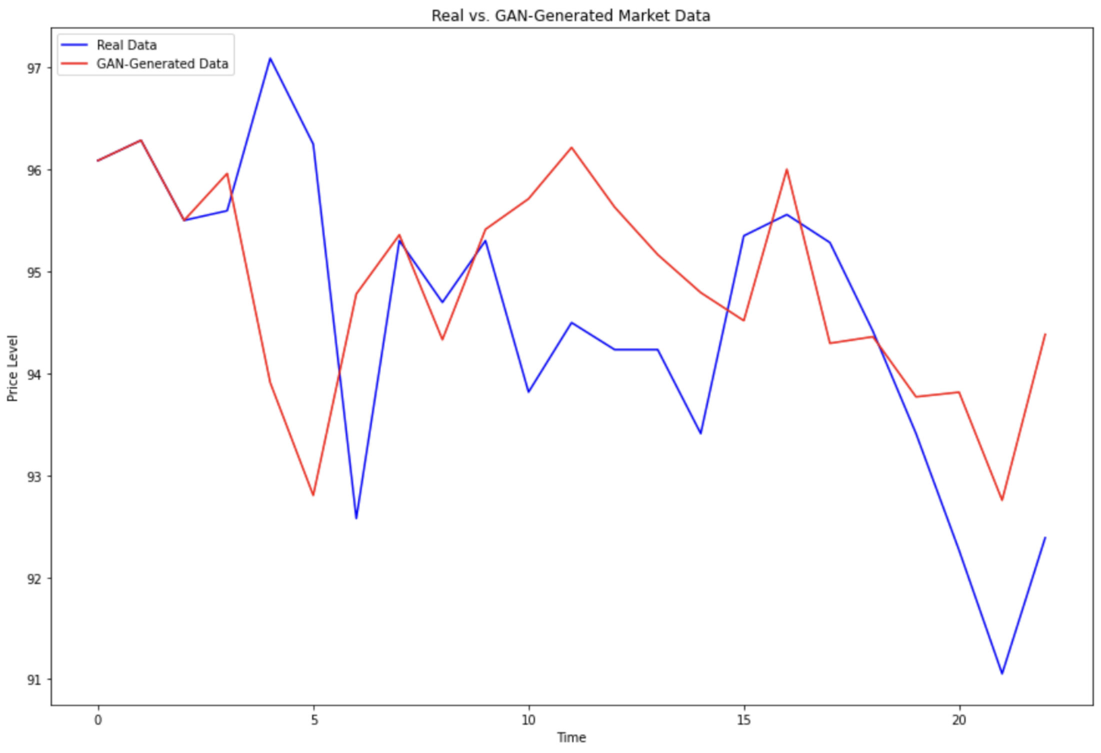

Figure 2-4: GAN-Based Spot Price Simulation

As we can qualitatively see in the plot above, the generated data exhibits similar properties as the real data and has similar movement patterns. The real and generated data both have the same starting point, at a price level of around $96, but the GAN has learned the dynamics of AAPL's historic price movement and has generated another plausible scenario of how the price might have evolved over time. This systematic data generation framework has great implications for a variety of financial applications, most importantly enhancing algorithms that require lots of data. The generation of many alternative paths from a starting point can be used as data to train and evaluate models and strategies. Ideally, the models and strategies that traders and investors develop are robust to different market conditions and possibilities, which can be captured by the GANs trained on historical market data.

For example, the generator of the GAN would ideally be able to produce a diverse set of market data, including data under both bullish and bearish markets. Using the trained generator, the diagrams below show AI-generated instances of the evolution of AAPL in bullish and bearish conditions.

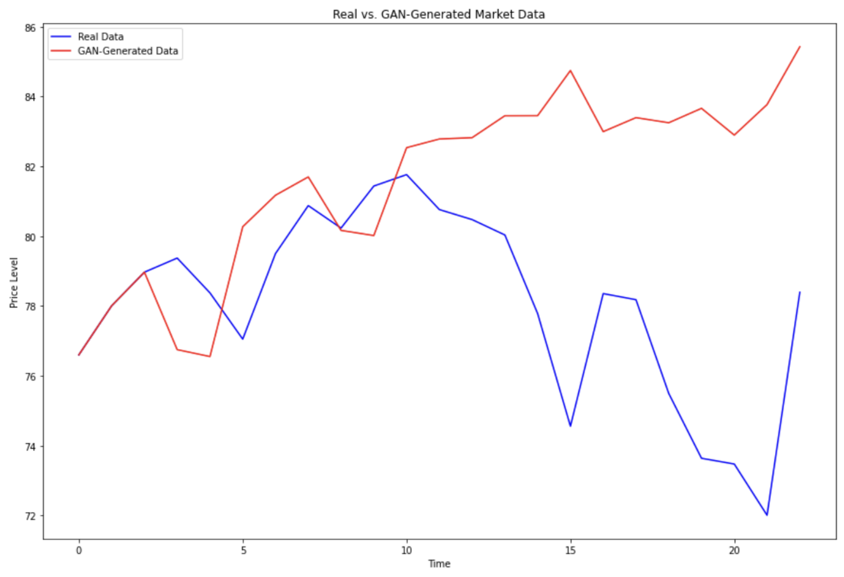

Figure 2-5: Upwards Trend Path Simulated by GAN

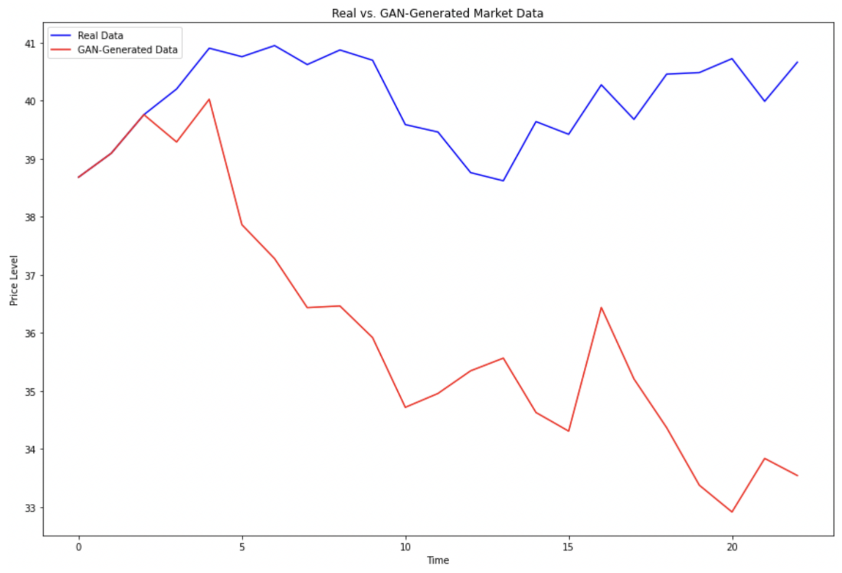

Figure 2-6: Downwards Trend Path Simulated by GAN

The figure above shows an AI-generated path that has an upward trend compared to the real historical data. As shown above, the red lines represent the GAN-generated data and the blue lines represent the actual data. In figure 2-5, the generated data appears to have an upward trend relative to the actual historical data, while in figure 2-6, the generated data appears to have a downward trend relative to the actual historical data. Although the real and generated data have different trends, the temporal properties of stock price dynamics appear to be relatively similar. This divergence in the real and generated paths, along with the preservation of price movement dynamics, is essential for being able to use GAN-generated data for training and evaluating models and strategies. It is possible to use this generator to generate an abundance of realistic market data under various market scenarios, similar to using Monte Carlo simulations. The difference is that while Monte Carlo simulations are parametric and probability-based, which may not capture the complexities and patterns of real market dynamics, GAN-generated data exhibits properties similar to empirical data and has a similar underlying distribution. It is interesting to observe, however, that a plot of GAN-generated series looks very similar to Monte Carlo simulations since both methods capture a variety of possibilities of stock price movement. A plot of GAN-based market simulation for AAPL is displayed below.

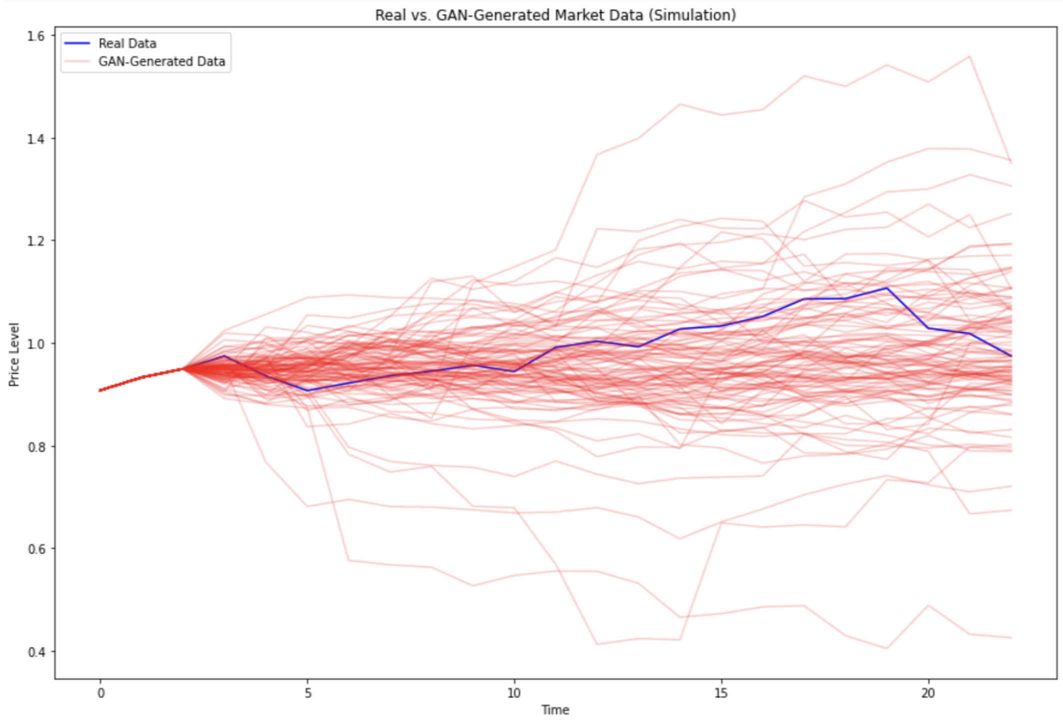

Figure 2-7: GAN-Based Market Simulation of Price Paths

2.3 Evaluating GAN-Based Market Simulations

To be confident in the accuracy of GAN-generated market data in representing empirical data, there must be a systematic framework for quantitatively evaluating the similarity of synthetic data to real data. Unlike synthetic image generation with GANs, where the quality of the generated data can be visually observed by humans, synthetic time series and financial data are harder to evaluate directly. It is easier for humans to observe the facial features of generated images than it is to identify common-occurring statistical properties present in time series data.

However, since GANs are an unsupervised machine learning framework, there isn't a single or standard way to evaluate performance [13]. With supervised machine learning methods, the standard way of measuring performance is to make predictions on a test data set and compute a metric such as accuracy or root mean square error. Since GANs are generating new data, it is infeasible to use the same evaluation methods. However, this does not mean that the similarity between GAN-generated synthetic data and real data cannot be measured. There are various metrics and statistics apparent in time-series data that can be measured, evaluated, and compared between synthetic market data and real market data. Ideally, these metrics will be similar in value which indicates that the GAN has learned the underlying distribution of the financial time-series data. Although it is difficult to explain GAN outputs, the metrics will bring confidence into the GAN learning market dynamics from data.

2.3.1 Time-Series Distributional Statistics & Metrics

There are various time-series metrics and statistics that can be computed that describe the temporal dynamics and conditional distribution of the time-series data. For example, distributional statistics such as mean, median, minimum, and maximum may be useful for comparing real and synthetic data to ensure the synthetic data has a relatively similar distribution. There should be some degree of variation in the statistics so that the GAN is able to generate a diverse set of data.

However, properties that exist in financial time-series data are often more complex in nature. It is simple to generate data with similar distributional statistics, as Monte Carlo methods do well, but difficult to mimic the exact temporal features that exist in financial data, since the exact properties of stock returns are unknown. To quantitatively evaluate the ability of the GAN to generate meaningful representations of market data, time-series feature extraction can be used. Maximilian Christ et al. (2018) explore various time-series feature extraction [14] methods to gain insight into the temporal dynamics of time-series data. The **tsfresh** package encapsulates over 794 time-series features that can be used to describe the properties and behaviors of time-series data. A comprehensive list of all extracted features can be found on the official **tsfresh** documentation page. A summary of the time-series metrics is displayed below for real market data, GAN-generated data, and Monte Carlo simulations.

Statistics	Mean	Median	Variance	Skewness	Kurtosis
Real Market Data	20.45	19.86	1.64	0.61	-0.96
GAN Market Data	20.78	20.04	2.28	0.26	-1.66
Monte Carlo Simulation	22.37	21.49	1.32	0.21	-0.25

Table 2.1: Distributional Statistics of Real vs. Synthetic Market Data

Autocorrelation	Lag 1	Lag 2	Lag 3	Lag 4	Lag 5
Real Market Data	0.84	0.72	0.61	0.47	0.36
GAN Market Data	0.95	0.86	0.75	0.66	0.52
Monte Carlo Simulation	0.67	0.42	0.35	0.21	0.15

Table 2.2: Autocorrelation of Real vs. Synthetic Market Data

As shown from the distributional statistics and autocorrelations of the real and GAN-generated market data above, they appear to have similar properties. GAN-generated market data seems to emulate the patterns of real market data more closely than Monte Carlo simulations. This implies that there may be some specific properties or dynamics of stock price movement that are unable to be modeled by simple probabilistic simulations. These results demonstrate the efficacy of using GANs to generate representative market data that may be useful for other purposes.

2.3.2 t-SNE Comparison

There are a large number of tsfresh time-series features to compare manually. One method to evaluate how similar the metrics of the generator's synthetic data are to the empirical data is to use the t-distributed stochastic neighbor embedding (t-SNE) [15] algorithm. As high-dimensional data is difficult to visualize, the t-SNE method visualizes the high-dimensional inputs by mapping the data from a high dimension to a low dimension. Using this method, we can compare the t-SNE mapping visualizations between the time-series metrics of generated market data and real market data. Using the generator to generate 1000 samples, the t-SNE plot of real (teal) and synthetic (red) financial time-series data metrics is displayed below.

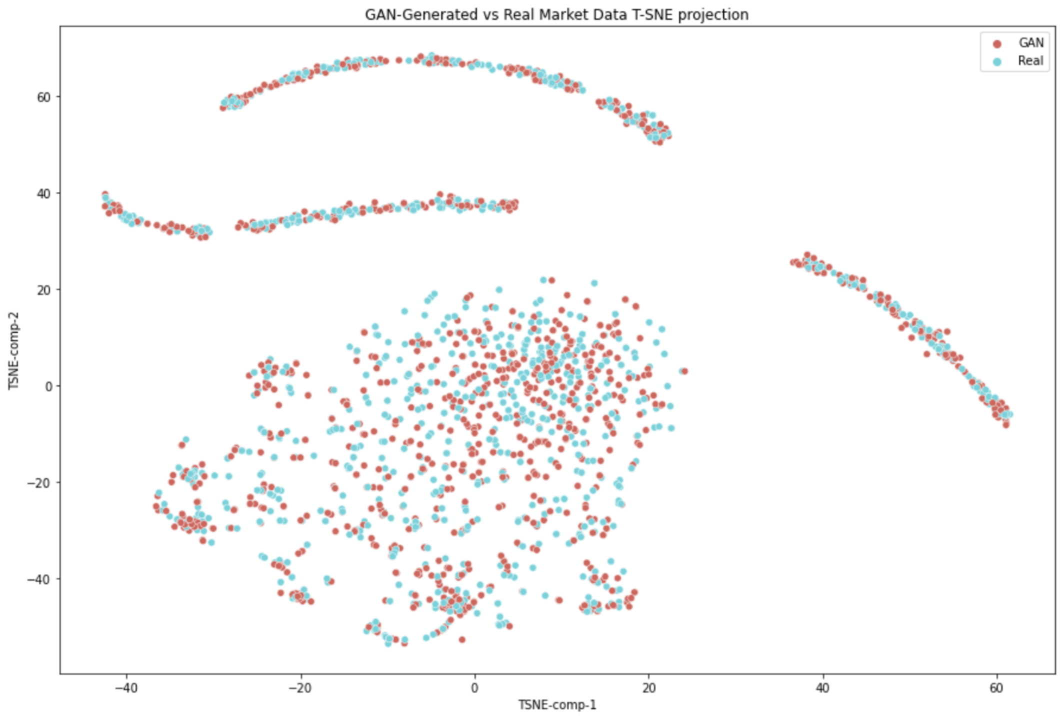

Figure 2-8: t-SNE Visualization of Real vs. Synthetic Market Data

The t-SNE visualization shows that the high-to-low dimensional mapping of the real data closely follows that of the GAN-generated data. This shows that the time-series features previously described are similar for both real and synthetic data, implying the GAN's ability to learn the properties of empirical stock returns. Being able to model complex market dynamics has great potential as will be explored in later chapters.

Chapter 3

Deep Reinforcement Learning for Derivatives Pricing and Hedging

3.1 Motivations

Previous work on derivatives pricing relies heavily on assumptions about the market and underlying stock movement to be consistent with theoretical standards for solving partial differential equations and deriving pricing models. For example, Monte Carlo simulations rely on the assumption that stock prices move according to a geometric Brownian motion stochastic process. As mentioned in Chapter 1, one of the most famous and widely-used options pricing models is the Black-Scholes Merton model. This framework is derived from a partial differential equation that also relies on many market condition assumptions that may not necessarily hold in real-world markets. For example, the model assumes through the option's lifetime: no dividend payouts, constant risk-free interest rate, constant underlying volatility, and no transaction costs. The model also assumes the market has no-arbitrage conditions and there are no transaction costs.

However, these factors are all empirically dynamic and can affect underlying price movements and options premiums, but are not accounted for. These models serve as a good foundation for understanding some of the important factors that impact option prices, but in the real market, market dynamics can be affected by a large number of constantly changing factors. Many market frictions exist and are changing over time, which impact the value of financial derivatives. Furthermore, there are many types of different financial derivatives, each with unique properties, with options being one of the most popularly traded. A practical model would be developed on the basis of real market data that reflects the properties of market conditions on which the derivatives are traded, including transaction costs and other frictions.

3.2 Derivatives Pricing Models

To understand the fundamental ideas behind derivatives pricing and hedging, we first examine the theoretical approach with traditional options pricing models applied to risk-neutral pricing. As discussed in Chapter 1, the Black-Scholes model is the most widely-adopted foundation for understanding and analyzing option value and the factors that influence it. However, the model is not based on historical stocks and options data, but instead assumes the properties of stock price movement and makes many idealistic assumptions about market dynamics. Iterative improvements have been proposed to improve the framework, such as modeling using stochastic volatility as with the Heston model. Furthermore, derivatives pricing models have been developed for other types of assets other than equities, with different movement dynamics.

For example, the Vasicek model is commonly used to model the evolution of interest rates to price interest rate derivatives. Another example is Black's model, an extension of the original Black-Scholes model, commonly used to price options on futures. Many models have been developed for various purposes and financial instruments, but they all rely on parametric assumptions instead of historical market data. This makes the basis of these models scientific, as the models are able to explain the specific factors that influence derivative prices, but potentially inaccurate as market dynamics are constantly changing. These models can also be applied in the context of devising hedging strategies to manage the risk of changes in the underlying factors.

3.2.1 The Greeks

Derivative prices are affected by a large number of variables that traders must constantly account for when developing trading and hedging strategies. There are some obvious factors that affect option prices that are captured, for example, by the Black-Scholes model: underlying price, time to expiration, volatility, and interest rate. The impact that a change in each of these aforementioned factors has on option prices is captured by the Greeks, which measure the sensitivity of option prices to these factors. Traditionally, traders and investors devised hedging strategies using the Greeks for the options they traded to account for the risk factors of option prices. Hence, various popular hedging methods used by many traders today include those such as delta hedging and gamma hedging, so that portfolios may remain delta or gamma neutral. The idea is to reduce the sensitivity of the options portfolio's value to underlying price movements as much as possible. A table of the most commonly-used Greeks and the corresponding definitions is displayed below.

Greek	**Value**	**Sensitivity to**
Δ Delta	$\frac{\partial V}{\partial S}$	Underlying Price
Γ Gamma	$\frac{\partial^2 V}{\partial S^2}$	Delta
Θ Theta	$\frac{\partial V}{\partial t}$	Time
ρ Rho	$\frac{\partial V}{\partial r}$	Interest Rate
ν Vega	$\frac{\partial V}{\partial \sigma}$	Volatility

Table 3.1: The Option Greeks

The option pricing models previously discussed provide a method to derive these Greeks for various options. Using the Greeks to devise hedging strategies is a reasonable approach in the absence of sufficient data and computational capacity [16]. However, this approach to solving the hedging problem is not as efficient as it could be, given the recent advancements in data-driven machine learning methods. Nonetheless, the Greek-based method is still widely practiced as its idea is intuitive, which is to adjust positions to reduce the impact of changes in the impactful factors.

3.2.2 Dynamic Greek Hedging

Perhaps one of the most practiced approaches used for hedging is delta-hedging, which involves adjusting the hedge ratio of a derivative position in response to changes in the underlying asset price, reducing the risk exposure of the position. The delta of an option represents its sensitivity to underlying price changes. Since underlying asset prices, stock prices, for example, are dynamically changing over time, it is important to dynamically rebalance portfolio positions to mitigate risks. Many traders look to maintain a delta-neutral position, in which the overall delta of the portfolio is zero, in order to mitigate risks in underlying price movements. However, due to transaction costs and other market frictions, perfect hedges are very difficult to achieve, and dynamically rebalancing delta positions can be extremely costly and often impractical.

Hedging is a problem that has many complexities to it based on various market risks. Delta hedging is widely known for its simplicity in interpretation and implementation. However, derivative prices are often impacted by a larger number of factors and market frictions that exist in real markets. Instead of the risk-neutral pricing framework, it may be beneficial to explore machine learning methods for solving the hedging problem to provide more accurate and robust results.

3.3 Overview of Deep Reinforcement Learning

Reinforcement Learning

Reinforcement learning (RL) is a method that overcomes many limitations of traditional machine learning. It involves exposing an agent to an environment with different states, available actions, the corresponding rewards for those actions, and letting the agent learn to maximize rewards through a Markov decision process. The power of reinforcement learning stems from its robustness and ability to quickly adapt to changing conditions relative to other types of machine learning methods. The results of state-of-the-art reinforcement learning methods have been promising, such as AlphaGo, the AI developed by DeepMind that beat the world Go champion, and ChatGPT, the language model developed by OpenAI that can answer questions and generate text in a human-like manner.

In an environment with current state s_t at time t, an agent can perform a set of actions $a_t \in A$ that results in reward r_t at time t and moves the environment to the next state s_{t+1}. The environment is represented as a Markov decision process and the agent decides the optimal action to perform at each state s_t according to its policy, $\pi : s_t \to a_t$. A diagram showing the reinforcement learning framework for the hedging problem is displayed below.

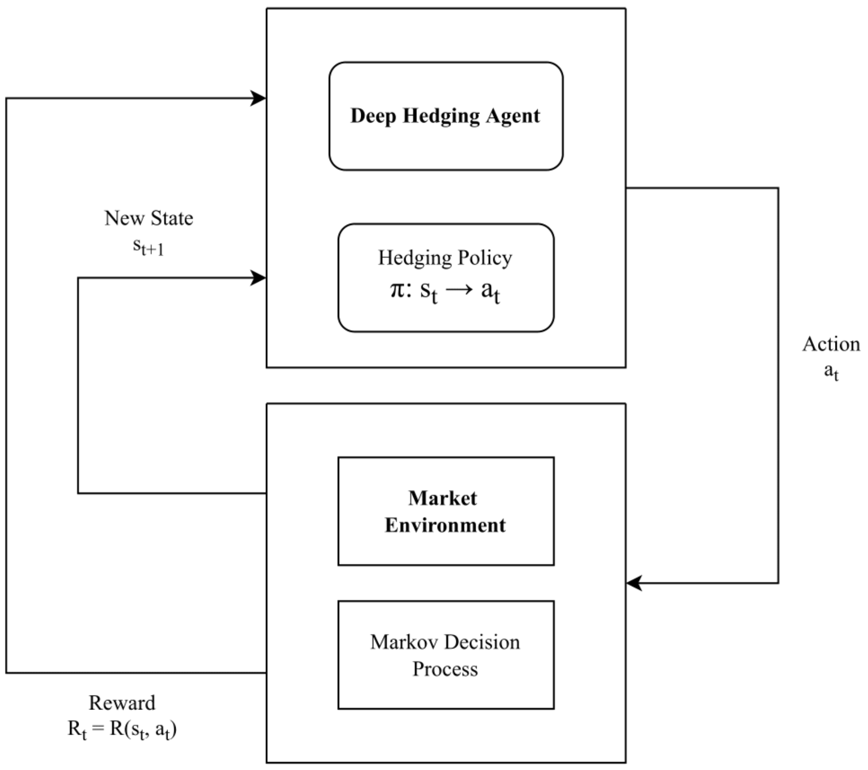

Figure 3-1: Reinforcement Learning for Hedging Framework

This framework can be applied to a financial market environment with options trading with empirical economic and financial factors. The benefit of this approach is that it allows these factors to change dynamically across various time periods, which the agent will account for and gradually learn as it trains and explores the market environment. The goal is to implement a deep reinforcement learning system that can accurately price derivatives and dynamically hedge under actual market conditions and frictions. This can be done by using empirical data to learn the optimal policies for pricing and hedging through a policy gradient method. In the market environment, the agent will be given market conditions at each state and will incrementally learn the optimal hedging policy for correctly pricing derivatives and compute a PnL and Sharpe ratio for the dynamic hedging strategy as a reward function. Finding the optimal hedging policy, $\pi^{\star}$, results in maximizing the discounted cumulative reward:

$$\pi^{\star}(s_t) = \max_{a_t \in A} \mathbf{E}[\sum_{t=1} \gamma_t R_t | s_t, a_t] \tag{3.1}$$

with γ_t as the discount factor at time t, as the agent will also want to account for the rewards from future states as the result of its actions. There are two ways to optimize the reinforcement learning agent. One way is to directly learn the expected rewards with a Q function, $Q(s_t, a_t) = \mathbf{E}[\sum R_t | s_t, a_t]$ given a state and action pair (s_t, a_t) and then using the Q function to optimize the objective in equation 3.1. This method is commonly known as Q-learning, a form of value learning algorithm, to learn the value of an action at a particular state. The second method is to directly learn the optimal policy, $\pi^{\star}(s_t)$, without learning a mapping function of state and actions to rewards. This approach is known as policy learning and has produced promising results in state-of-the-art reinforcement learning algorithms.

Combining Deep and Reinforcement Learning

Early concepts of deep learning have been present since the 1940s, such as artificial neurons and perceptrons that could learn simple representations of data. However, because of early artificial neural networks' limited complexity combined with limitations in data and computing abilities, deep learning technology did not become widely adopted by institutions until the 21st century. In the present time with exponentially growing amounts of data and advanced processing units for fast computation and matrix algebra, the true potentials have deep learning have become gradually realized. The core of deep learning lies in its ability to learn incredibly complex and abstract relationships, both linear and non-linear, from vast amounts of data and be able to make accurate predictions using the learned representation. Often when the amount of data is enormous and highly dimensional, deep learning has proven to be significantly more effective compared to traditional forms of statistical learning.

Nevertheless, deep learning on its own has limitations, particularly in the context of modeling and predicting financial markets. Deep learning learns from the data it is provided but suffers from data drift, when the underlying distributions of the data change, and concept drift, when the actual factors that influence predictions changes. Because of these issues and the dynamic nature of the markets, it is difficult to develop a sustainable systematic framework for investing and hedging based solely on deep learning methods. However, combining the prediction power of deep learning with the robustness and flexibility of reinforcement learning, known as deep reinforcement learning, may yield much better results when applied to the financial markets.

There are various state-of-the-art deep reinforcement learning algorithms that have recently been developed and have seen success in many applications, from playing games to autonomous driving. Deep Q-networks (DQN) is one type of algorithm that uses deep neural networks to approximate Q values as previously described. This framework was first used by DeepMind [17] to play Atari games by using a convolutional neural network to learn the value function of future rewards based on the game pixels. Another successful algorithm is known as proximal policy optimization (PPO) [18], a value function learning-based method, which has performed relatively similar or better than other state-of-the-art algorithms. Both of these deep reinforcement learning algorithms, and many others, can be applied to training agents to learn optimal hedging policies. Petter Kolm et al. demonstrate how proximal policy optimization can outperform traditional delta hedging and other deep reinforcement learning algorithms in finding the optimal hedging policy and optimizing PnL, training time, and amount of data required for training [19].

In a simplified world with only a limited number of states of the market, traditional reinforcement learning may be used, since the state-action-reward combinations are easily computed. However, in environments such as the financial markets, there are a very large number of states, with combinations of different factors including risk-free rates, stock movements, costs, etc. To account for the large dimensionality and complexity, this study examines the use of deep reinforcement learning methods to learn the states and rewards of the market by fitting a deep neural network on empirical data and using the learned states and rewards to train a reinforcement learning agent. The deep reinforcement learning framework for hedging is displayed below.

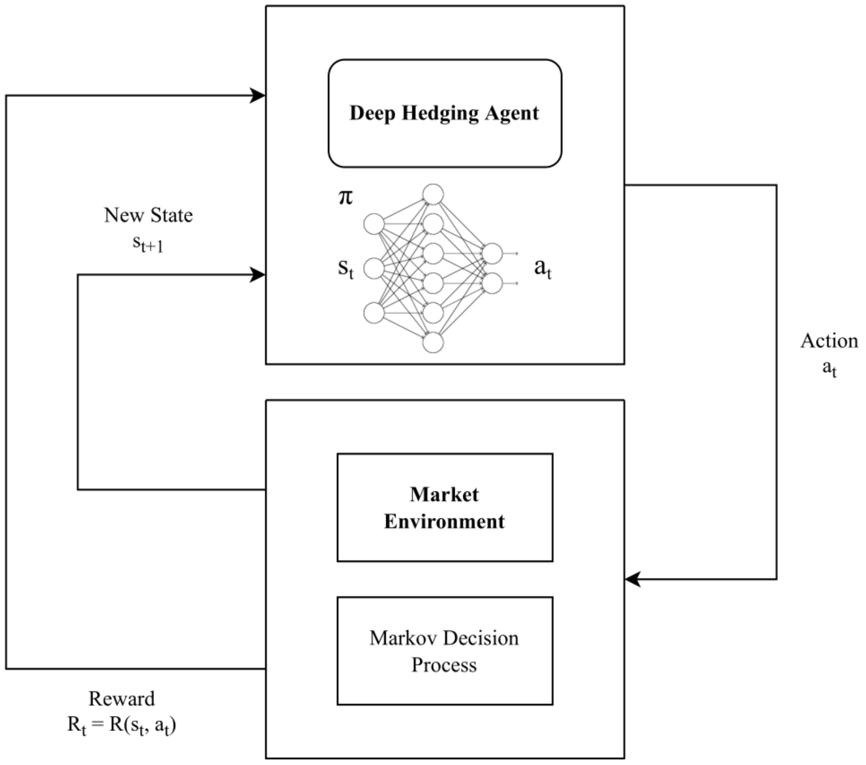

Figure 3-2: Deep Reinforcement Learning for Hedging Framework

3.3.1 Model-Based vs. Model-Free

Some investors try to understand the overall market structure and make trading decisions based on their understanding, while other investors base trading decisions on their past experiences of successes and failures. In the reinforcement learning methodology, agents can also learn in these two types of ways. The former way, where the agent tries to learn a model of the environment to make predictions on the consequences and resulting reward of its current actions is known as model-based reinforcement learning. In this approach, the agent predicts rewards before any action is taken. This allows the agent to predict the future outcome of its actions based on the model of the environment it has learned. However, this makes the agent greedy and the agent may carry out any action despite immediate consequences. Furthermore, the learned model of the environment may be inaccurate when environment dynamics change, leading the agent to make poor decisions. Financial markets are one example of an environment that is often unpredictable and constantly changing, and a representation of the overall market in the present day may not be representative of the next. Consequently, model-based approaches may have weaknesses when applied to solving financial problems, just as investors may find it difficult to consistently predict the outcomes of their trading decisions before those trades are actually made.

To account for the dynamic nature of the financial markets, it may be more appropriate for agents to optimize their policies based on the rewards after actions have been taken, just as investors examine their PnL and Sharpe ratios after trades are executed. The latter way, where the agent learns to optimize its policy solely on its experience interacting with the environment during the training process is known as model-free reinforcement learning. In the context of hedging policies, the agent will learn by executing various strategies and observing the results of those strategies to optimize its policy. This allows the agent to be more robust to changing market conditions and learn directly from experience in the market.

3.3.2 Multi-Agent Reinforcement Learning

Oftentimes, simulations are carried out by more than one agent. For example, in financial markets and exchanges, there are many market participants including traders and investors that are simultaneously trading with each other. The orders that are made and executed between these market participants contribute to price movements.

However, understanding and being able to simulate the interactions between these market participants may be beneficial for determining optimal trade executions and strategies. These market participants can be represented as multiple agents in the reinforcement learning framework. Instead of just a single agent in the environment, the market environment consists of multiple agents competing against each other to achieve the maximum reward. The multi-agent deep reinforcement learning framework for multiple trading agents is displayed below.

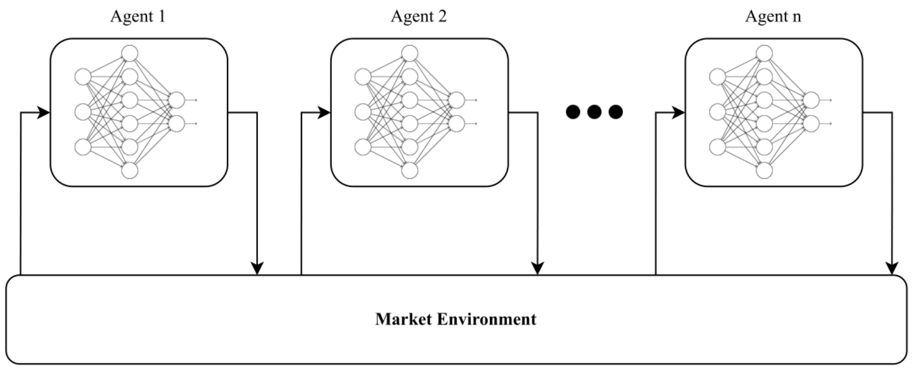

Figure 3-3: Multi-Agent Deep Reinforcement Learning Setup

Furthermore, it may be possible to represent all the agents in the system with a single generator agent that produces market transaction data. Victor Storchan et al. propose a framework called "MAS-GAN" [20] to generate realistic market data by having a GAN represent the pool of market participants and learning to produce synthetic order book and bid-ask price data. This system can be used to not only train deep hedging algorithms on realistic multivariate time-series data but also evaluate their robustness to responses in the market, which is not accounted for in backtesting.

3.4 Deep Hedging

3.4.1 The Hedging Problem

Portfolios are constantly subject to a variety of risk factors that may significantly impact the portfolio's value and cause high volatility in returns. Hedging is a popular method to combat the risk that subjugates portfolios. The main idea behind this approach is to reduce risk and volatility by taking an offsetting position in another asset, such as a derivative of the underlying. However, hedging can be costly in a market with transaction costs and other frictions, along with the financial constraints of various investors. An effective hedging strategy must not only account for all the transaction costs involved but also all other imposed constraints.

Consider a discrete-time market at times $t_n \in T = \{0, t_1, t_2, ..., t_N\}$ with a tradeable risk-free asset of price $B_n = e^{rt_n}$, and D tradeable risky assets of prices $S_n = [S_n^1, S_n^2, ..., S_n^D]$. There is a tradeable derivative in the market on the risky asset with a payoff equal to $\phi(S_n)$. In a complete market, it is possible to find a portfolio that fully replicates the payoff of a derivative. However, real markets are incomplete and strategies cannot replicate derivative payoffs, creating the objective of deciding the optimal policy for pricing and hedging traded derivatives to minimize the portfolio's value risk. Let the value of the portfolio at time t_n be $V_{t_n} = B_{t_n}(V_{t_0} + DCG_{t_n})$ where V_0 is the initial capital and DCG_n is the discounted cumulative gain of the hedging portfolio at time t_n. The asset allocations to each risky asset, S_n, is defined by $\vec{\delta} := \{\delta_n\}_{n=0}^N$ resulting in the hedged portfolio. To find the optimal hedging policy that minimizes the hedged portfolio risk, the objective of the hedging problem then can be defined as:

$$\delta^* = \min_{\delta \in \Delta} \rho(V_{t_N}^{\delta} - \phi(S_{t_N})) \tag{3.2}$$

where δ^* is the optimal hedging policy that minimizes portfolio risk in a set of Δ possible hedging strategies. In this case, ρ represents a risk measure such as value at risk, conditional value at risk, quadratic penalty, etc. The ultimate goal of the hedging problem is to optimize this objective and find the optimal hedging strategy under realistic market conditions including transaction costs and other frictions. However, because of the complexity and dimensionality of a realistic market environment, there is no closed-form solution for the optimal hedging strategy in equation 3.2. Instead, to approximate the optimal hedging policy δ^*, the hedging strategies Δ can be parameterized as deep neural networks that can be trained to learn on data.

Ideally, to minimize a portfolio's risk exposure to underlying price movements, the portfolio should constantly be rebalanced to remain as close to delta-neutral as possible. However, as real markets typically have transaction costs, hedging can be very costly. Consequently, traditional Greek hedging methods such as delta hedging are not optimal and do not perform as well under real market conditions, having limited flexibility. Alternatively, a reinforcement learning approach can potentially learn the relationship between market states and the optimal hedging policy [21].

3.4.2 Applying Deep Reinforcement Learning to Hedging

Deep hedging is a systematic framework for using deep reinforcement learning to learn the optimal policy for derivatives pricing and dynamic hedging based on the aggregation of market data. The deep hedging approach uses neural network representations to learn and optimize hedging policies. Hans Buehler et al. (2019) demonstrate in the original paper that neural networks can approximate well optimal hedging policies under general market conditions in the presence of frictions such as transaction costs. As some advantages of deep learning methods include being able to handle high dimensionality and being able to learn complex non-linear relationships, deep hedging has the potential to model these relationships between market factors and derivative prices. Deep hedging agents learn to dynamically adjust a portfolio's hedges in response to changing market conditions, which can make the algorithm significantly more robust and precise compared to traditional Greek hedging methods.

The goal of deep hedging is to minimize the overall risk of a portfolio by continuously adjusting the level of hedging in response to changes in market conditions. This involves the use of various hedging instruments including options and futures to hedge against potential large swings in the portfolio's value by trying to minimize the portfolio's risk. Deep hedging moves away from the risk-neutral world, where markets are complete and there is no arbitrage, to the real-world measure. As machine learning methods are data-driven, this framework can be applied to a wide range of financial assets, including stocks, bonds, commodities, currencies, and crypto.

Reinforcement learning is an appropriate tool for the task of hedging because it can be adapted to work with different combinations of market conditions, portfolio complexities, and investment objectives. Unlike traditional optimization tasks, reinforcement learning agents remove the need to manually solve for optimal positions by interacting and learning the optimal strategy directly from a realistic market environment through trial and error. At a high level, deep hedging uses deep neural networks to predict the evolution of the markets and then uses this prediction to allow agents to learn how to dynamically adjust a portfolio's hedge positions to minimize risk.

Considering the hedging problem objective described in equation 3.2, a deep neural network can be used to represent hedging strategies and predict corresponding rewards in the market environment. This neural network can be used to allow the reinforcement learning agents to gradually learn the optimal hedging policy using this policy approximation framework. The structure of the hedging strategy neural network representation is displayed below.

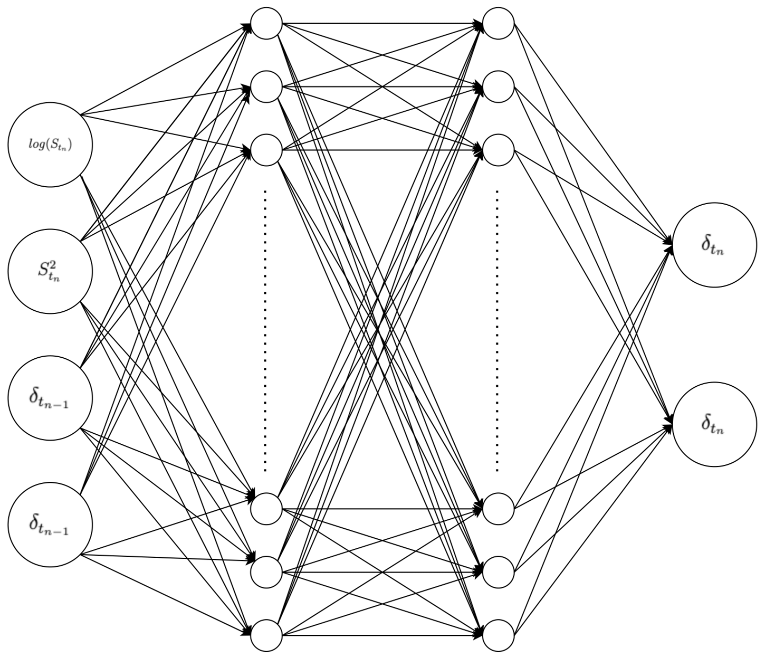

Figure 3-4: Learning Hedging Policy from Market State Deep Neural Network

3.4.3 GAN-Based Market Environment

The flexibility of the deep hedging framework allows the algorithm to learn from any type of market environment with any number of variables and factors. However, for the deep hedging algorithm to be practical for institutional investors and traders, the market environment must be as realistic and representative of real-world markets as possible. The original deep hedging system developed by Hans Buehler et al. (2019) was implemented under a Heston world with stochastic volatility, an improvement upon Black-Scholes, but is still parametric and makes assumptions about market dynamics. The authors suggest the use of a market simulator powered by machine learning methods, which may be more appropriate to represent real market conditions.

As explored in Chapter 2, GANs can be applied to the deep hedging framework by generating a diverse range of synthetic market scenarios that can be used to stress-test the hedging algorithms. The synthetic data that the GAN generates has statistical properties resembling real data. This data can help ensure that the hedging algorithm is robust and can handle a wide range of market conditions. GANs can be used to augment the training data for the hedging algorithm by generating additional synthetic data points that are representative of the real-world market. This can potentially help to improve the hedging algorithm's performance by increasing the amount and diversity of realistic training data available.

Market environments have frequently been simulated using a model such as Black-Scholes for the sake of simplicity and interoperability of market dynamics. In this implementation of deep hedging, the market environment is created by using GAN-based market simulation to capture the general properties of real markets, moving the system into the real-world measure. This creates a non-parametric model of the market, removing the need for making various assumptions about market dynamics. The overall framework of multi-agent deep reinforcement learning and GAN-based market simulation for derivatives pricing and dynamic hedging is displayed below.

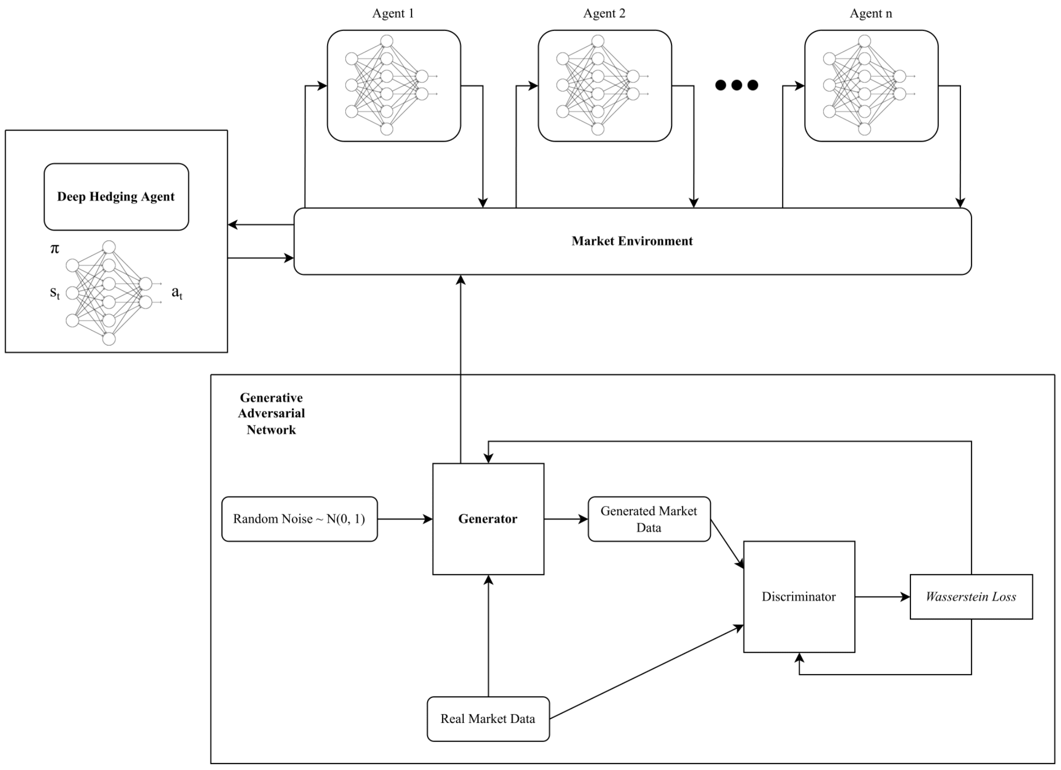

Figure 3-5: Framework of Multi-Agent Deep Reinforcement Learning and GAN-Based Market Simulation for Derivatives Pricing and Dynamic Hedging

3.5 Evaluating Deep Hedging Algorithms

Using machine learning methods for derivatives pricing and hedging moves away from the risk-neutral pricing framework and does not require the assumptions and computations of a risk-neutral world. As previously mentioned, the reinforcement learning framework of deep hedging provides flexibility as to the type of market environment and constraints in the hedging problem. Previous works on using deep hedging methods have focused on the implementation of a Black-Scholes or Heston world. It is demonstrated by Hans Bühler et al. [2] that deep hedging is able to converge to the results of Black-Scholes delta in the absence of market frictions and transaction costs. However, using Black-Scholes or Heston to simulate underlying price movements still makes the deep hedging system model-based, as it relies on foundational assumptions about market dynamics. Instead, it may be more appropriate to use the GAN-based market simulator to represent the market environment.

3.5.1 Black-Scholes World vs. GAN-Based Simulations

The primary goal of using machine learning methods in deep hedging is to move away from the model-based and risk-neutral world framework that requires many assumptions about market dynamics. Ideally, a model-free system would learn and make decisions primarily on data instead of a model of the market, which is the purpose of the reinforcement learning system. To realize the true potential of machine learning and deep hedging methods for practical use in real-world markets, a model-free environment must be implemented, such as the one previously explored in Chapter 2, that does not require assumptions about how the market works. To observe the differences in using a Black-Scholes world and a GAN-based market simulator to train the deep hedging system, the realized hedged PnLs of the optimal hedging strategies can be visualized for both types of market environments. Running 100,000 market simulations with 30 time steps using both approaches yields the distribution of hedged PnLs displayed in the histogram below.

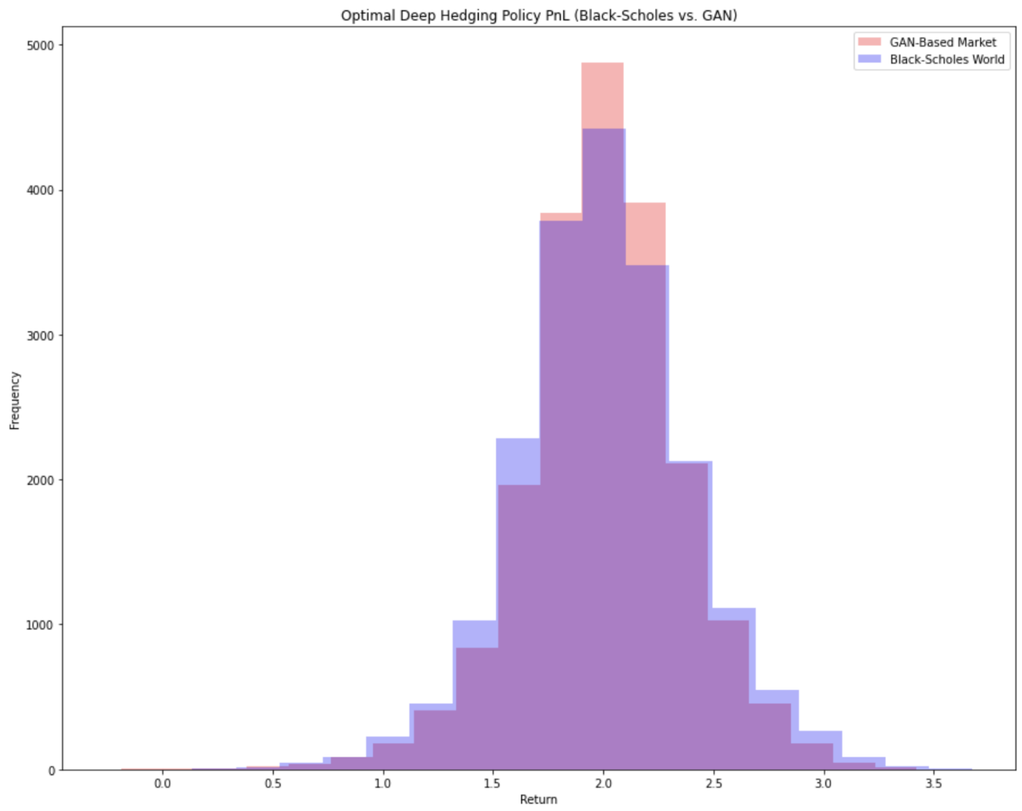

Figure 3-6: Optimal Deep Hedging Policy PnL (Black-Scholes vs. GAN)

Derivative prices are sensitive to changes in the underlying asset prices. This sensitivity, as described in Section 3.2.1, is referred to as the derivative's delta. The delta, or sensitivity to underlying price movements, of the hedged option can be visualized as a function of the underlying price over time. One of the goals of using hedging strategies is to reduce the portfolio's delta exposure as much as possible so its value is less sensitive to underlying price movements. This implies that the delta plot will ideally be less steep to reduce the sensitivity and risk exposure. Over time as the option nears expiration, its value becomes more sensitive to underlying price movements and the delta becomes much steeper than before. The portfolio deep hedging deltas for both the Black-Scholes and GAN-based environments are plotted below.

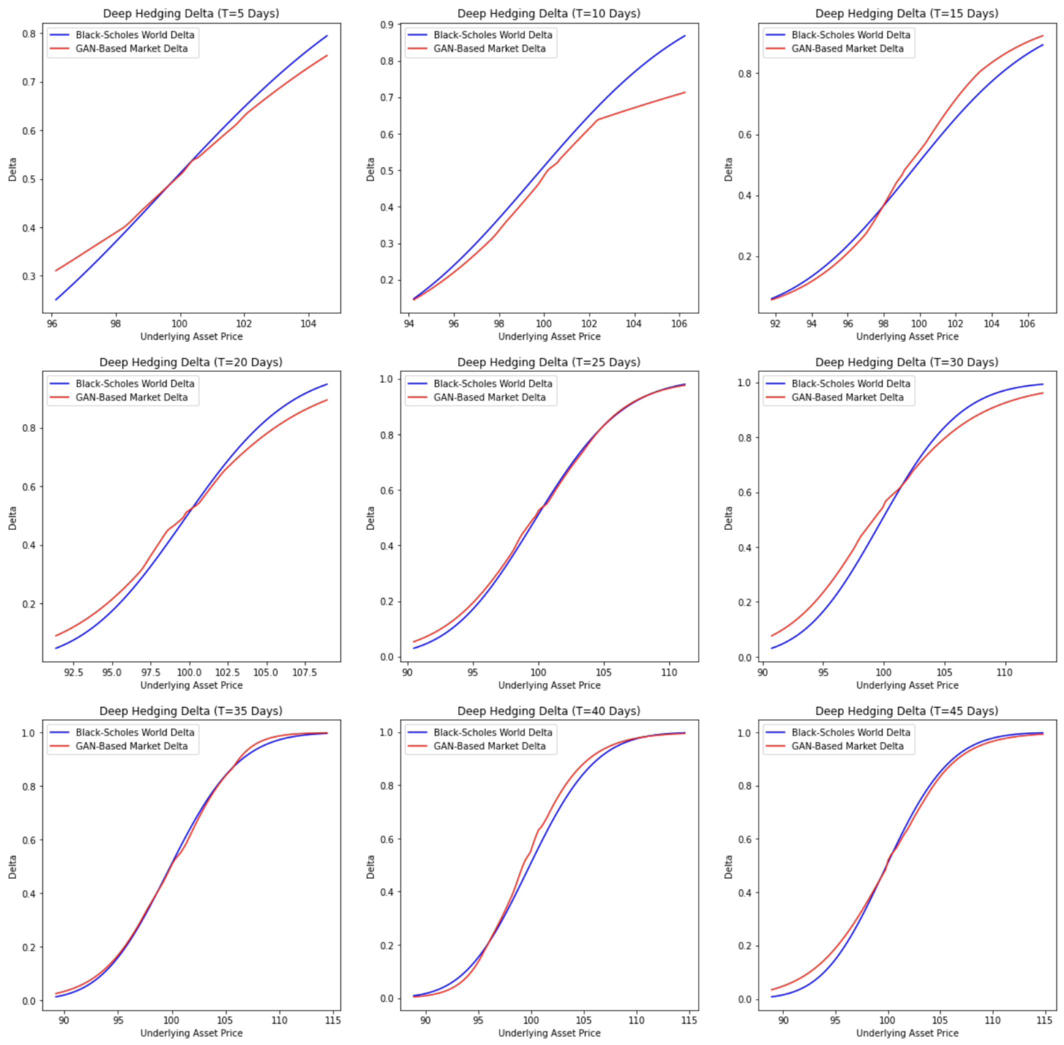

Figure 3-7: Deltas for Deep Hedging Over Time (Black-Scholes vs. GAN)

As shown in the distribution of PnLs in Figure 3-6, it appears that deep hedging has more consistent returns using AI-generated data than Black-Scholes simulations, as the returns are more thinly spread when trained on a GAN-based market. Furthermore, it can be seen in Figure 3-7 that using GAN-based market simulation results in a smaller overall delta for various underlying prices. This implies that the deep hedging algorithm was able to reduce the delta risk exposure and sensitivity to underlying price movements of the portfolio using AI-generated market data instead of a parametric Black-Scholes simulation.

To have a robust hedging strategy, the PnLs of the portfolio should have consistently low volatility in different market scenarios. The volatilities of the returns for the strategy should be relatively stable across different market simulations so that the deep hedging agent can perform consistently and handle various market risks in the real world. The distribution and kernel density estimate (KDE) for the volatility of hedged returns for deep hedging on a Black-Scholes world and GAN-based market environment for an option with strike $K = 100$ are displayed below.

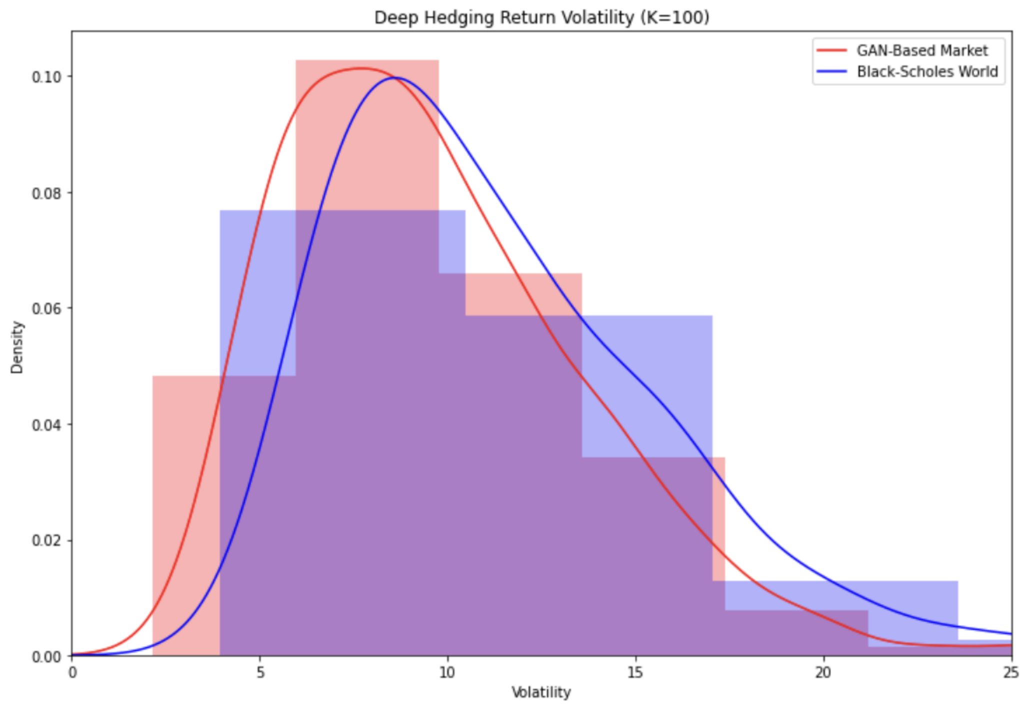

Figure 3-8: Histogram of Deep Hedged Return Volatility (Black-Scholes vs. GAN)

Return Volatility	Mean	Standard Deviation
Black-Scholes Market	11.72	5.75
GAN-Based Market	9.72	4.26

Table 3.2: Distribution of Deep Hedged Return Volatility (Black-Scholes vs. GAN)

Observing the results of using a Black-Scholes and a GAN-based market environment for training deep hedging methods, it appears the latter produces a more robust hedging policy for reducing volatility against various market risks and conditions. Across all the simulations, the hedging strategy PnLs and volatilities are more thinly spread in the GAN-based market. The GAN is likely capturing specific properties of financial data that cannot be modeled by parametric methods like Monte Carlo simulations. Overall, it appears that the AI-generated data allows deep hedging agents to better learn the market structure and make more consistent hedging decisions. Overall, these results are promising for traders and investors to use deep hedging in the real world to manage real market risks.

This demonstrates the practicality of using AI-generated data in training and developing machine learning methods for derivatives pricing and dynamic hedging, and potentially for other financial applications as well. Although there is still more work to be done in optimizing the GAN in learning the underlying market data distribution and experimenting with different deep reinforcement learning algorithms, this proof-of-concept framework removes the need for market assumptions by learning directly from data. Since this system is model-free and data-driven, there is great potential for using GANs and deep reinforcement learning to hedge risks in other asset classes as well. The multi-agent deep reinforcement learning and GAN-based market simulation system provides a flexible and scalable framework for investors to systematically hedge a portfolio of diverse assets in a complex market with various risks, frictions, and constraints using historical data.

Chapter 4

Potentials of Reinforcement Learning and GANs in Finance

4.1 Identifying Statistical Arbitrage Opportunities

In Chapter 3, we explored the implementation of a GAN-enabled deep reinforcement learning framework to price and hedge options. We can adapt this framework to find instances of options mispricing by identifying under-valued and over-valued options in the market and applying statistical arbitrage trading strategies.

4.1.1 Capturing Mean Reversion with GANs

An essential component of statistical arbitrage strategies is based on the mean-reverting properties of financial time series to take advantage of price inefficiencies and divergence. In a perfectly efficient market, it would be difficult to implement consistent statistical arbitrage strategies because all available information is reflected in the prices of financial instruments and there would be no price inefficiencies, as stated by the efficient market hypothesis (EMH). Although there is a degree of market efficiency, real markets tend not to be perfectly efficient due to various biases and irrationality exhibited by traders. This results in various asset mispricings that can be arbitraged by using properties such as mean reversion. Although the term statistical arbitrage implies there is no risk to the strategy, there is actually a significant amount of risk when deploying statistical arbitrage strategies in the real markets as price movements are often unpredictable. In order to develop robust statistical arbitrage strategies, simply backtesting on historical data may not be sufficient to confirm the strategy's performance, as previously discussed in Chapter 1.

The risks present in traditional statistical arbitrage methods can be addressed by using reinforcement learning and generative adversarial networks. As explored previously in Chapter 2, GANs have the ability to learn and replicate the statistical properties that empirical financial data exhibit. In the GAN-generated data, there are also patterns of price divergences in a wide variety of market scenarios. To make statistical arbitrage models and strategies more robust, it is possible to optimize and evaluate those models and strategies on AI-generated data with the same mean-reverting properties as historical data [22]. Furthermore, as statistical arbitrage strategies are highly subject to market risks such as changing market conditions, a reinforcement learning framework can potentially improve trading strategies. Since model-free deep reinforcement learning, combined with GAN-based market simulation, has the ability to quickly adapt to different market scenarios, it may be a powerful tool for learning the optimal statistical arbitrage policy compared to statistical models.

4.1.2 Improving Pairs Trading

One popular type of statistical arbitrage strategy is pairs trading which relies on the assumption of mean reversion between two highly-correlated assets. Often, the difficult part of implementing a pairs trading strategy is identifying the correlated assets to simultaneously trade. The GAN-based approach in this study primarily focused on generating univariate financial time-series data that have similar properties to real data. However, this method can be extended to generating multi-variate time-series data that behave similarly to real multi-variate time-series. By simulating large amounts of time-series data of various correlated candidates, the strength of the correlations can be evaluated. Using this approach also provides the benefit of evaluating a pairs trading strategy's robustness to different market scenarios instead of a simple backtest on historical data, which may provide biased results.

Furthermore, there are often various financial and economic constraints in deploying trading strategies. As it may be difficult to account for these factors and large amounts of market data using traditional methods, it may be possible to implement reinforcement learning methods to make this framework more appropriate for realistic market conditions. For example, it is possible to use the described generative AI framework to generate data in a synthetic market environment and allow a deep reinforcement learning agent to learn the optimal policy for trade execution timing and position sizing. This framework would allow the flexibility to include any constraints while trying to optimize an objective function. Furthermore, this reinforcement learning-based strategy would become more robust when trained on realistic synthetic market data.

4.2 Developing Robust Risk Modeling Frameworks

4.2.1 Stress Testing using GAN-Based Market Simulation

The ability to simulate the possibilities and consequences of extreme market conditions is crucial for risk management. As the future is always unpredictable, Monte Carlo simulations have been widely adopted for simulating possible market scenarios. However, as previously discussed in Chapters 1 and 2, Monte Carlo simulations have limitations in simulating and representing market data. Instead, GANs are more capable of capturing true market dynamics by learning directly from historical market data. A potential application of GANs in risk management practices is to use GAN-based market simulations in stress testing and market risk assessment. GANs could be used to generate scenarios for extreme market conditions, such as severe stock price declines or sharp increases in interest rates. The consequences of market crashes are often unpredictable and hard to model with simple probabilistic simulations, but extreme market conditions can be emulated by GANs learning from, for example, market data during the 2008 financial crisis, or the COVID-19 pandemic. The market simulations generated by the GAN could then be used to stress test a portfolio and see how it would perform under these market conditions.

4.2.2 Value-at-Risk using GAN-Based Approach

A commonly used risk model to assess portfolio tail risks is Value-at-Risk (VaR) which involves implementing a confidence interval on return data to gauge return risks. The three most common approaches to building VaR models include historical simulation, delta-normal method, and Monte Carlo simulation. However, both delta-normal and Monte Carlo simulations are parametric methods that require the assumption about underlying return distributions. As previously discussed in Chapter 1, these parametric methods have limitations and may sometimes be inaccurate in modeling asset returns. For example, the delta-neutral method assumes log returns are normally distributed and estimates a normal distribution using this assumption, and Monte Carlo simulations require assumptions about underlying price movement dynamics. The historical simulation method makes use of past market data to build a confidence interval. However, also previously discussed in Chapter 1, historical data has limitations since it is scarce and doesn't capture enough market scenarios.

The benefit of using market simulations is the wide range of market scenarios that are generated. Instead of using traditional Monte Carlo methods, the GAN-based market simulation proposed in Chapter 2 can be used to generate various paths of asset returns. As the GANs are trained on historical market data and try to learn the underlying distribution, GAN-based market simulations carry the benefits of both the historical and Monte Carlo simulation methods. The AI-generated data will not only resemble real market data and have similar properties, but the GAN is able to generate an abundance of data, as shown in Figure 2-7. Augmenting traditional risk management tools like Value-at-Risk can potentially make the models more robust to changing market conditions and yield more precise results.

Chapter 5

Conclusion

5.1 Summary of Analysis

This research explores how combining artificial intelligence and machine learning methods fueled by data can devise and improve trading and hedging strategies from risk-neutral methods. Deep reinforcement learning is a robust method that can learn the optimal strategy to perform various tasks given different circumstances or market scenarios. It is the technology behind AlphaGo, the A.I. that beat the world Go champion, and is also the backbone to many more amazing applications of A.I. Whereas traditional methods rely on statistical models and stochastic processes to price and hedge options, this study explores a method that relies on empirical data and reinforcement learning algorithms to learn the optimal strategy. This data-driven approach allows the algorithm to learn the specific data-generating process instead of relying on many assumptions about the market and computing options greeks.

In order to train and evaluate this data-driven framework, it is necessary to have an abundance of realistic market data available that can expose the model to various market scenarios and evaluate its robustness to these scenarios. A popular method for testing trading strategies is using historical market data to train a trading strategy and evaluate its performance, known as backtesting. However, this method has many downsides that may bias the results of the test. First, backtesting provides limited data limited to the available market data recorded and the historical events that have occurred. This limitation may not fully be able to evaluate the deep reinforcement learning-based strategy's robustness to new market scenarios. Second, backtesting does not account for the market's response to the execution of the trading strategy being tested. Realistically, when an order is made, there may be a market response by other traders, which is not captured in historical data. These backtesting drawbacks can bias trading strategy evaluation results, and it may not be the case that the strategy will perform well in the future.

To address this issue, the first part of this research explores and implements a systematic data-driven approach to generate realistic synthetic market data from historical data. This method utilizes generative adversarial networks (GANs) to perform adversarial learning to generate new data that looks exactly like real data. This consists of two agents — a generator and a discriminator. The generator takes as input real historical market data and generates new fake data. The discriminator tries to distinguish the generator's output from the real data. Eventually, the generator can learn how to trick the discriminator by generating new realistic market data that can be used for training and testing trading strategies.

The second part of the research focuses on integrating the first part of deep hedging and deep reinforcement learning algorithms with the second part of GAN-based market simulations. The synthetically generated market data is used to train the algorithms by exposing them to a wide variety of market scenarios, representative of those that occurred in the past, as well as those that may occur in the future given the right circumstances. This will improve the robustness of the algorithm by allowing it to learn to adapt to different economic circumstances. The results of this GAN-based deep reinforcement learning framework are compared to traditional models like the Black-Scholes model. We conclude by discussing the efficacy of this systematic approach to developing and testing various data-driven trading strategies through machine learning approaches.

5.2 Results and Discussion

The goal of this study is to explore data-driven machine learning methods that can outperform traditional statistical methods in simulating market data and devising hedging strategies. In Chapter 2, the statistical properties and attributes of GAN-generated time-series market data closely matched those of real market data relative to a parametric Monte Carlo simulation. Based on the similarity of GAN-generated data to real market data relative to Monte Carlo methods, it may be appropriate to use GANs for market simulation purposes. Subsequently, in Chapter 3, deep reinforcement learning methods were explored to price and hedge derivatives in the GAN-based market environment. Based on the distribution of PnLs and volatilities of hedged returns in the two types of market environments, it seems that GANs are able to produce more robust deep hedging agents in solving the hedging problem. There is more work to be done in improving and optimizing these methods into a fully deployable system, but the preliminary results are promising and show the efficacy of this framework in a practical market setting.

5.3 Future Work

This study has primarily focused on developing a systematic proof-of-concept framework to use GAN-generated data to train deep reinforcement learning methods. There is more work to be done for optimizing these methods to produce more precise and robust results and ultimately become deployable in a live trading environment. For example, overfitting is an issue that plagues all forms of machine learning, including the methods explored in this research. GANs can potentially overfit if the generator learns to memorize the market data instead of learning the underlying distribution. RL agents can also overfit by memorizing a specific market environment instead of learning a generalizable hedging policy. Overfitting can cause the system to break when market conditions change, which they often do. Optimizing the system to mitigate overfitting includes hyperparameter tuning, regularization, and other methods.

As GANs consist of deep neural networks, they can learn the complexities of high-dimensional data despite overfitting problems. It is possible to use GANs to simulate multivariate time series data, which appears very frequently in the context of finance. For example, GANs can learn the underlying dynamics of the market microstructure by being trained directly on historical order book data. This makes it possible to train a GAN to generate multi-dimensional time-series data of bid-ask prices. Furthermore, GANs can also handle high-frequency data and can perhaps learn the patterns and dynamics of high-frequency price movements and mimic that behavior in the data it generates. These applications have significant potential for market makers to develop and enhance robust automated trading strategies and market-making algorithms.

The deep hedging framework explored in this study is flexible and robust compared to traditional risk-neutral pricing and hedging methods. Whereas classic options pricing models represent a market with underlying returns following a specific stochastic process, the deep hedging framework can be applied to any sort of market model. Therefore, any type of asset and derivative, along with realistic market conditions such as transaction costs, liquidity risks, and other financial constraints, can be modeled using deep hedging to find the optimal pricing and hedging policy. This study primarily focused on the application to equity markets but can be adjusted to develop hedging strategies in currencies, commodities, fixed income, crypto, etc.

Neural networks and artificial intelligence are known for their black box-like features since many of their predictions are not explainable. This concept applies to GANs and deep reinforcement learning methods in general as well. However, interpretability can potentially reveal useful information about the factors driving the decisions and outputs of GANs and reinforcement learning agents. In the context of GANs, using explainability methods such as activation maps on the generator and discriminator networks can reveal the properties of financial data that the GAN looks at when discriminating and generating new data. This can potentially reveal some unique properties that are generally exhibited by stock returns. Furthermore, in the context of reinforcement learning, using explainability methods can reveal the decision-making process of the deep hedging agent and the factors contributing to its hedging policy. This can potentially reveal the most important market factors and conditions contributing to the hedging decisions made by the agent. Both of these insights may be valuable to traders and investors for analyzing the market and improving investment strategies.

5.4 Final Thoughts

This study explores how GANs can be used as an alternative non-parametric approach to simulate and generate market data as opposed to traditional parametric approaches such as Monte Carlo methods that rely on assumptions about underlying distributions. The systematic market simulation framework is used to train deep hedging agents to find the optimal options pricing and hedging policy. Deep reinforcement learning combined with GAN-based market simulations makes dynamic hedging strategies more robust and precise compared to traditional hedging techniques.

This study describes just one application of using generative AI to implement and improve other artificial intelligence tools used in solving financial problems. However, there are many more potential applications of these tools to be used in solving other problems in finance. Traditional methods have generally been adopted and practiced due to their simplicity. The increasing access to computing power and the growing abilities of artificial intelligence have enabled machine learning methods to become more accurate and robust, significantly outperforming traditional methods. As AI becomes an increasingly popular research topic among financial services firms, I am excited to see the future growth and applications of AI in the financial industry.

Bibliography

[1] Fischer Black and Myron Scholes. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81:637–654, 1973.

[2] Hans Bühler, Lukas Gonon, Josef Teichmann, and Ben Wood. Deep Hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.

[3] Steven Heston. A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options. *The Review of Financial Studies*, 6(2):327–343, 1993.

[4] Igor Halperin. QLBS: Q-Learner in the Black-Scholes(-Merton) Worlds. *arXiv preprint*, 2017.

[5] James M Hutchinson, Andrew W Lo, and Tomaso Poggio. A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks. *The Journal of Finance*, 49:851–889, 1994.

[6] Petter Kolm and Ritter Gordon. Dynamic Replication and Hedging: A Reinforcement Learning Approach. *The Journal of Financial Data Science*, 1(1):159–171, 2019.

[7] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv preprint*, 2014.

[8] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. *Proceedings of the 34th International Conference on Machine Learning*, PMLR 70:214–223, 2017.

[9] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. *arXiv preprint*, 2017.

[10] Olof Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint*, 2016.

[11] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series Generative Adversarial Networks. *NeurIPS Proceedings*, 2019.

[12] Mohammad Diqi, Marselina Endah Hiswati, and Adri Saputra Nur. StockGAN: robust stock price prediction using GAN algorithm. *International Journal of Information Technology*, 14:2309–2315, 2022.

[13] Magnus Wiese, Lianjun Bai, Ben Wood, and Hans Buehler. Deep Hedging: Learning to Simulate Equity Option Markets. *arXiv preprint*, 2019.

[14] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing*, 307:72–77, 2018.

[15] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[16] Hans Bühler, Baranidharan Mohan, and Ben Wood. Deep Hedging: from Theory to Practice, 2019.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv preprint*, 2013.

[18] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint*, 2017.

[19] Jiayi Du, Muyang Jin, Petter Kolm, Gordon Ritter, Yixuan Wang, and Bofei Zhang. Deep Reinforcement Learning for Option Replication and Hedging. *The Journal of Financial Data Science*, 2(4):44–57, 2020.

[20] Victor Storchan, Svitlana Vyetrenko, and Tucker Balch. MAS-GAN: Adversarial Calibration of Multi-Agent Market Simulators. 2020.

[21] Jay Cao, Jacky Chen, John Hull, and Zissis Poulos. Deep Hedging of Derivatives Using Reinforcement Learning. *The Journal of Financial Data Science*, 3(1):10–27, 2021.

[22] Hans Bühler, Phillip Murray, Mikko S. Pakkanen, and Ben Wood. Deep hedging: learning to remove the drift. *risk.net*, 2022.