End of study report

Abderrahim Namouh

Mention Sciences des données et de l'information (SDI)
Filière Métiers d'Analyse et d'Aide à la Décision (M2AD)

April 2024 - October 2024

School Tutors :
Filière : Sanaa Legendre
sanaa.legendre@centralesupelec.fr
Mention : Christian Bongiorno
christian.bongiorno@centralesupelec.fr

Company Tutor :
Mourad Jemal
morad.jemal@mazars.fr

Abstract

This project explores liquidity provision strategies for Uniswap v3, a leading decentralized exchange in the decentralized finance (DeFi) ecosystem. The study has three main objectives: (1) to conduct a comprehensive market analysis to gain insights into the behavior and dynamics of Uniswap v3, including historical market data related to liquidity pools, trade volumes, and liquidity concentration; (2) to develop an independent, locally implemented backtesting environment for evaluating liquidity provision strategies; and (3) to optimize these strategies using deep learning models.

To achieve these goals, we first performed an in-depth market analysis to understand liquidity patterns and trade dynamics. We then built a stress testing and backtesting system based on the Uniswap v3 protocol, allowing for dynamic evaluation of liquidity operations such as rebalancing and liquidity reallocation. Deep learning models were trained within this environment to identify strategies that maximize fee gains while minimizing risks like impermanent loss.

Our findings demonstrate that optimizing liquidity provision requires balancing the depth and width of concentrated liquidity ranges. The deep learning models outperformed baseline strategies, providing superior returns. This project contributes a robust tool for future stress testing and illustrates the potential of deep learning in enhancing liquidity provision strategies on decentralized exchanges.

Contents

1 Introduction

1.1 Background and Motivation

1.1.1 Overview of Decentralized Finance (DeFi)

Decentralized Finance, commonly known as DeFi, refers to a new financial system built on blockchain technology that operates without traditional intermediaries such as banks or brokers. DeFi leverages smart contracts—self-executing contracts with the terms of the agreement directly written into code—to facilitate financial transactions and services. These services include lending, borrowing, trading, and investing, all conducted in a decentralized manner.

The rise of DeFi marks a significant departure from conventional financial systems. By removing intermediaries, DeFi aims to provide a more open, transparent, and accessible financial system. This new paradigm offers several key advantages:

- **Transparency:** Transactions and smart contract codes are publicly accessible on the blockchain, allowing anyone to audit them. This openness fosters trust and accountability.
- **Accessibility:** DeFi services are available to anyone with an internet connection and a digital wallet, regardless of geographical location or socio-economic status. This inclusivity can potentially democratize access to financial services.
- **Efficiency:** Automated processes in DeFi reduce the need for manual intervention, thus lowering the costs and increasing the speed of transactions.
- **Interoperability:** DeFi protocols are often built on open-source platforms, allowing different applications to interact and integrate with one another seamlessly. This interconnected ecosystem enables innovative financial products and services.

DeFi has experienced exponential growth in recent years, driven by the increasing adoption of cryptocurrencies and blockchain technology. The total value locked (TVL) in DeFi protocols has surged, reflecting the growing trust and reliance on decentralized financial systems. As DeFi continues to evolve, it promises to bring further innovations and improvements to the financial sector, offering a compelling alternative to traditional financial systems.

1.1.2 Decentralized Exchanges (DEXs)

Decentralized Exchanges (DEXs) are a fundamental component of the DeFi ecosystem, enabling users to trade cryptocurrencies directly with one another without the need for centralized intermediaries. Unlike traditional centralized exchanges (CEXs) that operate through an order book and hold users' funds, DEXs empower users by allowing peer-to-peer transactions via smart contracts, enhancing security and control over assets.

Key Features of DEXs:

- **Automated Market Makers (AMMs):** DEXs like Uniswap and SushiSwap utilize AMMs instead of traditional order books. AMMs rely on mathematical formulas to price assets, enabling continuous liquidity through liquidity pools.
- **Liquidity Pools and Providers:** Users, known as liquidity providers (LPs), contribute pairs of tokens to liquidity pools. In return, they earn a portion of the trading fees proportional to their share of the pool, incentivizing the provision of liquidity.

Advantages of DEXs include:

- **Security:** By eliminating centralized control, users retain full ownership of their assets, mitigating risks associated with hacks or mismanagement often seen on centralized platforms.
- **Privacy:** DEXs generally do not require user identification (KYC), preserving privacy while still facilitating trades.
- **Censorship Resistance:** Since DEXs are decentralized and global, they are less vulnerable to censorship or regulatory interference.

However, DEXs also face significant challenges:

- **Liquidity:** Especially in early stages or for niche assets, liquidity can be low, leading to price slippage and higher transaction costs.
- **User Experience:** While improving, DEX interfaces are often less intuitive than CEXs, posing barriers to adoption for non-technical users.
- **Regulatory Requirements:** As DEXs operate without intermediaries, they have attracted scrutiny from regulators, particularly in relation to fraud, criminal activity, and money laundering. The absence of KYC (Know Your Customer) and AML (Anti-Money Laundering) processes on many DEXs has raised concerns about their use for illicit activities. Governments are exploring ways to impose regulatory frameworks that ensure compliance with financial laws while maintaining the decentralized nature of these exchanges.

Despite these challenges, DEXs such as Uniswap, SushiSwap, and Curve have gained widespread popularity as key players in DeFi, revolutionizing how trades and liquidity are managed.

1.1.3 Why Uniswap v3?

Uniswap v3 was chosen for this project not only because of its significant innovations in decentralized finance (DeFi) but also due to its alignment with Forvis Mazars' aim to anticipate upcoming European Union regulations in the crypto market and DeFi sector. By actively engaging with the most advanced and widely used decentralized exchange (DEX), Forvis Mazars plans to effectively meet clients' needs and be prepared to offer compliant and robust solutions as regulatory frameworks evolve.

This project gives potential clients a forward-looking analysis and a way to evaluate the strength of liquidity provision strategies within a regulatory context. Given Uniswap v3's importance and ongoing innovation, it is the optimal platform for such an exploration. Compared to earlier versions and other DEXs, Uniswap v3 offers a high level of capital efficiency and flexibility for liquidity providers, making it a more advanced platform for strategic liquidity management.

The introduction of concentrated liquidity allows liquidity providers to allocate capital within specific price ranges, maximizing capital efficiency and earning potential. This feature enables providers to customize their risk-reward profiles, which is particularly useful in volatile markets. Moreover, the ability to control liquidity distribution minimizes the risk of impermanent loss—a key concern for liquidity providers in decentralized exchanges that will be further discussed in this report.

Uniswap v3 also addresses the evolving needs of institutional participants in DeFi by offering features such as range orders, which are similar to limit orders in traditional exchanges. These attributes make Uniswap v3 especially suitable for complex and strategic liquidity management, reinforcing its selection as the ideal platform for testing advanced liquidity provision strategies in anticipation of upcoming regulatory changes.

1.2 Objectives

This project is centered around three key objectives, each addressing different aspects of liquidity provision on Uniswap v3:

1. Market Analysis:
The first objective is to conduct a comprehensive market analysis to gain insights into the behavior and dynamics of Uniswap v3. This includes studying historical market data related to liquidity pools, trade volumes, and liquidity concentration.

2. Development of a Stress Testing/Backtesting Environment:
The second objective is to create an independent, locally implemented backtesting environment for evaluating liquidity provision strategies on Uniswap v3. This environment will simulate a variety of price paths and provide dynamic monitoring of liquidity operations, including rebalancing and reallocation of liquidity. By organizing operations chronologically and tracking their impact over time, the environment will enable accurate and efficient evaluation of liquidity strategies under various market conditions.

3. Optimization of Liquidity Provision Strategies using Deep Learning:
The third objective is to optimize liquidity provision strategies through the application of deep learning models. These models will be trained within the developed backtesting environment to identify strategies that maximize capital efficiency while minimizing risks such as impermanent loss. The use of deep learning allows for sophisticated pattern recognition and predictive capabilities. The goal is to demonstrate the effectiveness of neural networks in enhancing liquidity strategies on Uniswap v3.

Together, these objectives aim to provide a robust framework for understanding market dynamics, stress testing liquidity provision strategies, and optimizing performance using advanced modeling techniques.

1.3 Report Roadmap

To guide the reader through the progression of this report, we present a roadmap that highlights the structure and flow of the work. This will allow the reader to see the full picture and understand the logical progression from theoretical foundations to practical implementation, analysis, and conclusions.

- Uniswap v3 Theory and Concepts:
We begin by introducing the foundational theory of Uniswap v3, which is essential for understanding the more complex features and metrics discussed later in the report. This section covers the key mechanisms of Uniswap v3, including concentrated liquidity pools, the constant product market maker (CPMM) formula, and how these innovations improve upon previous versions (v1 and v2). The goal is to ensure that the reader has a solid grasp of Uniswap v3's theoretical framework before diving into the technical analyses and optimizations.

- Market Analysis:
With a thorough understanding of Uniswap v3 established, we proceed to the first objective: performing a comprehensive market analysis. This analysis explores historical market data to gain insights into the trends and behaviors that influence liquidity pools, trade volumes, and liquidity concentration on Uniswap v3. By understanding these market dynamics, we set the foundation for testing and refining liquidity provision strategies.

- Backtesting and Stress Testing Environment Development:
The next section presents the second objective of this project: the creation of a backtesting and stress testing environment for Uniswap v3. We describe the methodology behind the development of this environment, explain its key features, and demonstrate how it simulates liquidity operations (e.g., mints, burns, swaps) across

different market conditions. The reader will follow the steps we took to validate the environment by comparing simulated results with historical data, ensuring the accuracy of our model.

- Strategy Optimization:
After establishing the backtesting environment, our focus shifts to the third objective: optimizing liquidity provision strategies using deep learning models. This section goes into the methodology of our optimization approach, detailing the rationale behind the selection of neural network architectures, the process of training the models, and the methods used for validation. We compare the performance of the optimized strategies against baseline approaches, demonstrating how deep learning models can enhance liquidity provision by improving capital efficiency and reducing exposure to risks such as impermanent loss.

- Future Work and Conclusion:
In the final section, we outline how this research can serve as a foundation for future work in decentralized finance. We suggest several avenues for improvement in both the backtesting environment and the strategy optimization process. These include refining the models to account for transaction costs like gas fees, incorporating more sophisticated market dynamics, and extending the analysis to other decentralized exchanges beyond Uniswap v3. Additionally, we discuss how future research could explore alternative deep learning techniques and apply them to more complex liquidity provision scenarios. This work offers a robust starting point for ongoing exploration in optimizing decentralized finance strategies.

2 Uniswap v3 Overview

2.1 Definition

Uniswap is a decentralized exchange (DEX) protocol that operates on the Ethereum blockchain. It facilitates the swapping of ERC-20 tokens directly from user wallets without the need for a central intermediary. Uniswap employs an Automated Market Maker (AMM) model to enable continuous and decentralized trading through liquidity pools.

2.2 Key Features of the Uniswap Platform

- **Decentralization:** Uniswap operates in a completely decentralized manner, meaning that trades are executed directly between users (peer-to-peer) through smart contracts without the need for a centralized exchange or intermediary.

- **Automated Market Maker (AMM):** Uniswap uses the AMM model, which relies on mathematical formulas to price assets. The most common formula is the constant product formula $x \times y = k$, where x and y are the quantities of two tokens in a pool, and k is a constant. This model allows for continuous liquidity and pricing without needing an order book.

- **Liquidity Pools:** Users, known as liquidity providers (LPs), can deposit pairs of tokens into liquidity pools. In return, they earn a portion of the trading fees generated by the pool. This system incentivizes users to provide liquidity and helps maintain a robust and liquid market.

- **User-Friendly Interface:** Uniswap offers a simple and intuitive interface that allows users to swap tokens, add or remove liquidity, and track their transactions. This ease of use has contributed to its widespread adoption in the DeFi space.

- **Integration with Ethereum Wallets:** Uniswap integrates seamlessly with popular Ethereum wallets like MetaMask, enabling users to connect their wallets and trade directly from their wallets. This integration ensures a smooth and secure trading experience.

Uniswap has become a foundational component of the DeFi ecosystem, providing a reliable and efficient platform for decentralized trading. Its innovative use of AMMs and liquidity pools has set a new standard for DEXs, promoting greater accessibility and participation in the financial markets.

2.3 Mechanism: Uniswap v1 & v2

Uniswap has evolved significantly since its inception, with each version introducing new features and improvements to enhance functionality and user experience.

Launched in November 2018, Uniswap v1 was the first implementation of the automated market maker (AMM) model. It introduced the constant product formula, which allows users to trade tokens directly from their wallets by interacting with liquidity pools. This version laid the foundation for decentralized trading by enabling continuous liquidity without the need for order books.

2.3.1 The Constant Product Market Maker (CPMM) Formula

The core of Uniswap v1's AMM model is the constant product formula:

$$x \times y = k$$

Here, x and y represent the reserves of two different tokens in the liquidity pool, and k is a constant that is used to determine the current exchange rate. While in a world with no fees, the product remains unchanged, in Uniswap, due to adding received fees to the pool, the product increased overtime.

2.3.2 How It Works

- **Liquidity Provision:** Users, known as liquidity providers (LPs), add liquidity to the pool by depositing a certain amount of both tokens, determined by the current state of the liquidity pool. In the case where a liquidity provider is the first one to enter the pool, he has the freedom of depositing any amount, thus setting a price of his choice. For example, an LP might deposit 100 units of Token A and 200 units of Token B, establishing initial reserves.
  Note: If the initial price is set far from the current market price, the initial liquidity provider would be exposed to the risk of arbitrage opportunities and impermanent loss. This makes initial liquidity providers stay in accordance with the market.

- **Fees:** A fixed fee of 0.3% is defined and paid by the trader to LPs contributing to the pool in the token provided by the trader. The fees are deduced from the provided amount and the remaining is used according to the constant product model. Fees are added to the pool, which makes the product of the reserves increase overtime.

- **Trading Mechanism:** When a user wants to trade Token A for Token B, they add Token A to the pool and remove a corresponding amount of Token B, adjusting the reserves to maintain the constant product.

2.3.3 Example

Let's consider an example where the initial reserves in a liquidity pool are 1000 units of Token A and 2000 units of Token B. The constant k is:

$$1000 \times 2000 = 2000000$$

Initial State: The pool starts with:

$$x = 1000, \; y = 2000, \; k = 2000000$$

Trade Scenario: A user wants to trade 100 units of Token A for Token B. In Uniswap v2, a 0.3% fee is applied to the trade. Therefore, the effective amount of Token A added to the pool is:

$$\delta x_{effective} = 100 \times (1 - 0.003) = 100 \times 0.997 = 99.7$$

The effective amount of Token A added for calculating the output is:

$$x_{effective} = 1000 + 99.7 = 1099.7$$

To maintain the constant product, the effective new reserve of Token B ($y_{effective}$) should satisfy:

$$1099.7 \times y_{effective} = 2000000$$

Solving for $y_{effective}$:

$$y_{effective} = \frac{2000000}{1099.7} \approx 1818.68$$

The user receives:

$$2000 - 1818.68 \approx 181.32$$

units of Token B.

Thus, after the trade and accounting for the 0.3% fee, the pool now has:

$$x' = 1099.7 + 0.3 = 1100 \quad \text{units of Token A and} \quad y' = 1818.68 \quad \text{units of Token B}$$

The additional 0.3 units of Token A (from the fee) remain in the pool, slightly increasing the total reserves and thus the constant product k over time.

2.3.4 Implications

- **Price Impact:** The larger the trade relative to the pool's size, the more significant the price impact. This feature discourages large trades that would cause drastic price changes, promoting stability.
- **Continuous Liquidity:** The constant product formula ensures that there is always some amount of both tokens in the pool, providing continuous liquidity for traders.

- **Slippage:** Traders experience slippage, which is the difference between the expected price of a trade and the actual price. Slippage increases with the size of the trade relative to the pool's reserves.

Uniswap v1's constant product formula provided a simple yet effective mechanism for decentralized trading, setting the stage for future innovations in the Uniswap protocol.

2.3.5 Enhancements in Uniswap v2

Uniswap v2 brought several key improvements, making the platform more efficient and versatile. It enabled direct token swaps without the need for an intermediary, reducing transaction complexity. Flash swaps were introduced, allowing users to access liquidity temporarily, which opened up new possibilities for decentralized finance (DeFi) operations. The price tracking system was improved to provide more reliable data, and a governance-controlled fee mechanism was added, offering a potential revenue stream for future development and ecosystem support.

Uniswap v1 and v2 established the fundamental principles and mechanisms that made Uniswap a cornerstone of the DeFi ecosystem. These versions introduced and refined the AMM model, enhancing the user experience and expanding the functionality of decentralized exchanges.

2.4 Uniswap v3

Uniswap v3 represents a significant advancement in the decentralized finance ecosystem, introducing several innovations that enhance capital efficiency, liquidity provision, and user experience compared to Uniswap v2.

1. Concentrated Liquidity: Uniswap v3 introduces the concept of concentrated liquidity, allowing liquidity providers (LPs) to allocate their liquidity within specific price ranges. This enhances capital efficiency by enabling LPs to focus their funds where they expect the most trading activity, rather than distributing it uniformly across the entire price spectrum as in Uniswap v2.

2. Flexible Fee Tiers: Uniswap v3 offers multiple fee tiers (0.05%, 0.3%, and 1%) for different pools, allowing LPs to choose the fee structure that best suits their risk tolerance and expected volatility of the asset pair. This is an improvement over the single fee tier in Uniswap v2. Here we present some :

2.4.1 Concentrated Liquidity Pool in Uniswap V3

Uniswap V3 introduces the concept of concentrated liquidity, enhancing capital efficiency and flexibility for liquidity providers (LPs). In this model, LPs specify the price range $[S_l, S_u]$ within which they wish to provide liquidity, concentrating it where they believe it will be most effective. This section explains the mechanics of a concentrated liquidity pool, introduces the necessary notation, and derives the real and virtual amounts.

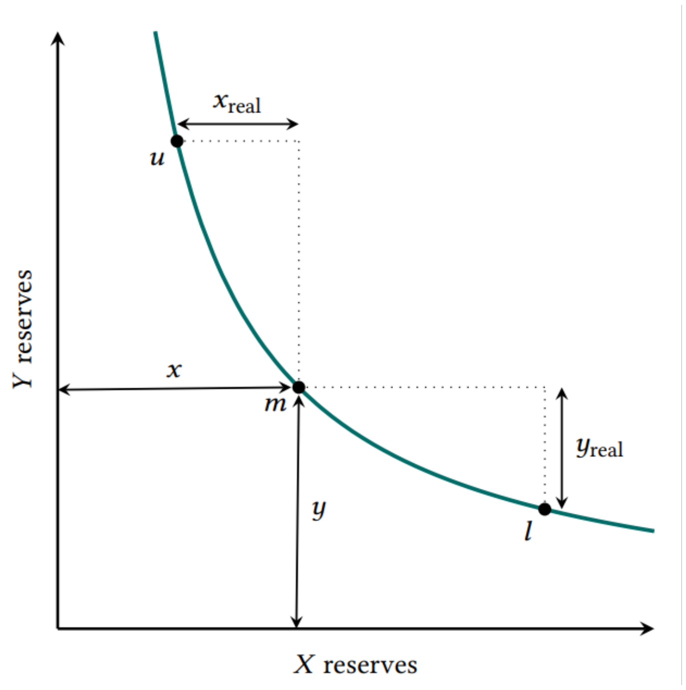

Figure 2: The curve of a Constant Product Market Maker (CPMM), showing both real and virtual liquidity in the framework of Uniswap v3.[3]

Notations:
In the following section, we'll use the notation bellow, the definition of each of these notions will be given progressively :

- x: Virtual reserve of token X.
- y: Virtual reserve of token Y.
- x_{real}: Real reserve of token X.
- y_{real}: Real reserve of token Y.
- L: Liquidity of the pool between ticks T_l and T_u.
- S: Marginal price, given by $S = \frac{y}{x}$.
- $[S_l, S_u]$: The price range within which liquidity is concentrated.

Virtual and Real Reserves:
In a concentrated liquidity pool, the concept of virtual reserves is utilized to describe the pool's behavior within a specific price range $[S_l, S_u]$. The virtual reserves act as if the liquidity in the entire pool matches that of the current price range, allowing the constant product formula to be applied.

The constant product formula for virtual reserves is:

$$x \cdot y = L^2$$

where L is the liquidity in the pool.

The marginal price S is given by:

$$S = \frac{y}{x}$$

From this, we can derive the virtual reserves in terms of liquidity L and price S:

$$x = \frac{L}{\sqrt{S}} \quad \text{and} \quad y = L \cdot \sqrt{S}$$

Deriving Real Reserves:
The real reserves can be obtained from the virtual reserves by considering the boundaries of the price range $[S_l, S_u]$.

For token X:

$$x_{real} = x - \frac{L}{\sqrt{S_u}} = \frac{L}{\sqrt{S}} - \frac{L}{\sqrt{S_u}}$$

For token Y:

$$y_{real} = y - L \cdot \sqrt{S_l} = L \cdot \sqrt{S} - L \cdot \sqrt{S_l}$$

Thus the CPMM equation becomes :

$$\left(x_{real} + \frac{L}{\sqrt{S_u}}\right)\left(y_{real} + L\sqrt{S_l}\right) = L^2$$

Ticks and Initialization:

In Uniswap V3, the concept of ticks is crucial for managing the price ranges within which liquidity is concentrated. Each tick represents a specific price point in the pool's price range.

Ticks are defined at every integer exponent of 1.0001, and the price corresponding to tick i is given by:

$$S(i) = 1.0001^i$$

Each tick is 0.01% (one basis point or *bps*) away from its neighboring tick.

An initialized tick is a tick used by at least one position. Not every tick can be initialized in a pool. Ticks are only initialized if their indexes are divisible by the pool's predefined tick spacing ts. The tick spacing is a parameter that determines how granular the price ranges can be for liquidity positions.

When a liquidity provider (LP) chooses to provide liquidity, they select a price range $[S_l, S_u]$. The liquidity is then allocated within this range and is active only when the market price falls within these bounds.

Between any two adjacent initialized ticks T_l and T_u, the liquidity behaves as if the entire pool's liquidity is concentrated within this range. This allocation ensures that liquidity is effectively utilized where it is most needed, enhancing capital efficiency.

Consider a pool with a tick spacing of 10. This means only ticks with indexes that are multiples of 10 can be initialized. For example, ticks at $i = 0, 10, 20, \dots$ can be valid initialized ticks.

If an LP wants to provide liquidity in the range from S_l to S_u, they must choose these bounds from the set of authorized ticks. Suppose the current price S is between ticks 30 and 40, and an LP provides liquidity between ticks 20 and 50. The pool will use the virtual reserves and apply the constant product formula within this range, ensuring that liquidity is efficiently managed.

Transaction Fees:
Collected transaction fees are distributed pro-rata to the liquidity providers who have deposited liquidity at the price the asset pairs are trading at. Uniswap V3 features different pool fee tiers ($f \in \{0.01\%, 0.05\%, 0.3\%, 1\%\}$), allowing for varying transaction fees based on the relative price volatility between the two cryptocurrencies in the pool.
Moreover, fees are not automatically added to the LPs positions, instead, a tracker of accumulated fees in each of the two tokens is implemented to track the amount of uncollected fees.The LP can retrieve them at any time.

Example of a Trade:
Consider a liquidity pool for tokens X and Y with virtual reserves x and y at the current price. A trader wishes to exchange δx of token X for token Y. The amount of Y received, δy, is given by:

$$\delta y = y - \frac{x \cdot y}{x + (1-f)\delta x} = y\left(1 - \frac{(1-f)\delta x}{x + (1-f)\delta x}\right)$$

where f is the transaction fee charged relative to the input amount δx.

As long as the pool's price does not move across an initialized tick, the same formula holds in Uniswap V3. If the price does cross a tick, the trade executes at the available liquidity depth until it reaches the next initialized price tick, continuing this process until the entire trade input is swapped.

3 Uniswap v3 Analysis

3.1 Data Retrieval

For this project, data was retrieved from the USDC/ETH 0.05% pool on Uniswap v3 using Mazars's own Erigon Node. A custom Python script was developed to interact with the node, leveraging the Web3, Pandas, and Numpy libraries. The script processed the data in batches, filtering logs for relevant events such as swaps, mints, burns, and collects. The retrieved values were transformed from their original formats *(e.g., sqrtX96, uint)* into usable forms, considering the token's decimals for accurate calculations.

The events collected include:

- **Swaps:** Include information about the exchange of tokens, including the amounts swapped, price after the swap *(sqrtPriceX96)*, liquidity, and the tick at which the swap occurred.
- **Mints:** Data on liquidity added to the pool, detailing the amount of liquidity and the price range (ticks) within which the liquidity is added.
- **Burns:** Information on liquidity removed from the pool, including the amount of token0 and token1 withdrawn, and the corresponding price range.
- **Collects:** Data on the fees collected by liquidity providers, showing the amounts of token0 and token1 withdrawn as fees and the position's underlying amounts that generated the fees.

For an exhaustive list on data stored in Uniswap v3 pool events, refer to the official documentation here.

3.1.1 Liquidity Distribution Construction

To reconstruct the liquidity distribution for a specific point in time, we first identified the block number corresponding to the time of interest using a built-in Web3 function. This allowed us to access the liquidity data as it existed in the Uniswap v3 USDC/ETH 0.05% pool at the specified block.

The liquidity distribution construction began by determining the active tick, which corresponds to the tick where the current price resided at that moment. The liquidity at this active tick was retrieved, representing the liquidity available around the current price.

Once the active tick was identified, we traversed through all ticks to the left and right of the active tick, jumping from one tick to another based on the tick spacing parameter. This parameter, predefined by Uniswap v3, represents the minimum price difference between two ticks.

For each tick, we checked if it was initialized. If the tick was initialized, we used the ΔL parameter to update the liquidity at that tick. ΔL represents the change in liquidity at a specific tick when the price crosses it, from the left to the right, allowing us to dynamically adjust the liquidity as we moved further away from the active tick.

The traversal continued both to the left and right of the active tick, calculating liquidity changes until we reached predefined minimum and maximum ticks. These bounds were selected based on the specific analysis needs, allowing us to control how far we needed to extend the liquidity distribution.

This approach provided a full liquidity distribution across a range of ticks, giving us insight into how liquidity was concentrated around the current price and how it dissipated as we moved further from the active tick. The ability to stop at a certain tick range gave flexibility in focusing on areas with more liquidity concentration, or alternatively, exploring the broader liquidity landscape.

Placeholder for Graphs:

- Liquidity distribution graph showing how liquidity is concentrated around the active tick and how it changes across the tick range.
- Graph illustrating liquidity dynamics as the price moves through different ticks over time.

3.2 Impermanent Loss

Impermanent loss arises when a liquidity provider (LP) supplies two tokens, typically denoted as X and Y, into a decentralized exchange pool. The LP is exposed to changes in the price, as well as in the composition of his position due to swaps, leading to potential loss in value compared to holding the tokens outside the pool.

Let:

- x_0 and y_0 be the initial quantities of tokens X and Y deposited into the pool.
- S_0 be the price at the time of deposit.
- S_t be the price ratio at a later time t, when the LP considers withdrawing the tokens.
- x_t and y_t be the quantities of tokens X and Y in the LP's position at time t.

3.2.1 Value of Holding the Tokens

If the LP had chosen to simply hold the tokens instead of providing liquidity, the value of the holding position, denoted $V_{hold}(t)$, would be:

$$V_{hold}(t) = x_0 \cdot S_t + y_0$$

This represents the value of the tokens X and Y based on the new price S_t at time t, assuming no liquidity was provided.

3.2.2 Value of the LP Position in the Pool

By providing liquidity, the LP's token quantities change as the price fluctuates. The value of the LP's position in the pool at time t, denoted $V_{position}(t)$, is:

$$V_{position}(t) = x_t \cdot S_t + y_t$$

This accounts for the updated quantities x_t and y_t, which adjust due to price movements and trading activity in the pool.

3.2.3 Defining Impermanent Loss

Impermanent loss (IL) is defined as the relative difference between the value of the LP's position had they held the tokens and the value of the position after providing liquidity. The formula for impermanent loss is:

$$IL(S_0, S_t) = \frac{V_{position}(t) - V_{hold}(t)}{V_{hold}(t)}$$

Substituting the values:

$$IL(S_0, S_t) = \frac{x_t \cdot S_t + y_t}{x_0 \cdot S_t + y_0} - 1$$

Impermanent loss is a percentage loss relative to simply holding the tokens. It becomes more significant as the price ratio S_t deviates from the initial price S_0. Importantly, impermanent loss is only realized if the LP withdraws liquidity when the price ratio has changed from its initial value.

3.2.4 Relevance in Evaluating Liquidity Provision Strategies

Impermanent loss is a key factor in evaluating the profitability of liquidity provision strategies, especially in volatile markets. In Uniswap v3, liquidity providers can concentrate their liquidity within specific price ranges, increasing their exposure to impermanent loss if the price moves outside their selected range. This makes tracking impermanent loss crucial when analyzing liquidity provision performance.

By balancing the fees earned with the potential impermanent loss, LPs aim to develop strategies that minimize their exposure to this loss while maximizing fee income. The concentrated liquidity feature of Uniswap v3 adds another layer of complexity, as liquidity outside the active price range no longer earns fees but remains exposed to impermanent loss.

In the following sections, we will analyze how different liquidity provision strategies handle impermanent loss and the trade-offs involved in managing liquidity across various price ranges.

3.2.5 Why Impermanent Loss is Always Negative

Impermanent loss is always negative because, when the price of the assets in a liquidity pool changes, the automated market maker (AMM) rebalances the LP's position. This rebalancing results in the LP holding more of the asset that has decreased in value and less of the asset that has increased. If the LP had simply held the tokens outside the pool, the only change in the LP's would be due to price movements. Mathematically, the value of the LP's position in the pool is always less than the value of holding the tokens, as shown by the impermanent loss formula:

$$IL(S_0, S_t) = \frac{V_{position}(t)}{V_{hold}(t)} - 1$$

While the complete mathematical proof is possible, the complexity of Uniswap v3 makes it less intuitive. As long as the price deviates from the initial value S_0, impermanent loss is negative. The more the price diverges, the greater the loss. For further detailed analysis on the matter, we invite you to check [3].

3.2.6 Trade-off Between Fee Generation and Position Width

We explore how liquidity providers (LPs) can optimize fee generation on Uniswap v3 by balancing position concentration with the risk of positions becoming inactive. While concentrated positions can yield higher fees due to increased capital efficiency, they are more likely to become inactive if the market price moves outside the specified range. This analysis aims to identify an optimal position width (α) that maximizes fee earnings while minimizing risk.

Fees and Position Width :

In Uniswap v3, the fees earned by an LP are inversely proportional to the position width (α) and directly proportional to the time the position remains active within its specified price range:

$$F \propto \frac{T_{\text{in}}}{\alpha}$$

where:

- F: Fees earned.
- T_{in}: Time the price remains within the liquidity range.
- α: Width of the liquidity position.

Price Modeling Using Geometric Brownian Motion :

To analyze the impact of position width on time in range, we model the asset price using a Geometric Brownian Motion (GBM):

$$dS_t = \mu S_t\, dt + \sigma S_t\, dW_t$$

where:

- S_t: Asset price at time t.
- μ: Expected return (drift term).
- σ: Volatility of the asset.
- dW_t: Increment of a Wiener process.

Simulation parameters include initial price S_0, estimated drift μ, volatility σ, time horizon T_{total}, and a large number of simulations to ensure statistical significance.

Impact of Position Width on Time in Range :

We define the relative time in the money (T_{rel}) as:

$$T_{\text{rel}} = \frac{T_{\text{in}}}{T_{\text{total}}}$$

To assess how position width α affects T_{rel}, we:

- Vary α to represent different position widths.
- Simulate price paths using GBM for each α.
- Calculate T_{rel} for each simulation.
- Compute the average T_{rel} for each α.

Findings:
Simulations show that:

- **Narrower Ranges** (α decreases):
  - Higher potential fees due to increased capital efficiency.
  - Lower T_{rel} as the price is more likely to exit the range.
- **Wider Ranges** (α increases):
  - Lower potential fees as capital is spread over a wider range.
  - Higher T_{rel} with increased likelihood of the price remaining within the range.

Identifying the Optimal Position Width :

By analyzing the ratio T_{rel}/α, we identify the position width that maximizes fee earnings per unit width. The optimal α balances earning higher fees and maintaining an active position.

Graphical Representation

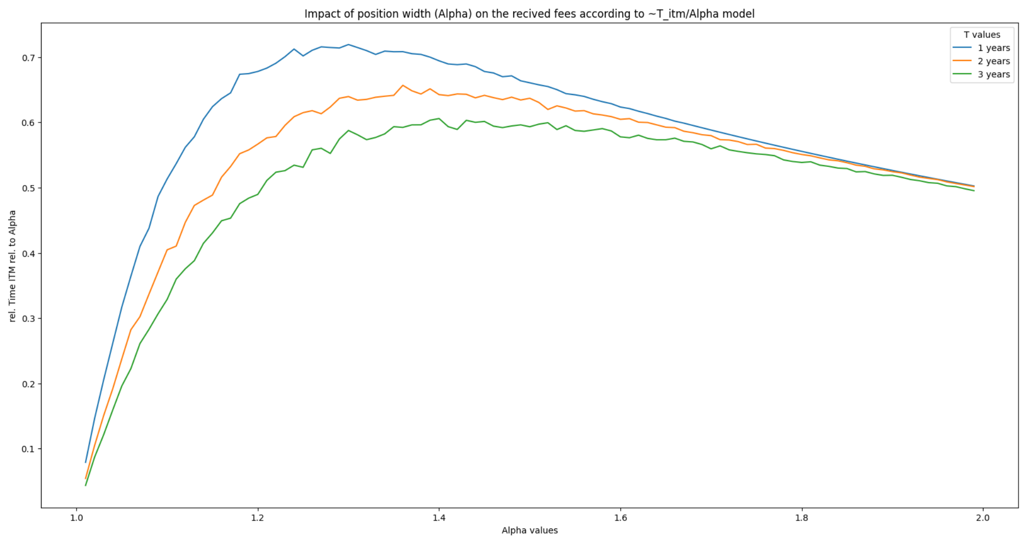

Figure 3: Ratio of T_{rel}/α vs. Position Width (α)

Discussion: Implications for Liquidity Providers

- **Strategy Optimization**: LPs can select a position width that aligns with their fee generation goals and risk tolerance.
- **Risk Management**: Understanding the relationship between position width and time in range helps mitigate the risk of positions becoming inactive.
- **Dynamic Adjustments**: LPs should adjust their position width based on changing market conditions.

Limitations

- **Model Assumptions**: GBM assumes constant drift and volatility, which may not hold in real markets.
- **Simplifications**: The model does not consider rabalancing costs or external market factors. This adds an additional layer of complexity for liquidity providers.

Conclusion :

This analysis illustrates how position width affects fee generation and position activity in Uniswap v3. Identifying an optimal α enables LPs to enhance their strategies to maximize returns while managing risks.

3.3 Market Analysis

A deeper understanding of the market dynamics within Uniswap v3, as well as decentralized exchanges (DEXs) in general, requires a thorough investigation of key market parameters. These parameters include the evolution of price and liquidity, alongside user behavior during both normal and stressed periods. In this section, we present key insights from our comprehensive market analysis.

3.3.1 Evolution of the volume of trades

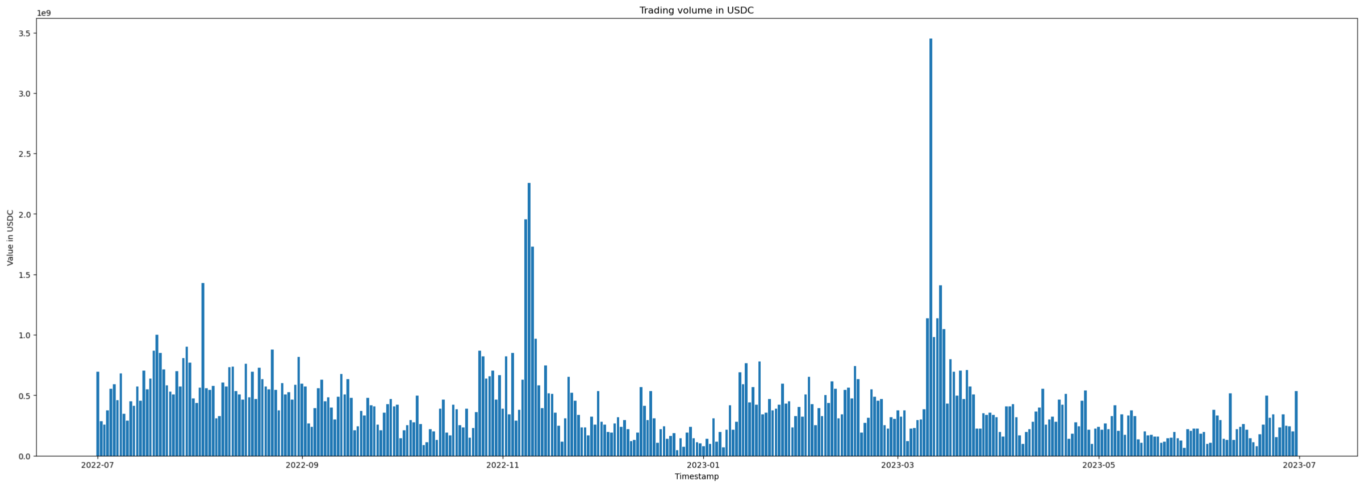

Figure 4: Evolution of Trade Volumes

Figure 4 depicts the evolution of trade volumes on Uniswap v3 over time. Notably, we observe distinct spikes in trading volume during periods of significant market stress, such as the collapse of FTX in November 2022 and the shutdown of Silvergate Bank in March 2023. These sudden surges in trading volume are a direct response to market participants' efforts to react rapidly and reposition in times of heightened uncertainty. This behavior is consistent with what is typically seen in centralized exchanges (CEXs), further validating the interconnectedness between DEXs and traditional financial markets. Moreover, the ability of Uniswap to handle these spikes in volume without compromising operational efficiency underscores the robustness of its liquidity provision mechanism.

3.3.2 Evolution of Mean Active Liquidity

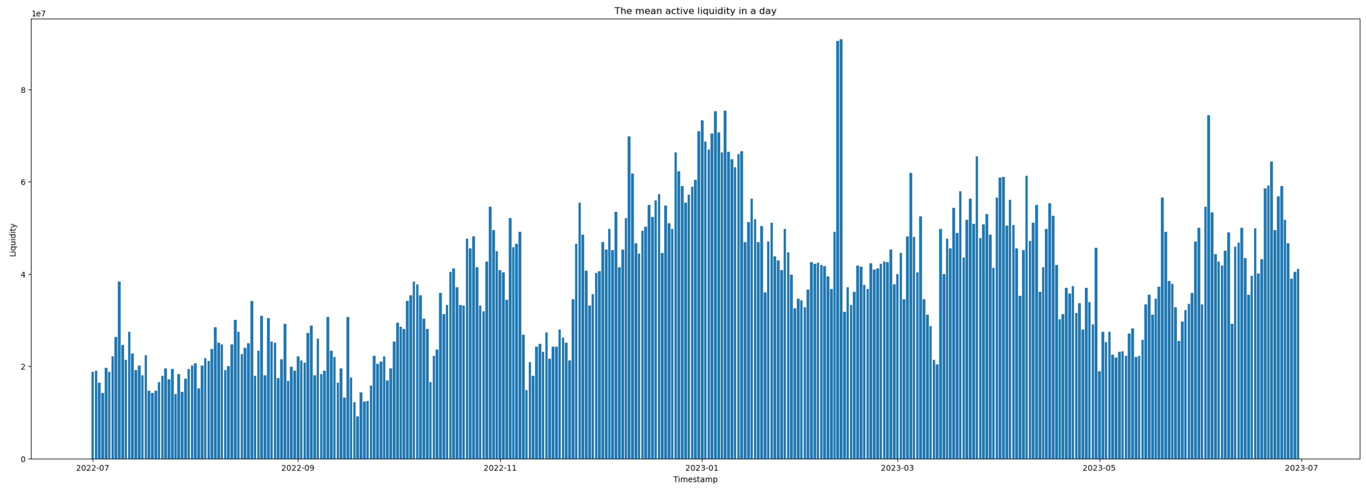

Figure 5: Evolution of Mean Active Liquidity

In Figure 13, we examine the mean active liquidity on Uniswap v3 over time. During periods of stress, liquidity depth becomes a critical factor in market stability. Our analysis reveals that during such periods, Uniswap experiences phases where liquidity depth is reduced, leading to a higher price impact for individual trades and contributing to increased market volatility. This volatility is exacerbated when active liquidity drops significantly, either due to liquidity providers (LPs) withdrawing capital (burning liquidity) or the market price moving to a range with low pre-existing liquidity.

The two possible causes for drops in active liquidity are as follows: (1) LPs exhibit a tendency to remove liquidity from the pool, resulting in a net reduction in available capital, or (2) the price transitions into a range where liquidity is scarce. Both scenarios are critical in assessing the resilience of the platform during stressed conditions.

3.3.3 Mint and Burn Activity

Figure 6 provides an overview of the net volume of mints and burns in the Uniswap v3 pool over time. Interestingly, contrary to the hypothesis that LPs might favor liquidity removal during stressed periods, the data indicate a relative equilibrium between the mint and burn activities. This balance suggests that there is no overwhelming tendency to withdraw liquidity, even during periods of heightened market stress. Instead, LPs maintain liquidity positions to capture transaction fees, a behavior which supports the operational resilience of Uniswap.

Nonetheless, localized imbalances in mint/burn activity can occur during extreme price fluctuations, leading to transient liquidity shortages in certain ranges.

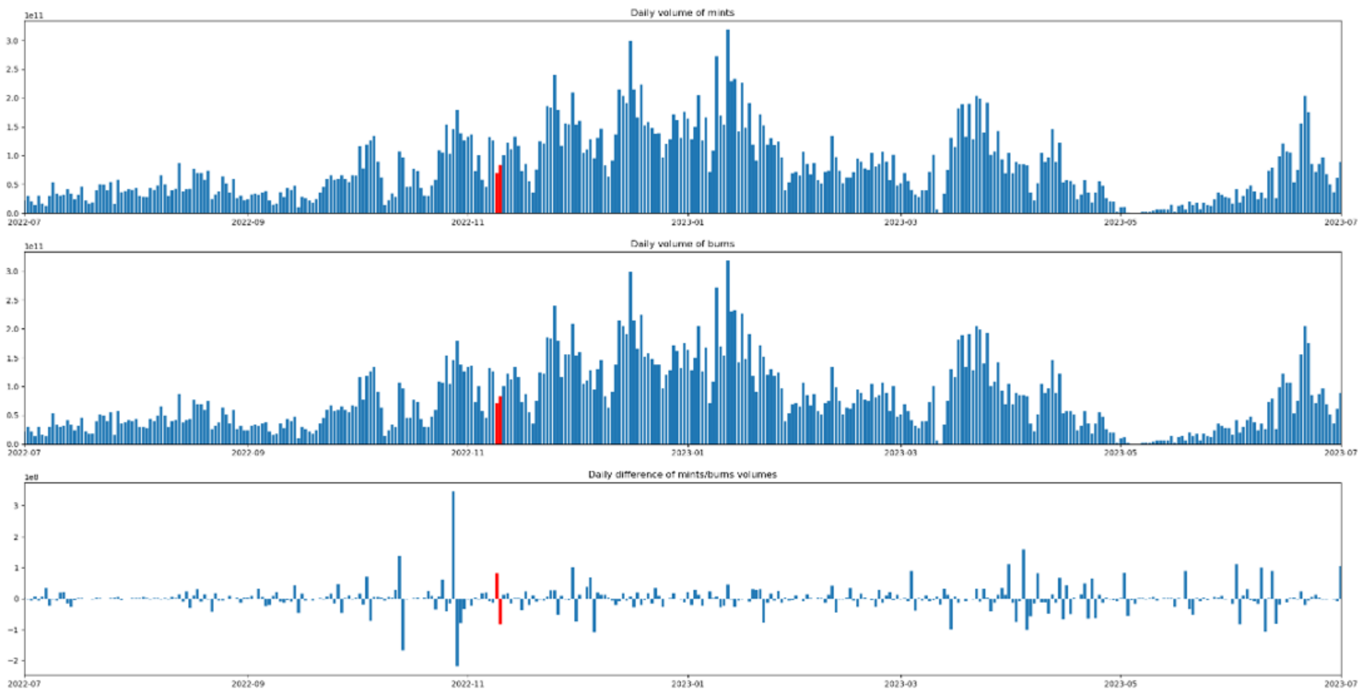

Figure 6: Evolution of Mints and Burns

3.3.4 Momentary Liquidity Concentration

A particularly interesting behavior observed during bear markets involves the momentary concentration of liquidity. Figure 7 shows the liquidity distribution over a two-hour window. In bear markets, LPs strategically mint highly concentrated liquidity just below the current price range, anticipating a further downward price movement. Once the price crosses their concentrated position, they quickly burn the liquidity, thereby minimizing exposure while maximizing transaction fees. This behavior demonstrates a pointwise operation that allows LPs to capture fees efficiently in low-liquidity environments, preserving the overall stability of the pool through the natural incentives of fee-driven liquidity provision.

Such behavior highlights the sophistication of LP strategies in bear markets and the importance of understanding liquidity distribution dynamics in volatile conditions.

3.3.5 Market Efficiency

In this section, we analyze the price dynamics and the efficiency of Uniswap v3 compared to Binance, focusing on the USDC/ETH pair. Using data collected over three distinct periods—a full history covering from the 1st of October 2022 to the 15th of November 2022, a crisis day on the 10th of November 2022 during the fall of FTX, and a typical day on the 6th of October 2022—we examine the spread between

Binance and Uniswap prices. This analysis is crucial for understanding Uniswap's market efficiency, particularly in times of market stress and under normal conditions.

Concept of Market Efficiency:
Market efficiency, as defined by Fama's Efficient Market Hypothesis (EMH), suggests that asset prices fully reflect all available information at any given time. Arbitrageurs play a key role in maintaining this efficiency, as they quickly exploit any price discrepancies across platforms, forcing prices back into alignment.

Since Binance is the largest crypto exchange in terms of both liquidity and trading volume, it is reasonable to assume that Binance prices reflect the true market value of assets, and thus, Binance can be considered an efficient market, particularly under normal conditions. Even though during periods of crisis even large markets like Binance may struggle to maintain perfect efficiency due to increased volatility, market actor panic, and sudden liquidity constraints, we will assume that any significant and sustained price discrepancy between Binance and Uniswap is a potential indicator of market inefficiency in Uniswap.

Price and Spread Analysis :
We plot the closing prices on Binance and Uniswap for the USDC/ETH pair across three different periods to observe the relationship between the two platforms and track the evolution of the absolute spread.

Full Period (1st October 2022 - 15th November 2022):
In the first figure, we present the prices on Binance and Uniswap over a period of one and a half months. The top panel shows the price trajectory, while the bottom panel shows the absolute spread between the two exchanges. Throughout this extended period, the spreads remain generally minimal, except during the crisis, when they become consistently pronounced. This suggests that, under normal conditions, prices on Uniswap and Binance tend to converge, with significant divergence during crisis events. However, several spikes in the spread are observed, likely due to aggressive adverse trades. These spikes are short-lived, and the market quickly returns to equilibrium, showing that the arbitrage mechanisms generally work efficiently to correct price discrepancies.

Zoom on the Crisis Period (10th November 2022):
The second figure shows the price dynamics during the 10th of November 2022, a day marked by the FTX collapse and extreme market turmoil. In the top panel, we observe that Binance and Uniswap prices begin to diverge more significantly compared to the full period. The bottom panel displays a dramatic increase in the absolute spread during this crisis, indicating a significant and sustained discrepancy between the two exchanges for about one hour. The sustained nature of the spread reflects how crises can disrupt price discovery on decentralized platforms like Uniswap, as

market participants panic, liquidity dries up, and arbitrage mechanisms may struggle to function effectively.

Zoom on a Normal Day (6th October 2022):
The third figure shows the price movements during a typical trading day, specifically chosen as a calm day with the highest observed spikes outside of the crisis period. On this day, the prices on Binance and Uniswap track closely together, with the spread remaining low and relatively stable throughout the day. This indicates that, under normal market conditions, Uniswap can maintain price efficiency, with any discrepancies between the two exchanges being negligible. However, occasional spikes in the spread are still observed, likely due to aggressive adverse trades. Despite these spikes, the market quickly corrects itself, demonstrating that the arbitrage mechanisms are effective under stable conditions.

Conclusion :
The spread analysis between Binance and Uniswap reveals important insights into Uniswap's market efficiency under different conditions:

- **Normal Conditions:** During stable periods, Uniswap demonstrates strong price efficiency, with prices closely aligned with Binance. Occasional spikes in the spread, likely due to temporary liquidity disruptions or aggressive trades, are swiftly corrected by arbitrage, maintaining overall market efficiency.
- **Crisis Conditions:** In contrast, during the crisis on 10th November 2022, the spread between the exchanges widens significantly and remains elevated for around an hour. This indicates that during market stress, such as the FTX collapse, Uniswap's efficiency declines as liquidity constraints, panic, and volatility challenge the effectiveness of arbitrage mechanisms.

In summary, Uniswap performs efficiently in calm markets but faces significant inefficiencies during crises, highlighting the importance of understanding these dynamics for liquidity providers and traders aiming to optimize strategies in decentralized markets.

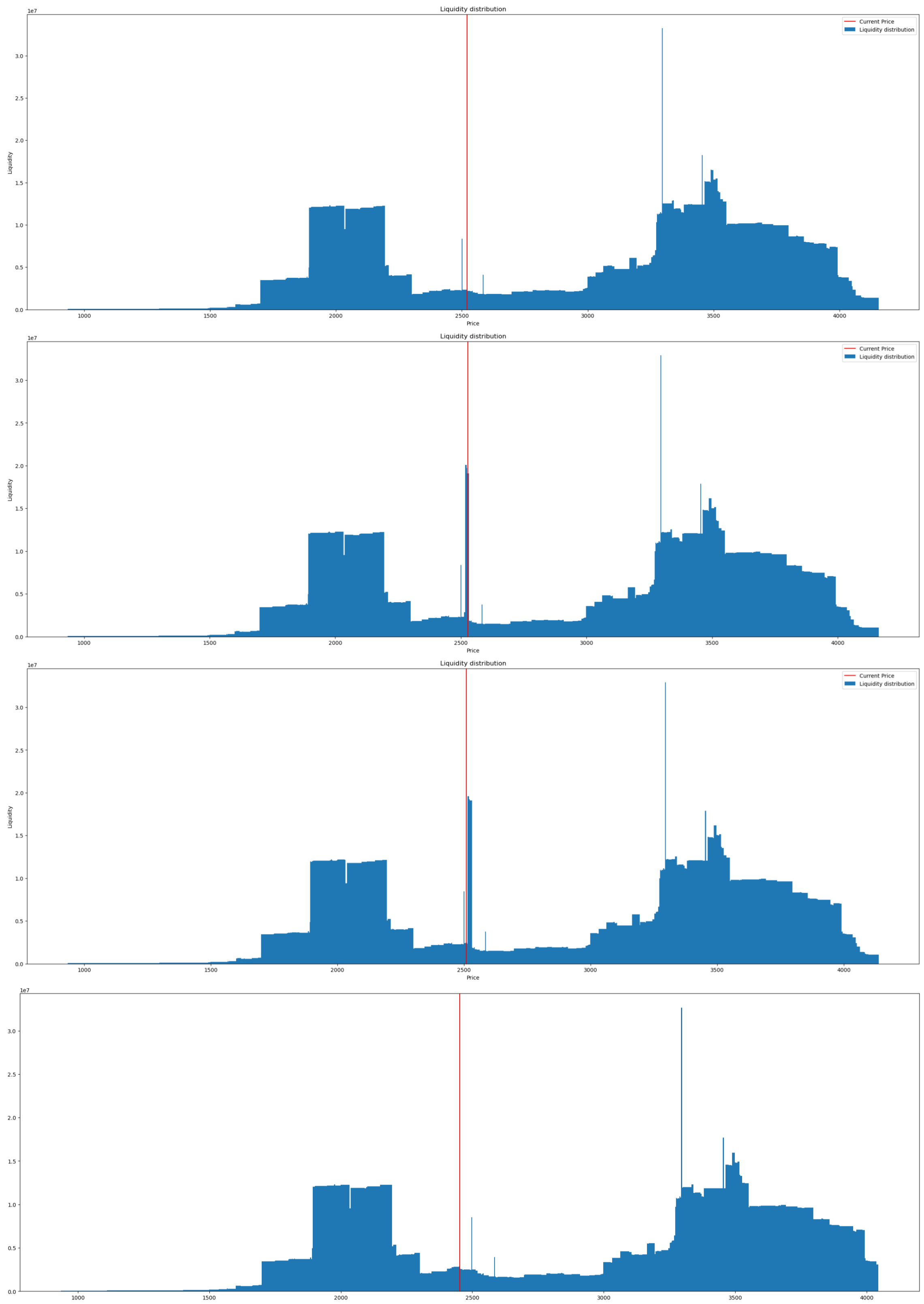

Figure 7: Momentary Liquidity Concentration in Bear Markets

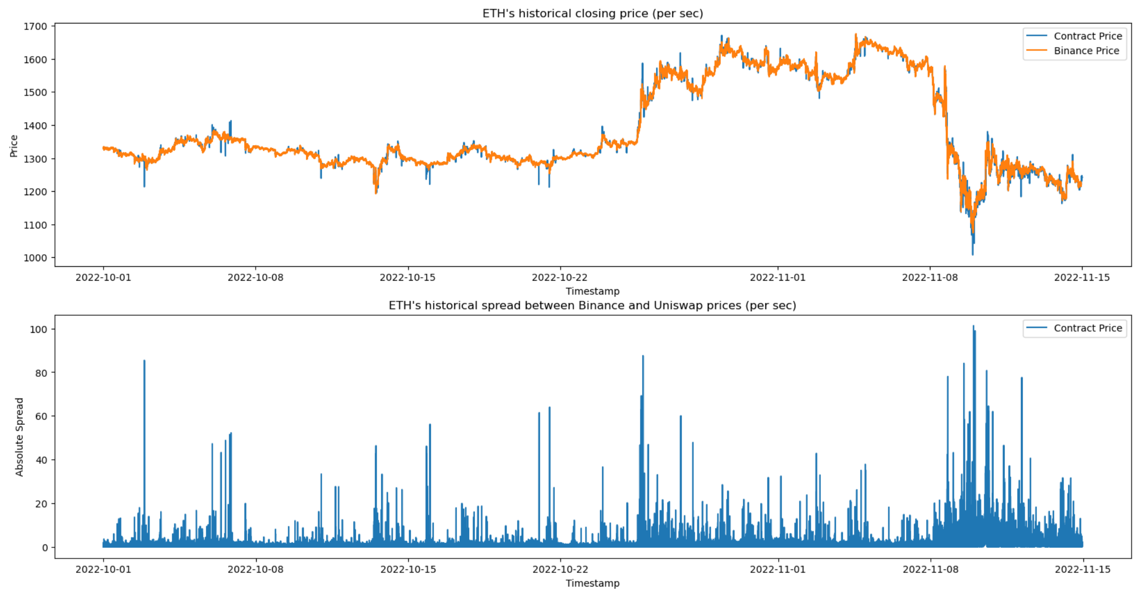

Figure 8: ETH's historical closing prices on Binance and Uniswap, and the absolute spread (per sec) from 1st October 2022 to 15th November 2022.

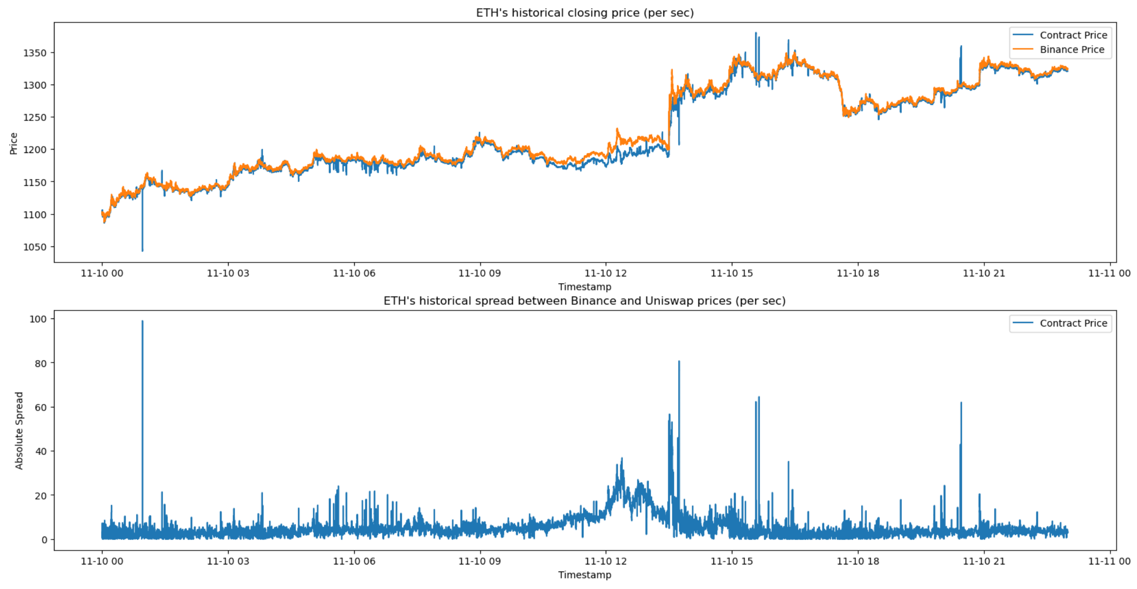

Figure 9: ETH's historical closing prices on Binance and Uniswap, and the absolute spread (per sec) during the FTX crisis on 10th November 2022.

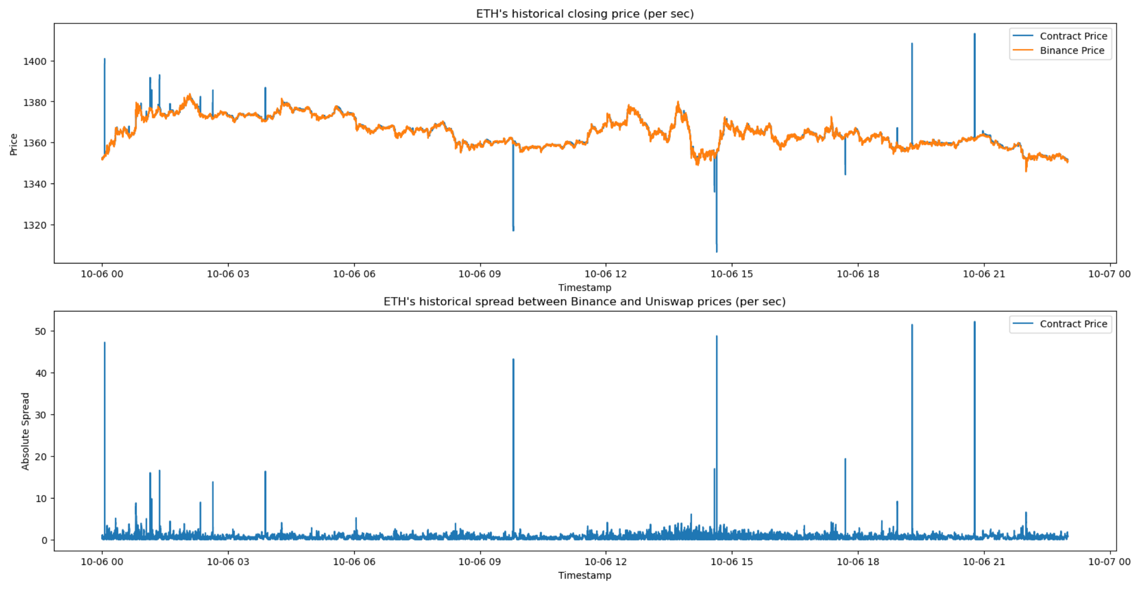

Figure 10: ETH's historical closing prices on Binance and Uniswap, and the absolute spread (per sec) on a normal day (6th October 2022).

4 Backtesting Environment for Uniswap v3

4.1 Introduction and Objectives

The development of a robust backtesting environment is essential for evaluating and optimizing liquidity provision strategies on Uniswap v3. The primary objectives of this environment are:

- **Accurate Simulation of Uniswap v3 Mechanics**: Replicate the core functionalities of Uniswap v3, including concentrated liquidity, fee calculations, and price movements, to provide a realistic testing ground for strategies.
- **Dynamic Strategy Evaluation**: Allow for the chronological monitoring of liquidity operations such as rebalancing, liquidity allocation, and fee accumulation over time.
- **Stress Testing Capabilities**: Enable the simulation of various market conditions, including high volatility and low liquidity scenarios, to assess the resilience of strategies.

By achieving these objectives, the backtesting environment serves as a crucial tool for liquidity providers to understand the potential risks and rewards associated with different strategies before deploying them in live markets.

4.2 Theoretical Foundations

In developing the backtesting environment for Uniswap v3, we focus on a major simplification for calculating the fees earned by liquidity providers (LPs). This simplification, inspired by the approach mentioned in [3], allows for efficient and accurate estimation of fees without the need of the general liquidity distribution of the pool. The genral liquidity distribution being extremely slow to construct and retrieve (About 1min for every snapshot), adding to that the need of continuously monitor it by incorporating the change that accur due to the actions of other liquidity providers, this simplification is humbly the only way of bringing this environment into life.

4.2.1 Simplified Fee Calculation

In Uniswap v3, the fees earned by an LP during a swap are defined by:

$$\text{Fees} = \left(\frac{l}{L}\right) \cdot f \cdot \Delta X \quad \text{or} \quad \left(\frac{l}{L}\right) \cdot f \cdot \Delta Y$$

where:

- l is the LP's active liquidity in the active price tick.

- L is the pool's total liquidity for the active price tick.
- f is the fee tier (e.g., 0.05%, 0.3%, 1%).
- ΔX and ΔY are the amounts of tokens X and Y paid by the trader during the swap, depending on the swap direction.

According to the Uniswap v3 whitepaper [1], the change in the pool's token amounts over a single swap is given by:

$$\Delta X' = L \cdot \Delta\left(\frac{1}{\sqrt{P}}\right)$$
$$\Delta Y' = L \cdot \Delta(\sqrt{P})$$

where:

- P is the price of token X in terms of token Y.
- $\Delta X'$ and $\Delta Y'$ represent the changes in the pool's token reserves excluding fees.

Over a single swap, the net change in the pool's tokens including fees is:

$$\Delta X' = \Delta X(1-f)$$
$$\Delta Y' = \Delta Y(1-f)$$

since the generated fees are not added back into the pool's liquidity but are instead accrued to the LPs.

By combining these equations, we can express the fees earned by an LP during a swap as:

$$\text{Fees} = l \cdot \frac{f}{1-f} \cdot \Delta\left(\frac{1}{\sqrt{P}}\right) \quad \text{or} \quad \text{Fees} = l \cdot \frac{f}{1-f} \cdot \Delta(\sqrt{P})$$

This simplification allows us to calculate the fees earned by an LP based solely on the changes in price and the LP's proportion of liquidity in the active tick, without needing of the general liquidity distribution at each moment. The total fees generated can then be obtained by summing over all swaps in the simulation period.

4.2.2 Features of the Backtesting Environment

The backtesting environment incorporates the following functionalities:

- **Adding Liquidity:** Allows users to add liquidity at a desired price range, effectively creating a new position.

- **Removing Liquidity:** Enables users to reduce or fully remove liquidity from a previously created position.
- **Simulating Price Changes:** Simulates the impact of price changes due to swaps, handling the attribution of corresponding fees generated, and managing active liquidity, including the critical crossing of initialized ticks.

By focusing on these key aspects, the environment efficiently handles the dynamics of liquidity provision and fee generation in Uniswap v3, providing a robust tool for evaluating and optimizing liquidity provision strategies.

4.3 Implementation Choices

The development of the backtesting environment for Uniswap v3 involved several critical implementation decisions aimed at accurately simulating liquidity provision strategies and assessing their performance from the perspective of an individual liquidity provider (LP). The primary goal was to create a tool that balances computational efficiency with a high degree of fidelity to the actual mechanics of Uniswap v3.

4.3.1 Object-Oriented Design

An object-oriented programming approach was adopted to model the complex interactions within a Uniswap v3 pool. The core component of the environment is the `LiquidityPool` class, which encapsulates the state and behavior of the pool, including liquidity positions, price movements, and fee accumulation. This design choice offers several advantages:

- **Modularity**: Each component of the pool (e.g., positions, ticks) is represented as an object or data structure, facilitating easier maintenance and potential extensions.
- **Reusability**: Functions for adding liquidity, simulating swaps, and calculating fees can be reused across different simulations and strategy tests.
- **Clarity**: The object-oriented structure enhances code readability and aligns closely with the conceptual model of liquidity pools and LP interactions.

4.3.2 State Tracking and Parameter Updates

As previously discussed in the section explaining the mechanism of Uniswap v3 and the simplifications applied, it is essential to update the key parameters of the pool after each event, such as a price change, addition, or removal of liquidity. Accurate state tracking ensures that the simulation reflects the true behavior of the pool and provides reliable results for strategy performance evaluation.

The environment continuously tracks the state of the pool through specific class attributes within the `LiquidityPool` class. These attributes are critical for managing the pool's dynamics and include:

- `positions`: A dictionary storing the LP's active liquidity positions, with each position identified by a tuple of its lower and upper price ticks $(\text{lower}, \text{upper})$ as the key. Each entry contains details such as the amount of liquidity, fee accruals, and position boundaries.
- `ticks`: A dictionary that records the initialized ticks in the pool, each associated with its corresponding change in liquidity ΔL. This structure allows the environment to efficiently update liquidity levels as the price crosses tick boundaries.
- `ordered_ticks`: A sorted list of all initialized tick values. Ordering the ticks facilitates rapid identification of the next tick to be crossed during price movements, optimizing the handling of active liquidity changes.
- `current_price`: A variable representing the current price S_t of the asset pair in the pool. This price is updated with each simulated market event and is crucial for determining the active liquidity and calculating fees.
- `l`: The LP's active liquidity at the current price. This parameter reflects the total amount of liquidity provided by the LP that is currently within the active price range and thus eligible to earn fees.
- `upper` and `lower`: Variables indicating the upper and lower tick boundaries that delimit the current active price range of the LP's positions. As the price moves, these boundaries are updated to reflect which positions are active.
- `fee_tier`: The fee tier f of the pool, representing the percentage fee charged on each swap (e.g., 0.05%, 0.3%, 1%).
- `fg_0` and `fg_1`: Gross accumulated fee in token0 and token1.
- `fo_0` and `fo_1`: Each tick has these two parameters, they are used to track fees generated outside of the tick, in token0 and token1. This parameter alone does not have a significance, but it is used along with `fg_0` and `fg_1` and the `current_price` to accurately and efficiently calculate the fees gained between two ticks. You can refer to Uniswap's whitepaper [1] for a more detailed explanation.

Updating Pool Parameters After each event, the environment updates these attributes to maintain an accurate representation of the pool's state:

1. **Price Change Events**:
   - The `current_price` is updated to the new price S_t.
   - The environment checks if the new price crosses any tick boundaries in `ordered_ticks`. If so, it adjusts `l`, `upper`, and `lower` accordingly by adding or subtracting the liquidity changes ΔL from `ticks`.
   - The cumulative fees `fg_0` and `fg_1` are updated based on the price movement and the LP's active liquidity l, by adding the new generated fees, using the fee calculation formulas:

$$\delta\text{fees}_{\text{token0}} = l \cdot \frac{f}{1-f} \cdot \Delta\left(\frac{1}{\sqrt{S_t}}\right), \quad \delta\text{fees}_{\text{token1}} = l \cdot \frac{f}{1-f} \cdot \Delta(\sqrt{S_t})$$

2. **Adding Liquidity**:
   - A new entry is added to `positions` with the specified lower and upper ticks, liquidity amount, and initial fee accruals.
   - The `ticks` dictionary is updated at the lower and upper ticks of the relevant position, to reflect the change in liquidity ΔL.
   - The `ordered_ticks` list is updated to include any new ticks, maintaining the sorted order.
   - If the current price is within the new position's range, `l`, `upper`, and `lower` are updated to include the new liquidity.

3. **Removing Liquidity**:
   - The specified position is removed from `positions`, if the relevant position doesn't have any liquidity left.
   - The `ticks` dictionary is updated to subtract the liquidity ΔL at the lower and upper ticks.
   - If the current price was within the removed position's range, `l`, `upper`, and `lower` are updated to reflect the change.

4.3.3 Contributions and Innovations

The backtesting environment introduces several innovations that enhance its utility:

- **Balance of Accuracy and Efficiency**: Implementation choices prioritize computational efficiency without significantly compromising the accuracy of the simulation, enabling extensive backtesting over long periods.

- **User-Friendly Interface**: The environment offers a clear and intuitive interface for defining strategies and interpreting results, making it accessible to both technical and non-technical users.

4.3.4 Limitations and Weaknesses

Despite its strengths, the environment has certain limitations:

- **Exclusion of External Costs**: While governance fees (fees collected by Uniswap's governance) are taken into account, Gas fees applied on adding and removing liquidity are not incorporated, mainly due to the difficulty of retrieving their historical data and tracking their evolution over time.
- **Streamlined Implementation**: The simulation adheres to the theoretical framework of Uniswap v3, but omits certain complexities present in the actual platform, which are necessary for efficiently managing operations at scale. For our purpose, we focus on a single liquidity provider (LP), justifying the exclusion of such complexities.
- **Core Feature Focus**: Our environment intentionally excludes non-essential features like fee discounts and other commercial mechanisms. The focus is on core functionalities, ensuring simplicity without undermining the analysis.

These limitations highlight areas for future enhancement to increase the realism and applicability of the simulation results.

4.4 Validation and Testing

To verify that our backtesting environment accurately simulates the dynamics of Uniswap v3's platform, we conducted tests using historical data to track the operations of a specific liquidity provider (LP). The goal of our testing was twofold:

- To compare the total value of fees gained by the LP through their operations with historical records.
- To assess the performance of each individual position opened by the LP and compare it with the actual performance observed in historical data.

4.4.1 How We Calculate the Return of a Position

We used Impermanent Loss (IL) as the primary metric to measure the performance of a position, incorporating accumulated fees into the formula:

$$IL = \frac{V_{pos} - V_{hold}}{V_{hold}}$$

where V_{pos} is the value of the position, including accumulated fees, and V_{hold} represents the value of simply holding the initial tokens outside the pool.

However, during this evaluation, we encountered two key challenges:

- **Historical Data Gaps:** Our historical data starts at a specific date, making it difficult to determine whether a position was opened prior to this date. This uncertainty can affect whether an event is truly a new mint of liquidity or merely a rebalancing of a pre-existing liquidity position. Similarly, large burns of liquidity might be misinterpreted as significant returns, while large mints not yet burned could result in unusually low returns.

- **Multiple Mints and Burns for a Single Position:** Even if we had precise information about the opening and closing of positions, calculating impermanent loss becomes complex for positions involving multiple mint and burn events. Since burn events do not always remove the exact liquidity that was initially minted, and mints may add liquidity to an existing position, accurately assessing performance is challenging.

To address these issues, we first ensured that the chosen LP's activity was fully available in our dataset by cross-referencing the activity history on Etherscan, an external data provider. We also validated that the LP had closed all positions that had been opened within our dataset. As shown in the figures below, the ranges opened by the LP align exactly with the ranges closed, providing an additional layer of confidence in the completeness of our data.

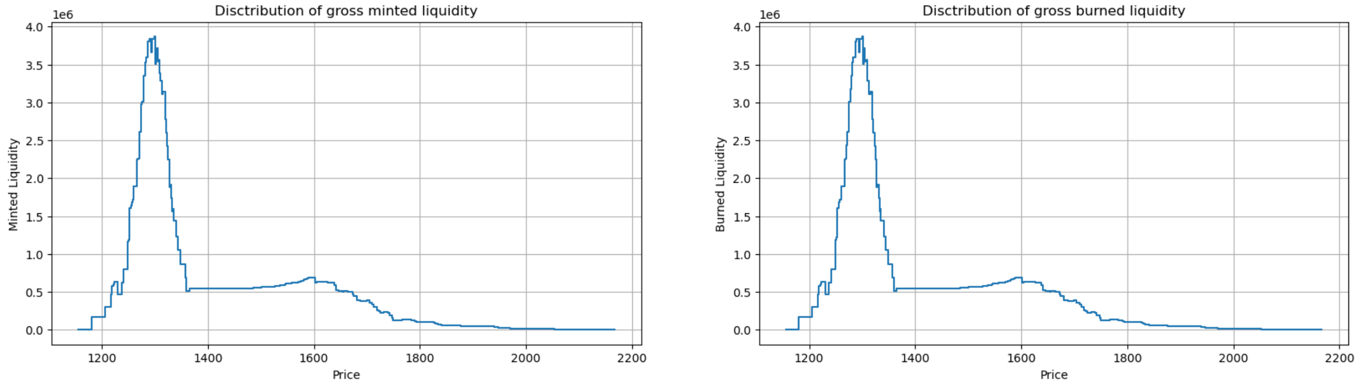

Figure 11: Ranges of liquidity opened and closed by the LP.

4.4.2 Refining the Performance Measure

To solve the second problem of handling multiple mint and burn events, we developed a refined measure inspired by the Impermanent Loss. This measure compares the value of the retrieved liquidity at the time of collection (including fees) with the value of the total tokens, evaluated at the last price at the moment of the final collect. This method amplifies the impact of impermanent loss on early mints by evaluating them at the last price in the time series, providing a more conservative estimate of returns.

Let $t = 0, 1, \dots, T$ represent the time index where $t = 0$ corresponds to the time when the position is first opened and $t = T$ represents the last time liquidity is collected (final burn event). We denote:

- $V_{\text{mint},t}$: The value of the liquidity minted at time t,
- $V_{\text{burn},t}$: The value of the liquidity burned at time t,
- F_t: Retrieved fees at time t,
- S_t: The price at time t,
- V_{hold}: The value of the tokens if they were held outside the pool throughout the duration of the strategy, evaluated at the final price S_T.

Thus:

- $V_{\text{mint},t} = Minted_t^0 \cdot S_T + Minted_t^1$, where $Minted_t^i$ is the amount minted of token i at time t.
- $V_{\text{burn},t} = Burned_t^0 \cdot S_t + Burned_t^1$, where $Burned_t^i$ is the amount burned of token i at time t.
- $F_t = fees_t^0 \cdot S_t + fees_t^1$, where $fees_t^i$ is the amount of fees collected in token i at time t.

The refined measure, IL_{refined}, is then calculated as:

$$IL_{\text{refined}} = \frac{V_{collect} - V_{hold}}{V_{hold}}$$

where $V_{collect}$ represents the sum of all liquidity retrieved, including fees, evaluated at the price at the time of retrieval:

$$V_{collect} = \sum_{t=0}^{T} (V_{\text{burn},t} + F_t)$$

Here, $V_{\text{burn},t} \cdot \frac{P_T}{P_t}$ adjusts the value of the burned liquidity from time t to its equivalent value at the final price P_T. The term F_t accounts for the total fees accrued up to time t.

On the other hand, V_{hold} is defined as the value of the total tokens that would have been held outside the liquidity pool, evaluated at the final price P_T:

$$V_{hold} = \sum_{t=0}^{T} V_{\text{mint},t} = (\sum_{t=0}^{T} Minted_t^0) \cdot S_T + (\sum_{t=0}^{T} Minted_t^1)$$

This method ensures that all events contributing to the position's performance (mints, burns, and fee collections) are taken into account, even when liquidity is added or partially burned multiple times during the strategy's execution.

4.4.3 Testing Results

Our simulation environment estimates a total of 6.80 ETH and 9451 USDC in accumulated fees, while the actual historical values are 6.61 ETH and 9185 USDC, respectively. This discrepancy can be attributed to several factors: our environment does not account for gas fees, which leads to an overestimation of the accumulated fees, and there may also be some inaccuracies in data handling. To reduce complexity, we have not implemented the most precise data types as used by Uniswap v3.

Despite these limitations, when plotting the returns of the LP's positions, we observe that the simulated results closely align with the historical performance. While our backtesting environment does not fully capture gas fees and is not infinitely precise, it successfully reflects the overall performance of each position. In particular, for positions with significant returns, the model effectively avoids inaccuracies that could arise from operational details.

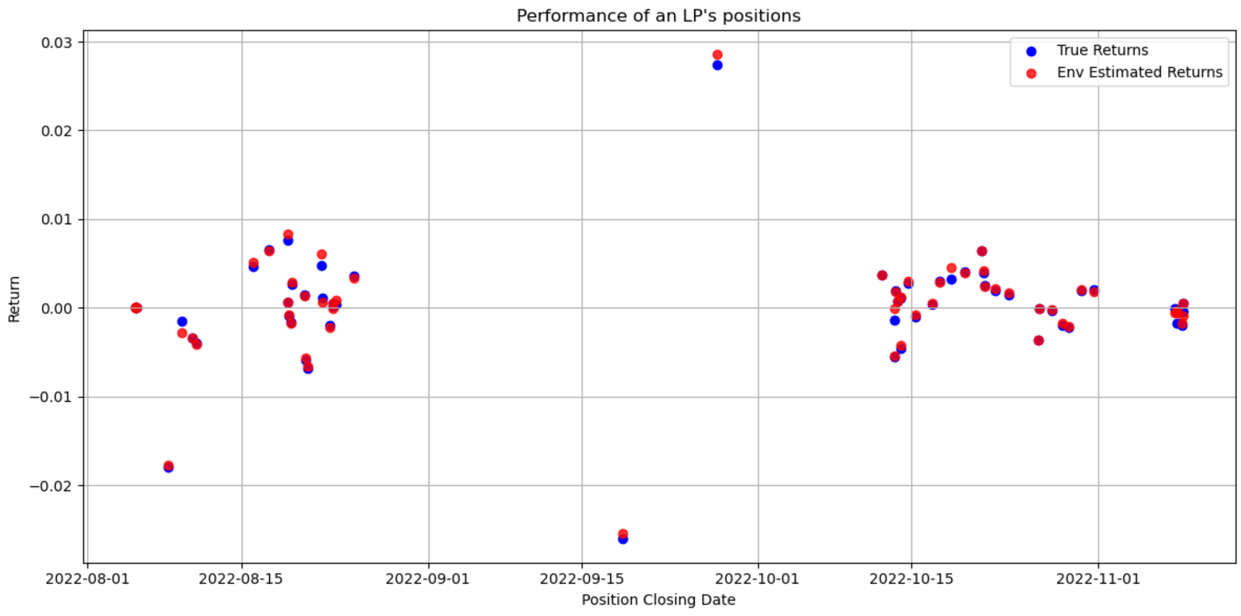

Figure 12: Returns of LP's positions: Simulated vs Historical.

5 Liquidity Provision Strategy Optimization

This section outlines an approach for optimizing a liquidity provision strategy using deep learning techniques. The primary objective is to maximize final wealth while minimizing risks related to price volatility and impermanent loss. Inspired by the work of [3], we employ a strategy that dynamically reallocates liquidity into predefined price ranges (buckets) with weighted allocations. This strategy is tested and refined within the backtesting environment described in earlier sections.

5.1 Problem Definition

The liquidity provider (LP) begins with an initial wealth W, which is distributed across several price buckets. The goal is to determine the optimal allocation of this wealth, both within the liquidity pool and outside of it, to balance risk and return. The strategy is defined as follows:

1. **Initial Wealth Allocation:** The LP initially holds a total wealth W, which is distributed between:
   - Liquidity to be provided within the pool, across different price buckets.
   - Wealth held outside the pool W_{hold}, offering flexibility and protection from impermanent loss.
2. **Weight Calculation:** We calculate the weight distribution w to determine how much liquidity should be minted in each price bucket. The objective is to

balance risk and return across $2\tau+1$ buckets, where τ represents the number of buckets on either side of the current price bucket B_0. The buckets are indexed by $i \in \{-\tau, \dots, -1, 0, 1, \dots, \tau\}$, each corresponding to a price range $[a_i, b_i]$, where $b_i = a_{i+1}$.

3. **Liquidity Minting:** Liquidity is minted in each bucket B_i by swapping the necessary amounts of token1 for token0 (See assumption in section 5.2.1), depending on the position of the bucket relative to the current price. The wealth allocated to each bucket is given by:

$$W_i = w_i W$$

where w_i is the weight assigned to the bucket, and W is the total wealth. The remaining wealth, W_{hold}, is kept outside the pool.

4. **Reallocation:** As the market price moves outside the current active range, the liquidity is withdrawn from all buckets, and the total wealth (including fees) is recalculated. The capital is then reallocated into new buckets centered around the updated market price. During reallocation, the liquidity provider must mint new positions by swapping between token0 and token1, ensuring that the new allocations match the current market price distribution.

5. **Final Wealth Retrieval:** At the end of the strategy, the LP retrieves the final wealth, including liquidity and accumulated fees. This final wealth is then compared against the wealth that would have been achieved by simply holding the initial tokens outside the pool.

5.1.1 Mathematical Formulation of Wealth Distribution

The strategy distributes a fixed amount of token1 liquidity across $2\tau + 1$ price buckets, indexed from $-\tau$ to τ. The wealth allocation for each bucket $W_i = w_i W$ is calculated based on the weight w_i, subject to the constraint that:

$$W = W_{hold} + \sum_{i=-\tau}^{\tau} W_i$$

where W_{hold} represents the wealth held outside the pool. This allocation structure provides flexibility to manage risk by holding part of the wealth outside the market exposure. We note $w^n = (w^n_{-\tau}, \dots, w^n_{\tau}, w^n_{hold})$ the vector of weights, generated for the reallocation number n, $n \in [\![0, N]\!]$ where N is the total number of reallocations.
.

5.1.2 Reallocation and Wealth Calculation

Let us first introduce some usefull notations:

- W^n is the wealth at reallocation n, after retrieving liquidity and just before minting the new one.
- W_b^n is the value of liquidity at reallocation n just before retrieving it.
- W_m^n is the value of the liquidity minted into the pool in the reallocation process.
- γ is the fee applied to burning or minting liquidity (e.g., gas fees). For simplicity, we model $\gamma \in [\![0, 1]\!]$, as a constant parameter, which can be varied to show its impact on wealth.

When the market price moves outside the active price range, a reallocation process is triggered. During this process:

- Available liquidity and accumulated fees, of total value W_b^n, is withdrawn from the pool.
- We have that $W^n = \gamma W_b^n + w_{hold}^{n-1} W^{n-1}$
- A new liquidity, valued in token1 at $W_m^n = \gamma W^n (1 - w_{hold}^n)$, is then reallocated into new price buckets centered around the current market price, following the distribution given by w^n.

Assumption: To effectively reallocate wealth, it is assumed that the LP can perform swaps between token0 and token1 with negligible market impact. This assumption is crucial because the retrieved liquidity may consist of uneven proportions of token0 and token1 (e.g., fully token0 or token1 if the price moves outside the liquidity range), which would need to be rebalanced to mint a new position centered around the current price. For the sake of this model, we assume the LP is small enough to have a minimal impact on the market during these swaps.

5.2 Loss Function and Utility Maximization

The loss function $\mathcal{L}(w_0, \dots, w_N)$, is constructed to measure the performance of the liquidity provision strategy relative to simply holding the initial assets, adjusted for price volatility and risk preferences. The loss function is defined as:

$$\mathcal{L}(w_0, \dots, w_N) = -u_a \left(\frac{V_{final} - V_{\text{hold}}}{V_{\text{hold}}} \cdot \frac{1}{\sigma_p} \right)$$

where:

- $V_{final} = W^{N+1}$ represents the total final wealth at the end of the strategy. Here, W^{N+1} is computed as if a final $(N+1)^{\text{th}}$ reallocation had taken place, where:
$$W^{N+1} = \gamma W_b^{N+1} + w_{hold}^N W^N$$

- V_{hold} represents the wealth the liquidity provider (LP) would have if they had simply held the initial tokens without providing liquidity.
- σ_p is the volatility of the price path over the duration of the strategy.
- The utility function $u_a(x)$ is defined as:

$$u_a(x) = \begin{cases} \frac{1-e^{-ax}}{a} & \text{if } a \neq 0, \\ x & \text{otherwise.} \end{cases}$$

  where a represents the risk aversion coefficient. A positive a indicates risk-averse behavior, while $a < 0$ corresponds to risk-seeking preferences. When $a = 0$, the LP is risk-neutral.

The choice of this loss function reflects the goal of incorporating impermanent loss, allowing us to assess the benefits of the strategy compared to simply holding the initial tokens. The inclusion of volatility ensures a fair evaluation across different price path scenarios, while the utility function introduces flexibility in controlling the level of risk aversion.

5.3 Price Process Modeling for Data Generation

To model Uniswap v3's price process, we adopted the price modeling framework proposed by [2], which was later utilized by [3] in their strategic liquidity provision framework. This model was chosen for its tunable nature and high flexibility, enabling us to simulate a wide range of price behaviors beyond those observed in historical data. It captures the interactions between Uniswap and centralized exchanges like Binance, while incorporating arbitrage activities that drive price corrections. Although this approach offers significant advantages, it has its limitations and cannot fully account for the inherent unpredictability of Uniswap v3's price movements. Therefore, it is crucial to evaluate the performance of our models by training and testing them on both real price paths and simulated price paths to ensure robustness and accuracy.

5.3.1 Notations and Definitions

The price process is defined as follows:

- $P = (P_0, \ldots, P_T)$, where each price at time t, P_t, consists of:
  - $P_{c,t}$: the price on Uniswap at time t,
  - $P_{m,t}$: the price on a centralized exchange (e.g., Binance) at time t.
- $T = \sum_{r=1}^{R}(k_r + Arb_r)$, where:

  - Arb_r is the number of arbitrage transactions in round r,
  - k_r is the number of non-arbitrage transactions in round r.

- λ_r is a measure of the impact of non-arbitrage transactions in round r, modeling slippage due to liquidity trades.
- $\gamma \in (0, 1)$ is the fee tier on Uniswap, which represents the cost of arbitrage transactions.

5.3.2 Model Dynamics

The price process modeling consists of R rounds, where each round simulates interactions between the centralized market price and the Uniswap price. The key steps are as follows:

1. **Centralized Exchange Price Evolution**: At the beginning of each round r, the centralized market price $P_{m,t}$ evolves according to a calibrated Geometric Brownian Motion..
2. **Uniswap Price Changes**: During each round, the Uniswap price $P_{c,t}$ is updated through k_r non-arbitrage transactions. Each transaction shifts the price based on its impact λ_r, resulting in the new price:
$$P_{c,t+1} = (1 + \lambda_r)P_{c,t}, \quad \text{or} \quad P_{c,t+1} = (1 + \lambda_r)^{-1}P_{c,t}$$
depending on the direction of the transaction, with an equal 50% probability for each direction.
3. **Arbitrage and Price Correction**: After each price change in Uniswap, arbitrage opportunities may arise due to differences between $P_{c,t}$ and $P_{m,t}$. Arbitrageurs will act to exploit these price discrepancies, adjusting $P_{c,t}$ toward $P_{m,t}$. Arbitrage is triggered if $P_{c,t}$ falls outside the arbitrage bounds $I_\gamma(P_{m,t})$, defined as:
$$I_\gamma(P_{m,t}) = \left[(1 - \gamma)P_{m,t}, (1 - \gamma)^{-1}P_{m,t}\right]$$
When $P_{c,t}$ exits this range, arbitrage transactions move the price back toward the nearest boundary within the interval, ensuring that Uniswap prices stay in sync with centralized exchanges.

5.3.3 Flexibility of the Price Model

This model offers high flexibility, as parameters like λ_r, γ, and k_r can be tuned to simulate various market conditions. For example, increasing λ_r can simulate low-liquidity environments where trades have a larger impact on prices, while adjusting γ allows us to model different Uniswap fee structures. These features make the model highly adaptable for studying price dynamics and liquidity provision strategies in different market scenarios.

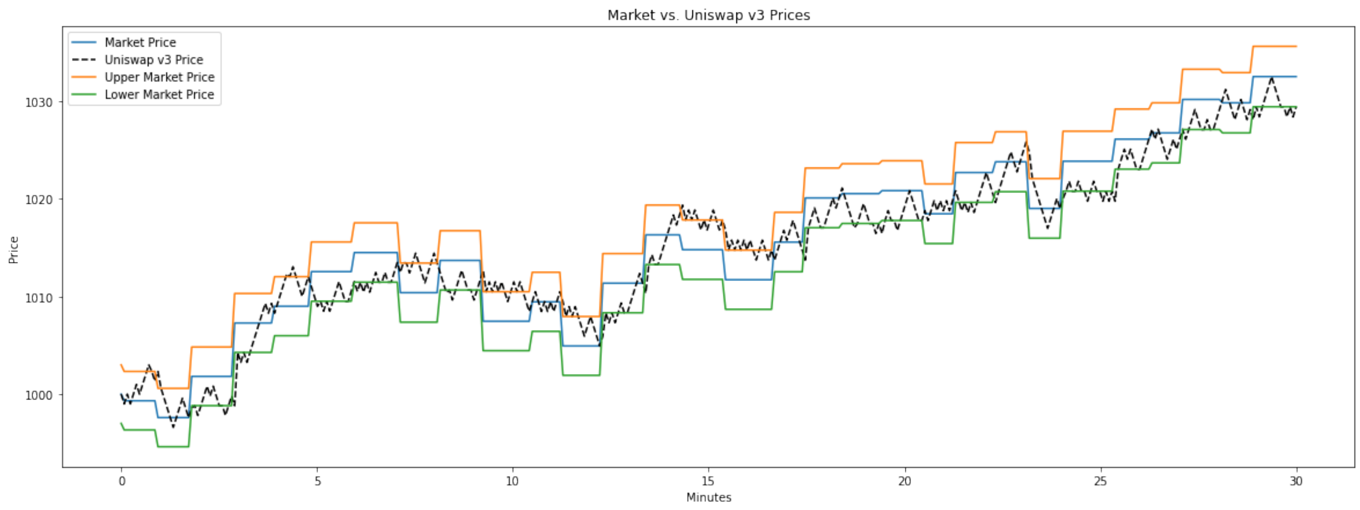

Figure 13: Price evolution example, with $\mu = 0.05$ and $\sigma = 0.02$ over 30 minutes. Upper and Lower market prices represent the upper and lower arbitrage bounds according to our price modeling.

5.3.4 Price Model Tuning

For our purposes, we calibrated relevant parameter values that reflect both Binance price history and Uniswap v3's USDC/ETH pool dynamics.

Binance Price:
Using minute-level price data for the USDC/ETH pair on Binance, we derived the drift (μ) and volatility (σ) of the price process. Additionally, we examined the log-normality assumption of our geometric Brownian motion (GBM) model, which is often not fully validated in financial time series.

As observed in the Q-Q plot, deviations in the tails indicate that the log-normality assumption is not entirely accurate, particularly for extreme values. Despite this, we proceed with a GBM model for simplicity. The empirically calibrated parameters are:

$$\mu = 6.02 \times 10^{-7}, \quad \sigma = 0.11\%$$

Parameters for Simulating Uniswap v3's Price Process:

For the simulation of Uniswap v3's price process, we centered the choice of model parameters on the empirical mean with a uniform variability of $\pm$ standard deviation (std). This approach allows us to model typical market behavior while incorporating a realistic range of fluctuations observed in the data.

Trades per Minute (k_r):
The parameter k_r, representing the number of trades per minute, was estimated based on the empirical mean of trades per minute observed in the USDC/ETH pool. To account for natural variability in trade frequency, we model this parameter

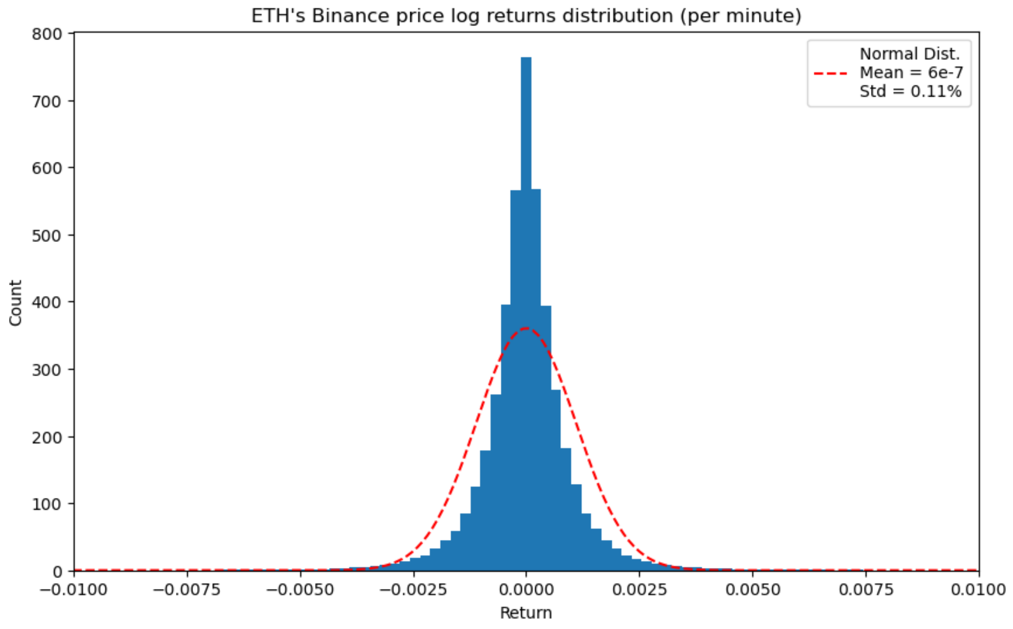

Figure 14: Log returns distribution (return per minute)

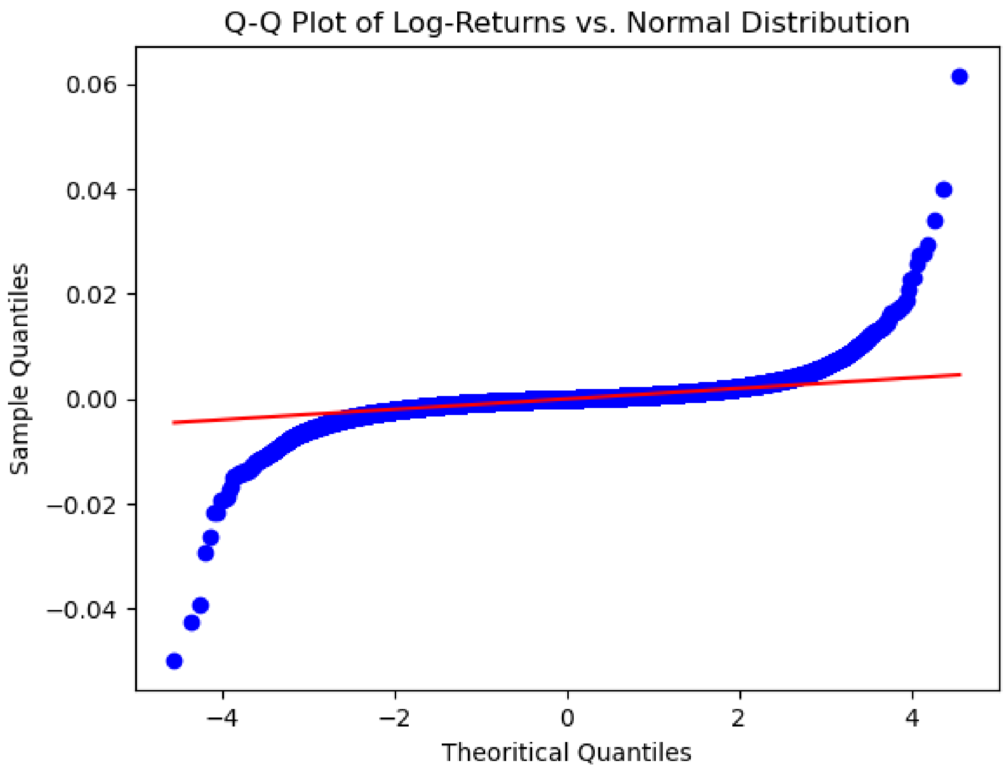

Figure 15: Q-Q plot of the log returns

by drawing random values from a Gaussian distribution with the same empirical mean and standard deviation σ_{k_r} as the observed data. This approach captures the stochastic nature of trade frequency.

Price Impact (λ_r):
The price impact parameter λ_r, which represents the average price change per trade, was similarly modeled by drawing values from a Gaussian distribution. This distribution is centered around the empirical mean price impact with a standard deviation of σ_{λ_r}, reflecting the observed variability in trade sizes and liquidity conditions. This method better captures the randomness and variability in price changes within the pool.

Fee Tier (γ):
The fee tier γ, which influences arbitrage behavior, was set to 0.05%, consistent with the fee tier of the Uniswap v3 USDC/ETH pool. This parameter directly affects the price bounds within which arbitrage opportunities are triggered, and no additional variability was introduced for this parameter as it is a fixed value set by the protocol.

This parameter selection methodology ensures that our simulations are centered around realistic market conditions while allowing for sufficient randomness to capture the dynamic nature of Uniswap v3's price process.

5.3.5 Data Generation

To generate sufficient training data for our deep learning model, we simulated 1000 independent Uniswap v3 price paths based on the price model discussed in previous sections. Each price path represents a 24-hour period, corresponding to $T = 1440$ minutes.

Storage of Generated Data:

The price paths are stored in a CSV file, where each row corresponds to a single 24-hour time series. The format allows for efficient storage and retrieval of time series data, facilitating quick access for training the deep learning models.

The choice of 1000 simulated price paths ensures a diverse and robust dataset, capable of capturing various market conditions, such as periods of low liquidity, high volatility, and frequent arbitrage. By simulating different scenarios, the training data provides a rich basis for the model to learn and generalize to unseen market environments.

5.4 Selected Deep Learning Models

The deep learning models used in our optimization process are Recurrent Neural Networks (RNNs), first introduced by [4], which are particularly effective for learning from sequential data such as time series. RNNs are capable of handling inputs of varying lengths, making them suitable for modeling dynamic price paths and liquidity reallocations over time. This flexibility is crucial given the variability in market conditions and the non-uniform nature of price data in our simulations.

5.4.1 Dynamic Optimization with Reallocations

In our approach, we incorporate liquidity reallocations as part of the optimization process, offering a more comprehensive evaluation over the entire price path. At each triggered reallocation, the RNN generates a weight vector that determines how the liquidity should be redistributed across different price buckets. When reallocation is necessary, the model simulates burning the current liquidity positions and minting new ones based on the predicted weights.

This dynamic approach allows the model to adapt liquidity positions in response to real-time changes in market conditions. The model leverages historical information from previous reallocations to make more informed decisions, optimizing the strategy over time and potentially leading to higher returns. This long-sighted strategy is designed to maximize performance by actively adjusting liquidity in volatile markets, thus minimizing risks such as impermanent loss.

5.4.2 Simple RNN

A Simple Recurrent Neural Network (Simple RNN) is one of the most fundamental types of RNNs, where the output from the previous time step is fed back into the network, allowing the model to retain information from previous inputs. The architecture consists of an input layer, a recurrent hidden layer, and an output layer. Each unit in the hidden layer is connected to the previous hidden state, enabling the model to learn from sequential data by maintaining a 'memory' of the past.

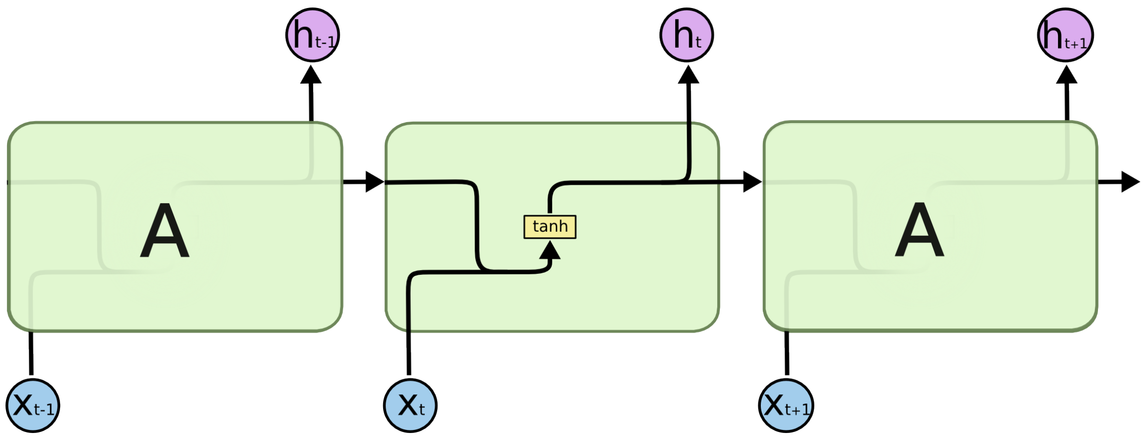

Figure 16: Simple RNN Architecture [5].

The Simple RNN in Figure 16 was used as one of the initial models to test how well the network could predict the optimal liquidity allocation weights based on past price paths and reallocation events, by applying a softmax activation function on the output to have an output vector that sums up to 1. While Simple RNNs can effectively model sequential data, they struggle with long-term dependencies due to the vanishing gradient problem, which can limit their performance in capturing more complex price dynamics.

5.4.3 Long Short-Term Memory (LSTM)

To address the limitations of Simple RNNs, we also employed Long Short-Term Memory (LSTM) networks, first introduced by [6], a more advanced type of RNN designed to overcome the vanishing gradient problem. LSTMs introduce a more complex architecture with three gates: the input gate, forget gate, and output gate. These gates allow the model to selectively retain or discard information, making LSTMs particularly well-suited for learning long-term dependencies in sequential data.

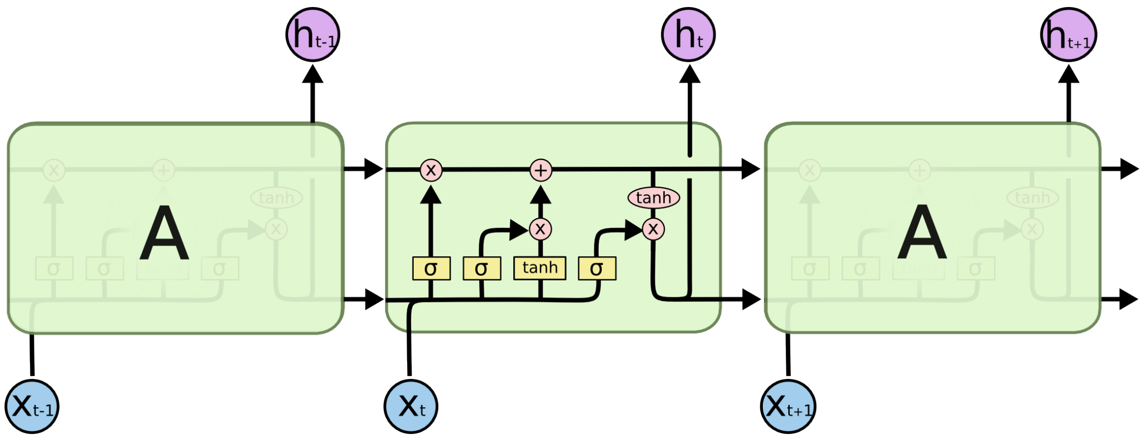

Figure 17: LSTM Architecture[5].

LSTMs (detailed in Figure 17 and [5]) were applied to optimize liquidity provision strategies by leveraging their ability to better capture the relationships between price movements and liquidity operations over extended time periods, by applying again a softmax activation function to make the output vector sum up to 1. This allows the model to make more accurate predictions regarding when and how to reallocate liquidity, improving overall strategy performance.

By using LSTM networks, we aim to capture both short-term and long-term dependencies in price behavior, leading to more robust and adaptive liquidity management strategies. The flexibility and memory retention of LSTMs make them a powerful tool for optimizing liquidity allocation over dynamic price paths.

5.5 Analysis of Liquidity Allocation Strategies

We evaluate three models:

- **Constant Model**: Serves as a naïve benchmark with uniform liquidity allocation.

- **Recurrent Neural Network (RNN) Model**: Trained over 20 epochs on 200 price paths, each representing a day's price movement, with a learning rate of 0.001. The architecture consists of 2 hidden layers with a hidden size of 16 and a dropout rate of 0.2. Tested with varying τ and a values.
- **Long Short-Term Memory (LSTM) Model**: Similar to the RNN model, trained over 20 epochs on 200 price paths, with a learning rate of 0.001. It also consists of 2 hidden layers with a hidden size of 16 and a dropout rate of 0.2. Tested with varying τ and a values.

The performance metrics are derived from 200 real price paths. This approach ensures not only an unbiased evaluation of the models' ability to generalize to new, unseen market conditions, but also a test of the robustness of our price modeling used in the creation of training data. The metrics assessed include Mean Return, Value at Risk (VaR) at the 95% confidence level, Conditional Value at Risk (CVaR) at the 95% confidence level, quartiles, as well as maximum and minimum returns. It is important to emphasize that the returns reflect the performance of one trading day, which is crucial for properly interpreting the magnitude of these values.

5.5.1 Results

The summary of the results can be found in Table 1 and Figure 18 & 19.

Model	**Mean Return (%)**	**VaR (95%) (%)**	**CVaR (95%) (%)**	**Q1 (%)**	**Median (%)**	**Q3(%)**	**Max Return (%)**	**Min Return (%)**
Constant, $\tau = 20$	-0.6678	-3.7632	-6.5962	-0.6940	-0.2541	0.1364	0.7901	-16.4441
Constant, $\tau = 50$	-0.2026	-1.2667	-2.4647	-0.1902	0.0508	0.1211	0.5447	-4.1874
Constant, $\tau = 100$	-0.0489	-0.5835	-1.1461	-0.0348	0.0286	0.0694	0.7170	-2.5577
RNN, $\tau = 20, a = 0$	-0.0038	-0.0207	-0.0414	-0.0042	-0.0011	0.0009	0.0113	-0.1064
RNN, $\tau = 50, a = 0$	-0.0174	-0.1759	-0.3130	-0.0296	0.0017	0.0198	0.3140	-0.5378
RNN, $\tau = 100, a = 0$	-0.0737	-2.0511	-3.1153	-0.0275	0.0692	0.2208	3.5249	-5.1799
LSTM, $\tau = 20, a = 0$	-0.0002	-0.0012	-0.0024	-0.0003	-0.0001	0.0000	0.0005	-0.0062
LSTM, $\tau = 20, a = 10$	-0.0010	-0.0053	-0.0106	-0.0011	-0.0003	0.0002	0.0027	-0.0271
LSTM, $\tau = 20, a = 100$	-0.0013	-0.0073	-0.0145	-0.0015	-0.0003	0.0003	0.0038	-0.0362
LSTM, $\tau = 50, a = 0$	-0.0025	-0.0321	-0.0520	-0.0058	0.0004	0.0042	0.0738	-0.0842
LSTM, $\tau = 50, a = 10$	-0.0014	-0.0155	-0.0253	-0.0034	0.0000	0.0017	0.0316	-0.0441
LSTM, $\tau = 50, a = 100$	-0.0019	-0.0262	-0.0475	-0.0071	-0.0001	0.0044	0.0828	-0.0766
LSTM, $\tau = 100, a = 0$	-0.0576	-2.6810	-3.6639	-0.0013	0.0046	0.1968	4.8973	-6.1056
LSTM, $\tau = 100, a = 10$	-0.0589	-2.6573	-3.7641	-0.0004	0.0018	0.2372	4.6097	-6.1268
LSTM, $\tau = 100, a = 100$	-0.0575	-2.6677	-3.6478	-0.0028	0.0056	0.1955	4.8588	-6.0846

Table 1: Performance metrics for different models, with returns expressed as percentages or in scientific notation when the values are too small.

Constant Models:

The Constant models exhibit a trend where increasing τ from 20 to 100 results in a less negative Mean Return, indicating improved performance. Specifically, the Mean Return increases from -0.6678% to -0.0489%. Similarly, both VaR and CVaR become less negative with increasing τ, suggesting reduced downside risk. The quartiles shift towards zero or positive values, and the minimum returns become less negative, indicating a decrease in extreme negative outcomes.

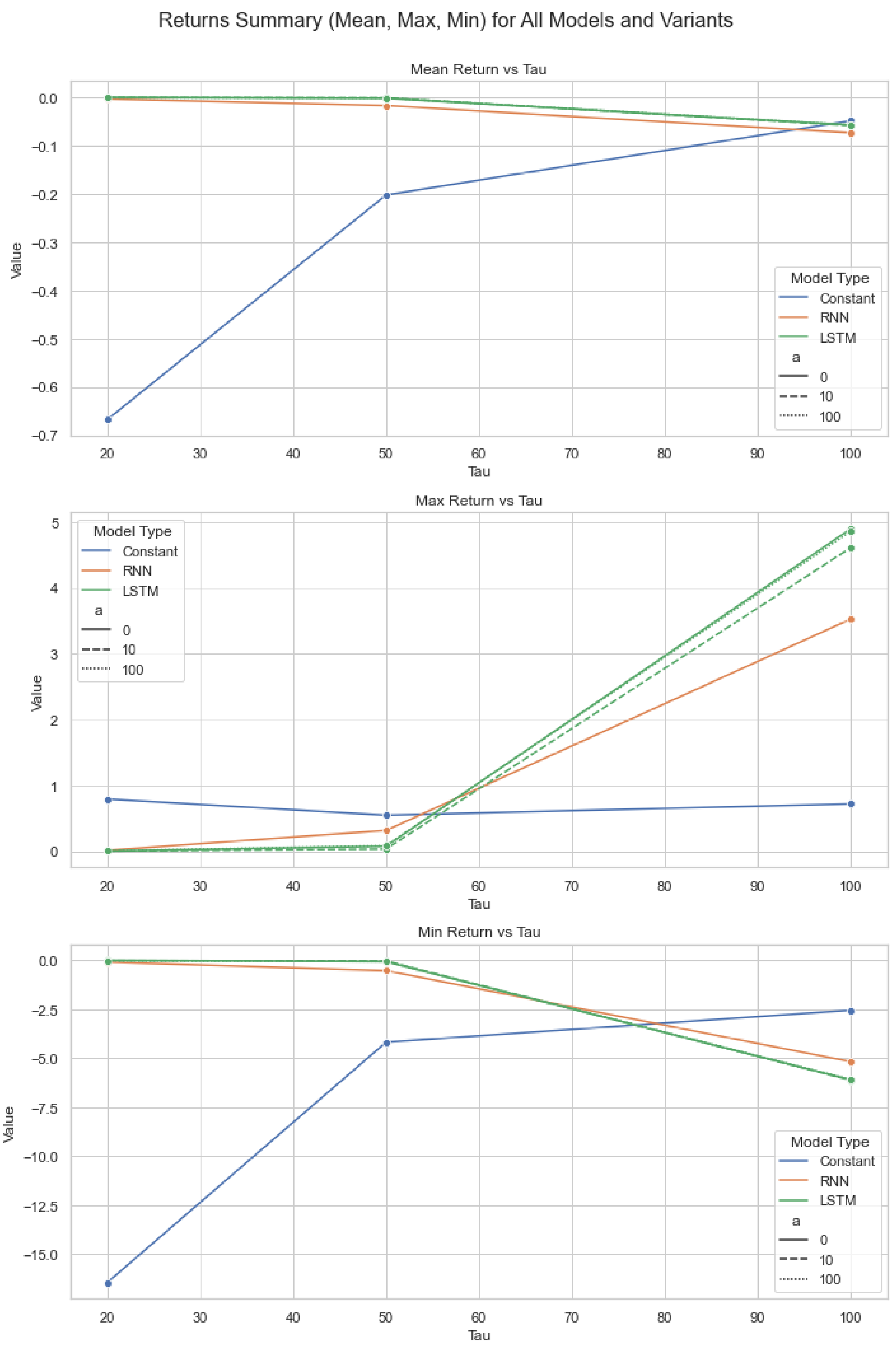

Figure 18: Returns Summary (Mean, Max, Min) for All Models and Variants.

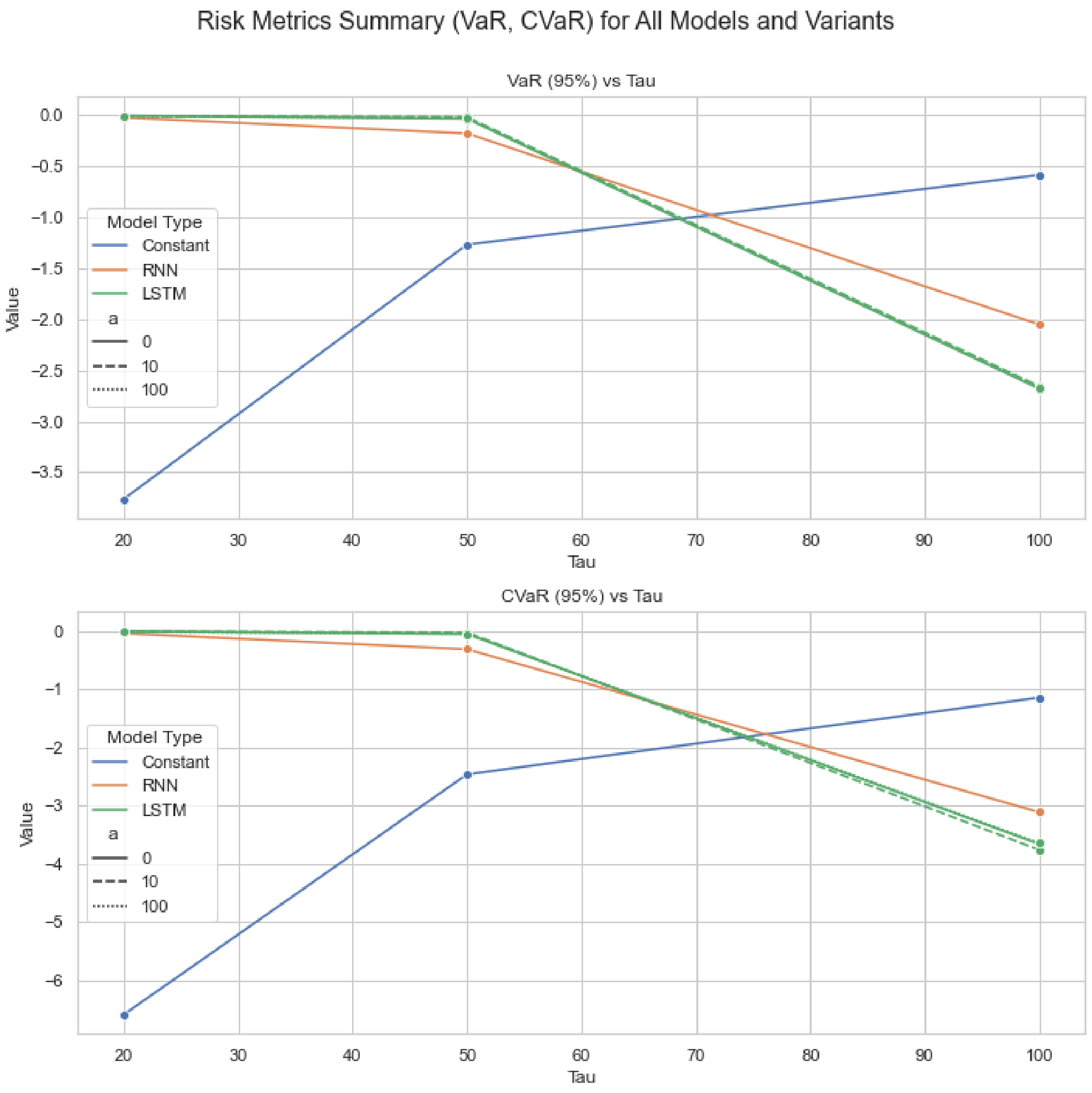

Figure 19: Risk Metrics Summary (VaR, CVaR) for All Models and Variants.

RNN Models:

For the RNN models with $a = 0$, decreasing τ from 100 to 20 leads to a Mean Return that approaches zero, improving from -0.0737% to -0.0038%. VaR and CVaR also become less negative with decreasing τ, indicating lower risk. However, the maximum returns decrease with smaller τ, suggesting a trade-off between potential gains and risk mitigation.

LSTM Models:

The LSTM models present a more nuanced picture. At $\tau = 20$, increasing the risk aversion parameter a from 0 to 100 results in a slightly more negative Mean Return, from -0.0002% to $-0.0013\% \times$. VaR and CVaR also become more negative with higher a, indicating increased downside risk. Quartiles shift negatively, reflecting a more conservative strategy that potentially limits gains.

At $\tau = 50$, the effect of varying a is less pronounced. The Mean Return fluctuates slightly, and no clear pattern emerges. For $\tau = 100$, changing a has minimal impact on the Mean Return and risk measures, suggesting that at higher τ, the influence of risk aversion diminishes.

Impact of the Number of Price Buckets (τ):

Across all models, τ significantly affects performance:

- **Constant Models**: Higher τ leads to better performance, with less negative Mean Returns and reduced risk, as evidenced by less negative VaR and CVaR values.
- **RNN Models**: Increasing τ from 20 to 100 results in more negative Mean Returns and increased risk, contrary to the trend observed in Constant models.
- **LSTM Models**: Similar to the RNN models, higher τ correlates with more negative Mean Returns and higher risk measures.

This divergence suggests that while a uniform allocation benefits from increased granularity (higher τ), the neural network models might overfit or fail to generalize effectively with more price buckets.

Impact of Risk Aversion Parameter (a):

The risk aversion parameter a influences the models differently:

- **At $\tau = 20$**: Increasing a in LSTM models leads to more negative Mean Returns and higher risk, indicating that higher risk aversion may limit the model's ability to capitalize on profitable opportunities.
- **At $\tau = 50$ and $\tau = 100$**: The effect of a is less significant, suggesting that at higher τ, the models' performance is less sensitive to changes in risk aversion.

Comparative Analysis:

- **Constant vs. Neural Network Models**: The neural network models (RNN and LSTM) demonstrate superior performance at lower τ values compared to the constant allocation strategy, particularly in terms of mean returns. This suggests that when the distribution of price buckets is smaller, neural networks can effectively capture short-term patterns and outperform the simpler constant strategy. However, at larger τ, the constant models show more stability, as the increased complexity of the neural network models leads to diminishing returns, likely due to overfitting and the challenges of modeling more complex distributions.

- **RNN vs. LSTM Models**: LSTM models generally offer marginally better mean returns compared to RNNs at the same τ, reflecting their ability to better capture temporal dependencies. This suggests that LSTM's enhanced memory capabilities make them more suitable for liquidity provision strategies, though both models still perform best with lower τ.

- **Effect of Risk Aversion (a)**: The impact of the risk aversion parameter a is most noticeable at lower τ values. Higher risk aversion does not consistently lead to better risk-adjusted returns. In some cases, extreme conservatism ($a = 100$) hampers performance, indicating that overly cautious strategies may forgo potential gains without significantly reducing downside risk.

5.6 Conclusion

This analysis highlights the nuanced performance of liquidity provision strategies across different model types and parameter settings. While the constant allocation model provides stability and consistency at higher τ values, both RNN and LSTM models show their strength at lower τ, achieving better mean returns by leveraging their capacity to learn short-term patterns in price movements. These results underscore the importance of carefully selecting model architectures and bucket sizes based on the nature of the market conditions.

The performance of neural network models, however, was constrained by computational power limitations, despite code optimization efforts. The complexity of these models—particularly with larger τ—requires substantial computational resources for training and evaluation. This limitation hindered more extensive exploration of the parameter space and more sophisticated model tuning.

Additionally, the technical challenges of implementing these strategies in a real-world decentralized finance (DeFi) environment remain significant. The inherent latency in blockchain transactions delays the execution of trades, making it difficult to react to fast-moving market conditions. While faster execution is possible, it comes with the downside of higher transaction costs, which can erode the profitability of the strategy. This trade-off between latency and cost must be carefully managed when deploying strategies on decentralized platforms.

Relying solely on price signals for model inputs proved insufficient in highly volatile conditions. Future strategies should integrate exogenous factors—such as market sentiment, macroeconomic indicators, and on-chain data—to provide early warning signals and improve model adaptability. These additional data sources could enhance returns by enabling more proactive reallocation of liquidity in response to market shifts.

Recommendations:

- **Model Selection**: Given their strong performance at lower τ, RNN and LSTM models should be favored in scenarios where short-term patterns are more relevant. For larger τ and more stable conditions, the constant model remains a competitive option due to its simplicity and lower computational burden.

- **Computational Resources**: To fully unlock the potential of neural network models, future work should invest in greater computational capacity. Distributed and cloud-based systems could allow for more comprehensive model training and evaluation, particularly when dealing with larger datasets and more complex architectures.

- **Incorporation of Exogenous Parameters**: Expanding the models to include exogenous parameters beyond price signals will likely improve performance, especially in unpredictable markets. Integrating macroeconomic data, sentiment analysis, and on-chain metrics could offer earlier insights and provide a more robust framework for decision-making.

- **Technical Challenges and Costs**: The trade-off between blockchain transaction speed and cost must be accounted for in future strategies. Optimizing this balance—whether through adaptive strategies that adjust transaction timing based on market conditions or through the use of more efficient execution methods—could enhance the overall returns of the strategy.

- **Transformer Models**: Future research should explore transformer-based architectures, which may offer superior performance in modeling price movements and liquidity provision strategies compared to RNN and LSTM models. Transformers are well-suited to capturing long-range dependencies and may provide new opportunities for optimizing strategies in decentralized markets.

6 Conclusion

6.1 Summary of Findings

This report provides an in-depth analysis of liquidity provision strategies on Uniswap v3, encompassing market analysis, the creation of a backtesting environment, and the optimization of strategies through deep learning models. The market analysis identified several key factors influencing liquidity provision, including the evolution of trade volumes, the behavior of active liquidity, and liquidity concentration during periods of market stress. The analysis also evaluated the efficiency of Uniswap v3, confirming its resilience under typical market conditions while revealing potential vulnerabilities during market disruptions, such as the collapse of FTX. Additionally, it was observed that hyper-concentrated liquidity mints positioned just below the current price during bear markets can act as a protective mechanism, mitigating liquidity risks by anticipating downward price movements.

The backtesting environment was developed to replicate the mechanics of Uniswap v3, allowing for accurate simulations of liquidity operations, such as mints, burns, and swaps. It also incorporated features that enable stress testing under different market conditions. Our testing validated that the environment provided reliable simulations of historical performance, allowing for a robust evaluation of liquidity provision strategies.

The optimization of liquidity provision strategies using deep learning models demonstrated that recurrent neural networks (RNNs) and long short-term memory (LSTM) networks can outperform constant allocation strategies when tuned correctly and applied to smaller price buckets. However, the deep learning models faced challenges with larger bucket sizes due to overfitting and the need for significant computational resources. Despite these challenges, the results suggest that neural networks have the potential to enhance liquidity strategies by dynamically adjusting liquidity allocation in response to market movements.

6.2 Future Directions

Looking ahead, several potential avenues for future work arise from this project. First, securing more computational resources would allow for deeper exploration of neural network models and better tuning of hyperparameters, particularly at larger price bucket sizes. This could unlock the full potential of deep learning models in liquidity provision strategies.

Second, future strategies should expand beyond price signals by incorporating exogenous parameters, such as macroeconomic indicators, on-chain metrics, and market sentiment. These factors could offer early signals, helping models adapt to sudden market shifts more effectively and improving risk-adjusted returns.

Third, future research should explore the application of more advanced deep learning architectures, such as transformers, which may offer superior performance in handling long-range dependencies in market data. Additionally, the challenges posed by blockchain latency and transaction costs need to be further addressed. Strategies that optimize the trade-off between execution speed and transaction costs could improve overall returns.

Finally, extending the scope of the analysis to other decentralized finance (DeFi) platforms beyond Uniswap v3 could provide a more comprehensive understanding of liquidity provision across different decentralized exchanges. This work establishes a strong foundation for ongoing research in optimizing decentralized finance strategies and demonstrates the potential of deep learning techniques to significantly improve liquidity provision strategies in DeFi markets.

7 Ethical Considerations and ESG in Decentralized Finance

At Forvis Mazars, there is a strong emphasis on promoting Environmental, Social, and Governance (ESG) principles, with sustainability, responsible investment, and corporate social responsibility (CSR) at the heart of their mission. These efforts reflect a commitment to helping both organizations and individuals align their strategies with broader sustainable development goals, balancing economic progress with environmental and social concerns.

7.1 ESG at Forvis Mazars: My Perspective and Participation

During my time at Forvis Mazars, I actively participated in various initiatives aimed at raising awareness on critical ESG topics, particularly related to the environment and CSR. I took part in multiple webinars and dedicated days where we worked closely with local associations focused on sustainability and climate action. These activities were designed not only to inform employees and stakeholders about the importance of ESG but also to encourage tangible actions that positively impact the community and environment.

Forvis Mazars encourages open dialogue and welcomes feedback on their efforts. This open culture allowed me to voice concerns and engage in critical discussions about whether our initiatives were targeting the right issues and being executed in the most effective way. This level of engagement reinforced the importance of continuously questioning and refining our approaches to ensure that our efforts truly contribute to meaningful change.

One of the key learnings from my involvement in these activities was the complexity of balancing environmental goals with the realities of economic and social impacts. We studied the risks associated with the economic transition imposed by regulatory bodies, analyzing how such transitions could affect industries, businesses, and communities. This experience deepened my understanding of the intersection between regulatory frameworks and the need for socially responsible practices. It also highlighted the importance of considering both the economic and social consequences of rapid regulatory changes, particularly for vulnerable populations who may be disproportionately affected.

7.2 Ethical Challenges in Decentralized Finance

In my field of study, decentralized finance (DeFi), ethical considerations are equally vital, especially when it comes to transparency, sustainability, and fairness. DeFi's potential to democratize access to financial services is undeniable, but it also raises significant concerns regarding governance, environmental impact, and inclusivity. The decentralized nature of blockchain technology introduces unique challenges in ensuring that the platforms we build uphold high ethical standards, particularly regarding environmental sustainability and equitable access.

7.2.1 Key Ethical Considerations in DeFi

- **Sustainable Development:** The decentralized finance ecosystem must strive to minimize its environmental impact, particularly given the energy-intensive nature of many blockchain networks. One of my key takeaways from Forvis Mazars is the need for proactive measures to mitigate environmental harm in any industry, including DeFi. As I continue my work in this space, I aim to explore ways to enhance sustainability, such as integrating more efficient consensus algorithms or supporting projects that prioritize green energy usage.

- **Inclusivity and Fairness:** Another significant ethical issue I've encountered is ensuring that DeFi platforms remain accessible to a broad range of users, including those from underserved communities. From my experiences at Forvis Mazars, I learned that inclusivity and social responsibility are not just goals—they are essential components of ethical leadership and governance. In DeFi, this translates to designing user-friendly platforms and educating potential users about the risks and rewards of decentralized finance.

- **Economic and Social Impact:** The economic transition enforced by regulatory measures, which I studied in detail during my time at Forvis Mazars, also applies to the evolving regulatory landscape in DeFi. The introduction of new regulations can have profound effects on financial stability, access to capital, and the livelihoods of those engaged in these platforms. As DeFi continues to evolve, it is critical to consider the broader economic and social impacts of regulatory compliance, ensuring that new rules support innovation while protecting both investors and the broader public.

7.3 Reflection and Future Learning

Through my experiences at Forvis Mazars, I've developed a deeper awareness of the ethical challenges inherent in both traditional and decentralized finance. Engaging in activities that raised awareness about climate change, CSR, and the economic risks associated with regulatory transitions has equipped me with a broader perspective on the responsibilities we hold as financial professionals.

Looking ahead, I am eager to apply what I've learned to the field of decentralized finance. My goal is to not only develop innovative solutions but to do so in a way that aligns with ESG principles, ensuring that my contributions help to create more sustainable, inclusive, and socially responsible financial systems. I am committed to continuing my learning journey, particularly in the areas of environmental sustainability, governance, and ethical leadership, and to playing an active role in driving positive change within the DeFi ecosystem.

Through this continuous reflection on ethical challenges, I remain focused on ensuring that the solutions I help develop not only meet the technical demands of the market but also contribute to a more equitable and sustainable financial future.

References

[1] Hayden Adams, Noah Zinsmeister, and Moody Salem. Uniswap v3 core whitepaper, 2021. Accessed: 2023-09-21.

[2] Zhou Fan, Francisco Marmolejo-Cossío, Ben Altschuler, He Sun, Xintong Wang, and David C. Parkes. Differential liquidity provision in uniswap v3 and implications for contract design, 2022.

[3] Lioba Heimbach, Eric Schertenleib, and Roger Wattenhofer. Risks and returns of uniswap v3 liquidity providers. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, AFT '22. ACM, September 2022.

[4] M I. Jordan. Serial order: a parallel distributed processing approach. June 1985-March 1986.

[5] Christopher Olah. Understanding lstm networks, 2015.

[6] Jurgen Schmidhuber Sepp Hochreiter. Long short-term memory. `https://www.bioinf.jku.at/publications/older/2604.pdf`. 1997.