DEEP REINFORCEMENT LEARNING FOR TRADING IN CRYPTOCURRENCY MARKETS USING LIMIT ORDER BOOK DATA AND SYNTHETIC AUGMENTATION

LLUC PALOU MASMARTI

Thesis supervisor
ARGIMIRO ARRATIA QUESADA (Department of Computer Science)

Degree
Bachelor's Degree in Data Science and Engineering

Bachelor's thesis

Facultat d'Informàtica de Barcelona (FIB)
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona (ETSETB)
Facultat de Matemàtiques i Estadística (FME)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Universitat Politècnica de Catalunya
Facultat d'Informàtica de Barcelona
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona
Facultat de Matemàtiques i Estadística

Degree in Data Science and Engineering

Bachelor's Degree Thesis

Deep Reinforcement Learning for Trading in Cryptocurrency Markets Using Limit Order Book Data and Synthetic Augmentation

Lluc Palou Masmartí

Supervised by Argimiro Arratia Quesada
Departament of Computer Science

January, 2026

Abstract

This thesis researches whether identifiable patterns exist in the market microstructure present in high-frequency limit order book data of cryptocurrency assets such as Bitcoin that contain exploitable information for developing profitable algorithmic trading strategies. The hypothesis posits that market inefficiencies during the price formation process or risk factor exposures manifest as recognizable patterns in limit order book dynamics, enabling profitable trading strategies when exploited by means of machine learning techniques.

The experimental framework proposes a statistically robust validation methodology for temporal data, developing a systematic trading strategy based on two main components: a Vector Quantized-Variational Autoencoder to extract discrete latent representations of market regimes, and a Transformer with Proximal Policy Optimization to learn optimal trading policies. These techniques are complemented by an additional model that learns the temporal dynamics of latent representations to generate synthetic data with the intent of supporting the main strategy learning.

Strategy performance is evaluated comprehensively using latent representations of market regimes, manually derived indicators from the limit order book, hybrid combinations of both, and synthetic latent representations, all under four different transaction scenarios where the order execution mode impacts transaction costs and, therefore, strategy profitability.

During the validation and evaluation stage on test data, the existence of exploitable patterns in market microstructure is demonstrated under partially idealized trading conditions. The most significant results show that the most profitable execution mode is through passive orders (limit orders) rather than a more aggressive (market orders) execution, systematically obtaining substantially superior returns compared to the market benchmark. Furthermore, the strategy learns to arbitrage exchange rebates at opportune moments to increase profitability. Return decomposition reveals that profitability does not stem from anomaly exploitation but from a combination of tactical positioning during market regimes conducive to strategy execution. Profitability lies in liquidity provision when the market experiences strong high-frequency demand.

In validation, the component responsible for learning market representations obtains promising and statistically robust results, although on test data it appears that a hybrid approach is more profitable, it should be noted that this latter result is not as robust. Regarding synthetic data, these prove inadequate for use according to the implemented modeling method, as the simplifications introduced to generate them artificially destroy the market microstructure characteristics present in real data.

Keywords

Algorithmic trading, Limit order book, Reinforcement learning, Deep learning, Market microstructure, Cryptocurrency, High-frequency

Resum

Aquesta tesi investiga si existeixen patrons identificables en la microestructura de mercat present en les dades del llibre d'ordres límit d'actius criptogràfics com el Bitcoin a altes freqüències que continguin informació explotable per desenvolupar estratègies de trading algorítmic rendibles. La hipòtesi postula que les ineficiències del mercat durant el procés de formació de preus o les exposicions a factors de risc es manifesten com a patrons reconeixibles en la dinàmica del llibre d'ordres, permetent estratègies de trading rendibles quan s'exploten mitjançant tècniques d'aprenentatge automàtic.

El marc d'experimentació proposa una metodologia de validació estadísticament robusta per a dades temporals, desenvolupant una estratègia de trading sistemàtica basada en dues components principals: un Autocodificador Variacional-Vectorialment Quantitzat per extreure representacions latents discretes de règims de mercat, i un Transformer amb Optimització de Polítiques Proximals per aprendre polítiques de trading òptimes. Aquestes tècniques es complementen amb un model addicional que aprèn les dinàmiques temporals de les representacions latents amb la intenció de generar dades sintètiques que proveeixin suport a l'aprenentatge de l'estratègia principal.

El rendiment de l'estratègia s'avalua de forma transversal utilitzant representacions latents de règims de mercat, indicadors derivats manualment del llibre d'ordres, combinacions híbrides d'ambdós, i representacions latents sintètiques, tot això sota quatre escenaris de transacció diferents on el mode d'execució de les ordres té un impacte en els costos de transacció i, per tant, en la rendibilitat de l'estratègia.

Durant l'etapa de validació i avaluació en dades de test queda demostrada l'existència de patrons explotables en la microestructura del mercat sota condicions de trading parcialment idealitzades. Els resultats més significatius mostren que el mode d'execució més rendible és mitjançant ordres passives (ordres límit) en lloc d'una execució més agressiva (ordres de mercat), obtenint sistemàticament rendiments substancialment superiors a la referència de mercat. A més, l'estratègia aprèn a arbitrar les recompenses de la casa d'intercanvi en moments idonis per incrementar la seva rendibilitat. La descomposició dels retorns revela que la rendibilitat no prové de l'explotació d'anomalies sinó d'una combinació de posicionament tàctic durant règims de mercat propicis per a l'execució de l'estratègia. La rendibilitat rau en la provisió de liquiditat quan el mercat experimenta fortes demandes a alta freqüència.

En validació, el component encarregat d'aprendre representacions del mercat obté resultats prometedors i robustos estadísticament, tot i que en dades de test sembla que una aproximació híbrida és més rendible, cal remarcar que aquest últim resultat no és tan robust. Referent a les dades sintètiques, aquestes demostren ser inadequades per al seu ús segons el mètode de modelatge implementat, ja que les simplificacions introduïdes per generar-les artificialment destrueixen les característiques de microestructura de mercat presents en les dades reals.

Paraules Clau

Trading algorítmic, Llibre d'ordres límit, Aprenentatge per reforç, Aprenentatge profund, Microestructura de mercat, Criptomonedes, Alta freqüència

Resumen

Esta tesis investiga si existen patrones identificables en la microestructura de mercado presente en los datos del libro de órdenes límite de activos criptográficos como el Bitcoin a altas frecuencias que contengan información explotable para desarrollar estrategias de trading algorítmico rentables. La hipótesis postula que las ineficiencias del mercado durante el proceso de formación de precios o las exposiciones a factores de riesgo se manifiestan como patrones reconocibles en la dinámica del libro de órdenes, permitiendo estrategias de trading rentables cuando se explotan mediante técnicas de aprendizaje automático.

El marco de experimentación propone una metodología de validación estadísticamente robusta para datos temporales, desarrollando una estrategia de trading sistemática basada en dos componentes principales: un Autocodificador Variacional-Vectorialmente Cuantizado para extraer representaciones latentes discretas de regímenes de mercado, y un Transformer con Optimización de Políticas Proximales para aprender políticas de trading óptimas. Estas técnicas se complementan con un modelo adicional que aprende las dinámicas temporales de las representaciones latentes con la intención de generar datos sintéticos para dar apoyo al aprendizaje de la estrategia principal.

El rendimiento de la estrategia se evalúa de forma transversal utilizando representaciones latentes de regímenes de mercado, indicadores derivados manualmente del libro de órdenes, combinaciones híbridas de ambos, y representaciones latentes sintéticas, todo ello bajo cuatro escenarios de transacción diferentes donde el modo de ejecución de las órdenes impacta en los costos de transacción y, por tanto, en la rentabilidad de la estrategia.

Durante la etapa de validación y evaluación en datos de test queda demostrada la existencia de patrones explotables en la microestructura del mercado bajo condiciones de trading parcialmente idealizadas. Los resultados más significativos muestran que el modo de ejecución más rentable es mediante órdenes pasivas (ordenes límite) en lugar de una ejecución más agresiva (ordenes de mercado), obteniendo sistemáticamente rendimientos sustancialmente superiores a la referencia de mercado. Además, la estrategia aprende a arbitrar las recompensas de la casa de intercambio en momentos idóneos para incrementar su rentabilidad. La descomposición de los retornos revela que la rentabilidad no proviene de la explotación de anomalías sino de una combinación de posicionamiento táctico durante regímenes de mercado propicios para la ejecución de la estrategia. La rentabilidad reside en la provisión de liquidez cuando el mercado experimenta fuertes demandas a alta frecuencia.

En validación, el componente encargado de aprender representaciones del mercado obtiene resultados prometedores y robustos estadísticamente, aunque en datos de test parece que un enfoque híbrido es más rentable, cabe remarcar que este último resultado no es tan robusto. Referente a los datos sintéticos, estos demuestran ser inadecuados para su uso según el método de modelado implementado, ya que las simplificaciones introducidas para generarlos artificialmente destruyen las características de microestructura de mercado presentes en los datos reales.

Palabras Clave

Trading algorítmico, Libro de órdenes límite, Aprendizaje por refuerzo, Aprendizaje profundo, Microestructura de mercado, Criptomonedas, Alta frecuencia

Contents

Introduction

Motivation

The adoption of data-driven algorithmic trading strategies has radically transformed the investment industry over the past decades. This revolution has been driven by the confluence of three key factors: the exponential increase in hardware computational power, the massive availability of financial data, and the advancement of statistical and *Machine Learning* (ML) methods. As a result, quantitative and systematic funds have gone from being marginal players to dominating global financial markets. As explained in *Machine Learning for Algorithmic Trading* [23], from which the following excerpts are extracted.

> *"According to the Economist, in 2016 systematic funds became the largest driver of institutional trading on the US stock market. In 2019 they accounted for more than 35 percent of the institutional volume, up from 18 percent in 2010."*

The magnitude of this paradigm shift is clearly reflected in the sector's figures:

> *"The market research firm Preqin estimates that almost 1,500 hedge funds carry out the vast majority of their trades with the help of computational models. Quantitative funds are now responsible for 27 percent of all US stock trades by investors, up from 14 percent in 2013."*

Moreover, the exceptional performance of some pioneering funds has also contributed to this growth:

> *"Systematic strategies that are mostly or exclusively based on algorithmic decision-making are best known for their introduction by the mathematician James Simons, who founded Renaissance Technologies in 1982 and made it the flagship of quantitative investment firms. Its secretive Medallion Fund, which is not available to outsiders, has earned an estimated annualized return of 35 percent since 1982."*

The Vanguard of Algorithmic Trading

In this context of growth in algorithmic trading, *High-Frequency Trading* (HFT) represents the most advanced technological frontier, where investment decisions are made and executed in fractions of a second, far exceeding human capabilities in terms of execution speed and information processing.

> *"HFT refers to automated trades in financial instruments that are executed with extremely low latency in the microsecond range and where participants hold positions for very short periods. The goal is to detect and exploit inefficiencies in the market microstructure, the idiosyncratic infrastructure of trading venues."*

The weight of these trading strategies in global financial markets has grown significantly:

> *"HFT has grown substantially over the past 10 years and is estimated to make up roughly 55 percent of trading volume in US equity markets and about 40 percent in European equity markets. HFT has also grown in futures markets to roughly 80 percent of foreign-exchange futures volumes and two-thirds of both interest rate and Treasury 10-year futures volumes."*

From Return Drivers to Price Discovery

The *Modern Portfolio Theory* (MPT) has established that asset returns are determined by their exposure to different systematic risk factors. Beyond the traditional *Capital Asset Pricing Model* (CAPM), which considered only the market risk, academic research and industrial practice have identified numerous additional factors that explain asset returns.

> *"The ongoing discovery and successful forecasting of risk factors that, individually or in combination with other risk factors, significantly impact future asset returns across asset classes is a key driver of the surge in ML in the investment industry."*

These return drivers, also called risk factors, were initially labeled as "anomalies" because they contradicted the *Efficient Market Hypothesis* (EMH), representing systematic inefficiencies that can be exploited:

> *"These risk factors were labeled anomalies since they contradicted the EMH..."*

While classic factors such as value, momentum or size have been identified on timescales of weeks or months, market microstructure contains patterns and anomalies operating in significantly shorter time windows. These very short-term inefficiencies are observable in the dynamics of the *Limit Order Book* (LOB), and are intrinsically linked to the price discovery mechanism. The LOB thus captures the real-time interplay between liquidity provision and consumption, forming the informational substrate on which HFT strategies operate.

The *Financial Background* chapter develops each of these financial theoretical concepts in detail, providing the necessary foundations to understand the conceptual framework on which the rest of the work is built.

Problem Statement

Despite the strong performance of algorithmic trading funds, the mass adoption of algorithmic trading and the theoretical foundations linking asset returns with systematic risk factors and even market microstructure anomalies, the following question remains open: can a retail trader, versed in data science and ML tools, design a systematic and profitable strategy capable of exploiting similar patterns?

This project addresses this issue by limiting its scope to the data present in the LOB, which collects the liquidity and microstructural dynamics of financial markets, key elements in the price discovery mechanism. The research analyzes whether *Deep Learning* (DL) and *Reinforcement Learning* (RL) methods can model these dynamics to identify structures or anomalies of this kind and whether these can be translated into profitable trading decisions.

Thus, the research question can be reformulated as follows:

> *"Can limit order book data, modeled using deep learning and reinforcement learning techniques, be used to design a profitable systematic trading strategy capable of identifying and exploiting such market structures or anomalies?"*

Specifically, the project focuses on the cryptocurrency market's BTC-USD pair. This choice responds to the substantial liquidity of the market, continuous operation 24/7, and, fundamentally, the availability of granular LOB data through streaming APIs from various exchanges, a significant advantage over traditional markets where access to this type of data is usually expensive and restricted.

Hypothesis

The evolution of technology and the availability of real-time data has led to the development of a wide range of algorithmic trading strategies, each with different objectives, operative horizons and sources of profitability.

Among the best known are: collocation strategies, which aim to minimize execution latency in order to exploit timing advantages relative to other market participants; arbitrage, which attempts to exploit temporary discrepancies between the theoretical price and the market price; pairs trading, based on the convergence of stationary relationships between assets; and smart factor investing, which seek systematic returns derived from observable risk factors, such as the above mentioned, value, momentum, or size.

This set of strategies illustrates how profitability can come from both structural sources (such as systematic risk factors, described by classical models) and opportunistic sources, derived from temporary market microstructural inefficiencies.

Based on the theoretical considerations being exposed in this introduction section and further explained in the *Financial Background* chapter, the project proposes to explore whether an informational edge exists within LOB data and whether it can be effectively exploited using DL and RL techniques.

On the one hand, therefore, the following hypothesis is formulated:

> *"The BTC-USD limit order book data contains microstructural patterns that can be used by deep learning and reinforcement learning models to develop systematic and profitable trading strategies, either by exploiting anomalies or by exposure to risk factors."*

On the other hand, the null hypothesis assumes that such anomalies or exposures to risk factors do not exist, or that they cannot be consistently captured and exploited by the models under consideration.

> *"The BTC-USD limit order book data does not contain microstructural patterns that can be translated into systematic and profitable trading strategies, either through anomalies or exposure to risk factors."*

Throughout the *Designing a ML-driven Trading Strategy* chapter, the hypothesis proposed will be contrasted by the empirical evidence obtained during the experimentation process. Finally, in the *Conclusions* chapter, it will be determined whether the hypothesis can or cannot be rejected, as well as the implications and possible applications that derive from it in the context of market analysis and the design of algorithmic trading strategies based on ML.

Methodology

The project structure reflects a systematic progression from conceptual foundations to empirical validation. After establishing the research question and hypotheses, the work develops through three parts: theoretical background, experimental implementation, and synthesis of findings.

The theoretical foundations are presented through two background chapters addressing the dual knowledge domains essential to the project. The Financial Background chapter establishes the economic framework, covering Modern Portfolio Theory, the Capital Asset Pricing Model, the Efficient Market Hypothesis, performance evaluation metrics (Sharpe ratio, information ratio), and market microstructure theory with emphasis on limit order book dynamics and price formation mechanisms. These theories provide the conceptual framework for understanding when exploitable inefficiencies may exist and how to evaluate the risk-adjusted performance of trading strategies.

The Machine Learning Background chapter develops the technical foundations, focusing on state-of-the-art deep learning and reinforcement learning architectures employed in the strategy. The chapter presents Vector Quantized-Variational Autoencoders for learning compact discrete representations of limit order book snapshots, autoregressive models for temporal dynamics and synthetic data generation, and Transformer architectures for processing sequential market information. Reinforcement learning foundations are established through the formulation of Markov Decision Processes, policy gradient methods, and the Proximal Policy Optimization algorithm used to train the trading agent. These state-of-the-art ML components form the technical toolkit for extracting patterns from high-frequency market data and translating them into executable trading decisions.

The Designing a ML-driven Trading Strategy chapter implements the complete experimental pipeline, progressing from raw data to validated trading policies. The chapter details the experimental framework establishing the validation methodology, the dataset construction from BitMEX limit order book snapshots, the preprocessing pipeline including feature engineering and transformation, the training and validation of representation learning and reinforcement learning components, and the comprehensive evaluation of learned strategies across multiple trading scenarios. This empirical process contrasts the stated hypotheses with observed results, determining whether exploitable microstructural patterns exist and whether they translate into economically viable trading strategies.

The Conclusions chapter synthesizes the main findings, evaluates which methodological components contribute meaningfully to final strategy performance, assesses economic viability under operational constraints, and identifies limitations and future research directions. The chapter determines whether the proposed hypothesis can be rejected based on empirical evidence and discusses the implications for algorithmic trading strategy design using machine learning techniques.

Financial Background

This chapter on financial background contains the main theories and frameworks that support the financial conception of the stated problem and hypothesis introduced, and that therefore give rise to the resolution in the form of implementation proposed in the Designing a ML-driven Trading Strategy chapter.

Readers who already have a good conceptual background in finance can skip this chapter. Conversely, readers interested in understanding the financial reasoning behind the development of the trading strategy are invited to let themselves be lost in the following introductory notes on the subject of finance.

With regards to the content of this notes, in first place, they introduce the Modern Portfolio Theory (MPT), which seeks to answer the fundamental question of how to simultaneously manage the risk and return of a basket of financial assets. Mainly, this is done through the formulation of a constrained optimization problem, in other words, solving mean-variance optimization. Modern Portfolio Theory is not concerned with explaining the origin of asset returns, but only with the optimal management of assets through the analysis of two key components: risk and return.

In comparison, the Capital Asset Pricing Model (CAPM) states that asset returns are governed by exposure to the market as a whole, this exposure is known as market risk, in addition to other risk factors, which were identified as anomalies before being denoted as such. Then, the Capital Asset Pricing Model establishes that the expected return of an asset depends linearly on its exposure to this systematic market risk. Introducing also the distinction between returns that can be explained by exposure to known risk factors and returns that exceed this explanation, a fundamental concept for understanding value generation in active management.

Following the matter's literature, the Efficient Market Hypothesis (EMH) proposes that in ideal competitive markets, asset prices reflect all available information, and therefore, it is not possible to consistently obtain superior returns by exploiting this information. The Efficient Market Hypothesis is formulated in three forms according to the type of information considered: weak, semi-strong, and strong. Despite the stated facts, empirical evidence suggests that markets are "mostly efficient" as there exist anomalies such as the aforementioned risk factors and short time inefficiencies observed in the market microstructural level.

The microstructure of markets focuses on the specific mechanisms by which trading is carried out and prices are formed: the institutional details of how orders are sent, matched, and executed. Unlike aggregate models that make ideal assumptions, the microstructure recognizes the existence of transaction costs, bid-ask spreads, informational asymmetries and finite latency in information processing. These aspects have a great impact on very short time horizons, which can lead to transitory inefficiencies during the continuous price discovery process. The limit order book, which records all pending buy and sell orders, becomes central to understanding these dynamics and represents a rich source of information on liquidity, order flow and the intentions of market participants.

Finally, the chapter presents useful metrics for evaluating portfolio performance, including the Sharpe ratio, the information ratio and a framework for assessing portfolio management called the Fundamental Law of Active Management. These metrics provide the fundamental tools to measure and compare the risk-adjusted performance of different strategies, and in particular, the Fundamental Law of Active Management, breaks down performance into two components: the quality of the informative insights and the frequency of the management decisions.

Modern Portfolio Theory

The Modern Portfolio Theory (MPT), developed by Harry Markowitz (1952) [35], marked a turning point in the way financial investment was understood. Before Markowitz, investors tended to evaluate assets in isolation, selecting those that individually offered the best expected returns. Markowitz's revolutionary contribution was to show that this approach was suboptimal: what really matters is not how each asset behaves separately, but how they behave together within a portfolio.

Principle of Diversification

The fundamental intuition behind this theory is that diversification can reduce risk without necessarily sacrificing performance. Markowitz mathematically formalized how the imperfect correlation between assets allows fluctuations in the returns of one asset to partially offset those of another. This phenomenon, known as the diversification effect, means that the risk of a well-diversified portfolio may be lower than the weighted average of the individual risks of its components.

The key lies in recognizing that the risk of an asset is not an intrinsic property of that isolated asset, but depends on how its returns covariate with the returns of other assets in the portfolio. For example, an asset that individually may seem risky can, in fact, reduce the overall risk of the portfolio if its returns are negatively correlated with those of other assets.

Mean-Variance Optimization

Markowitz modeled the portfolio construction problem as a constrained optimization problem using two main magnitudes: expected return (mean) and risk (variance of returns).

- **Expected Return:** For a portfolio of n assets with weights w_i (where $\sum_{i=1}^{n} w_i = 1$), the expected return of the portfolio is:

$$\mu_p = \sum_{i=1}^{n} w_i \mu_i,$$

  where μ_i is the expected return of the i-th asset.

- **Risk:** Portfolio risk, measured as the variance in returns, is determined by:

$$\sigma_p^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \sigma_{ij},$$

  where $\sigma_{ij} = Cov(r_i, r_j)$ is the covariance between the i-th and j-th asset returns. This expression captures the effect of diversification: when $\sigma_{ij} < \sigma_i \sigma_j$ (imperfect correlation), the variance of the portfolio is less than that which would be obtained if the assets were perfectly correlated.

  In matrix form, this calculation can be expressed compactly as:

$$\sigma_p^2 = \mathbf{w}^T \mathbf{\Sigma} \mathbf{w},$$

  where $\mathbf{w}$ is the weight vector and $\mathbf{\Sigma}$ is the covariance matrix of the returns of the assets.

The Modern Portfolio Theory posits that rational investors seek to maximize the previously mentioned expected return for a given level of risk, or equivalently, minimize the risk for a target level of return. This compromise between risk and performance is formalized in the following optimization problem:

$$\min_{\mathbf{w}} \quad \sigma_p^2 = \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w}, \quad s.t. \quad \mathbf{w}^T \boldsymbol{\mu} = \mu_{\text{target}} \quad \text{and} \quad \mathbf{w}^T \mathbf{1} = 1,$$

where $\boldsymbol{\mu}$ is the vector of expected returns, μ_{target} is the desired target return, and the constraint $\mathbf{w}^T \mathbf{1} = 1$ ensures that the weights add up to the unit (full capital investment).

This constrained problem can be solved using convex optimization techniques, and the solution provides the optimal portfolio composition based on the investor's objective.

Efficient Frontier

The resolution of the constrained optimization problem for different μ_{target} values generates a set of optimal portfolios that, represented in the risk-reward space, form the efficient frontier. This frontier is the geometric locus of all portfolios that maximize performance for each risk level, or that minimize risk for each performance level.

Markowitz showed that the efficient boundary is shaped like a concave curve in the (σ_p, μ_p) space. All portfolios located on this frontier are efficient in the Pareto sense: it is not possible to improve performance without increasing risk, nor reducing risk without decreasing performance. Portfolios located below the border are suboptimal, as there is always an efficient portfolio that offers more return for the same risk or less risk for the same return.

The efficient frontier provides a framework for portfolio management: an investor can select the portfolio on the frontier that best fits their risk tolerance, thus optimizing the risk-return compromise according to their individual preferences.

To conclude with this section, despite its mathematical elegance and influence on modern financial theory, the Modern Portfolio Theory has significant practical limitations. It requires estimating expected returns and their covariances, which are inherently uncertain magnitudes. In addition, it assumes that returns follow a normal distribution and that investors only care about the mean and variance, simplifications that are often not met in reality. Despite these limitations, the Modern Portfolio Theory remains the conceptual starting point for most modern portfolio management strategies.

Capital Asset Pricing Model

As exposed in the previous section, Modern Portfolio Theory provides a framework for building optimal portfolios that maximize performance for a given level of risk, but it does not address a fundamental question: how are the expected returns of assets determined? The Capital Asset Pricing Model (CAPM), independently developed by William Sharpe (1964) [45], John Lintner (1965) [30], and Jan Mossin (1966) [36], extends the Modern Portfolio Theory to provide a new one that explains the pricing of assets in capital markets.

While Markowitz focused on how an individual investor should build their portfolio, the Capital Asset Pricing Model examines what happens when all investors follow the principles of Modern Portfolio Theory and the market reaches a state of equilibrium.

The Capital Asset Pricing Model is based on several key assumptions that define the scenario of market equilibrium: it assumes all investors are risk-averse and optimize their portfolios according to mean-variance optimization; investors have homogeneous expectations, that is, they share the same estimates of expected returns and correspondng covariances; it assumes that the market is perfect, in other words, there are no frictions, transaction costs, taxes, or restrictions on short selling; and finally introduces the notion of a risk-free asset that all investors can access. Moreover, assumes that all assets are perfectly divisible and liquid.

Under these idealized conditions, the market reaches an equilibrium where all assets are correctly priced and no investor has incentives to alter their portfolio.

Market as a Portfolio

In a market equilibrium scenario, all rational investors maintain the same portfolio of risky assets, the market portfolio, which contains all available assets weighted by their market capitalization. This portfolio represents the aggregate market consensus on the optimal composition of risky assets.

In this setup, the Capital Asset Pricing Model states that the relevant risk of an individual asset is not its isolated volatility (the variance of its returns), but its contribution to the risk of the market portfolio. This contribution is measured by its beta:

$$\beta_i = \frac{Cov(r_i, r_m)}{Var(r_m)},$$

where r_i is the return of the i-th asset and r_m is the return of the market portfolio. This beta captures the asset's systematic risk, the portion of its risk that is correlated with market movements and therefore cannot be eliminated through diversification.

Hence, the Capital Asset Pricing Model establishes that, in a market equilibrium scenario, the expected return of an asset depends linearly on its beta:

$$\mathbf{E}[r_i] = r_f + \beta_i(\mathbf{E}[r_m] - r_f),$$

the term $\mathbf{E}[r_m] - r_f$ is the market risk premium, the compensation that investors demand for taking on the risk of the market portfolio rather than keeping the risk-free asset. The product $\beta_i(\mathbf{E}[r_m] - r_f)$ is therefore the risk premium of the i-th asset, proportional to its exposure to systematic market risk.

This relationship implies a crucial distinction between two types of risk: systematic risk, captured by beta, is correlated with market movements and cannot be eliminated through diversification; in contrast, idiosyncratic risk, specific to each asset, can be diversified by maintaining a broad portfolio. Then, according to the Capital Asset Pricing Model, the market only remunerates systematic risk, idiosyncratic risk is not compensated because investors can eliminate it at no cost through diversification.

From Anomalies to Risk Factors

In practice, observed returns are often broken down as:

$$r_i = \alpha_i + r_f + \beta_i(r_m - r_f) + \epsilon_i,$$

where α_i is the alpha of the asset, that is, the excess return not explained by market exposure, and ϵ_i is the idiosyncratic risk. According to the Capital Asset Pricing Model, in a market equilibrium scenario, α_i should be zero for all assets. A persistent positive alpha would indicate risk-adjusted outperformance, anomalies that contradict the model or suggest that the market is not in perfect equilibrium.

It is precisely in this alpha term that financial research has identified numerous anomalies, patterns of returns that are not explained by the Capital Asset Pricing Model. Banz (1981) [2] and Reinganum (1981) [41] documented the size effect: small firms tend to generate higher returns than the ones explained by the model based only on their beta. Basu (1977) [3] identified the value effect: stocks with low price-earnings ratios consistently outperform stocks with high ratios, even controlling for beta. Jegadeesh and Titman (1993) [25] demonstrated the momentum effect: stocks with recent good returns tend to continue outperforming in the short term.

The identification and analysis of these anomalies led to the development of multifactor models that extend the Capital Asset Pricing Model to incorporate additional sources of systematic risk. Fama and French (1993) [15] proposed a three-factor model that adds size (smb, small minus big) and value (hml, high minus low) to the main theory market factor:

$$\mathbf{E}[r_i] - r_f = \beta_{i,m}(\mathbf{E}[r_m] - r_f) + \beta_{i,smb}\mathbf{E}[smb] + \beta_{i,hml}\mathbf{E}[hml].$$

This model recognizes that asset returns depend not only on exposure to the market, but also on exposure to these additional risk factors. Subsequently, Carhart (1997) [7] added a fourth momentum factor, and Fama and French (2015) [14] extended the model to five factors incorporating profitability and investment.

The interpretation of these factors continues to be the subject of debate. From a rational perspective, factors represent legitimate risk premia: investors demand compensation for assuming exposure to these systematic risks. From a behavioral perspective, factors may reflect cognitive biases or limits to arbitrage that prevent these anomalies from being arbitrated away. In any case, the existence of these factors suggests that the original Capital Asset Pricing Model, with a single market factor, is an oversimplification of reality.

Efficient Market Hypothesis

The Capital Asset Pricing Model establishes that, in a market equilibrium scenario, no asset should generate persistent alpha, that is, higher returns that are not explained by systematic risk exposure. But this statement raises a fundamental question: why do asset prices so accurately reflect their systematic risk? What prevents arbitrage or mispricing opportunities from persisting?

The Efficient Market Hypothesis (EMH), formulated by Eugene Fama in his seminal work of 1970 [16] and posterior work of 1991 [13], provides an answer to these questions. The hypothesis is not related to an asset valuation model but a statement about the behavior of prices in ideal competitive markets: asset prices fully reflect all the information available at any given time. This efficient incorporation of information into prices is what guarantees, according to the Efficient Market Hipothesis, that the market is constantly in equilibrium or tends rapidly towards equilibrium after the arrival of new information.

Market Efficiency Under Question

The classic definition of Fama states that a market is efficient if prices "fully reflect" all available information. Formally, this means that it is not possible to use this information to consistently obtain superior returns, after adjusting for risk and transaction costs. In other words, there are no persistent arbitrage opportunities or trading strategies based on public information that consistently generate alpha.

Fama distinguished three forms of market efficiency according to the type of information reflected in prices, each with different implications for the chances of obtaining superior returns.

- **Weak-form efficiency:** In this form, prices reflect all the information contained in past price history. This implies that technical analysis[1] cannot consistently generate outward returns. Prices follow roughly a random walk: future changes are unpredictable from past changes. If there was a predictable pattern in prices, market participants would exploit it until the pattern disappeared.
- **Semi-strong form efficiency:** This way extends weak-form efficiency by claiming that prices reflect all publicly available information, not just historical prices. This includes financial reports, corporate announcements (earnings, dividends, merges), macroeconomic data, press releases, and any other publicly available information. The implication is that fundamental analysis[2] cannot generate consistent alpha either. Prices adjust so quickly to new public information that, when an investor can act on this information, it is already incorporated into prices.
- **Strong-form efficiency:** The strictest form of the Efficient Market Hypothesis postulates that prices reflect all existing information, including private or insider information. This would imply that even participants with access to confidential unpublished information would not be able to obtain superior returns. This form is the most restrictive and the least empirically plausible: evidence shows that insiders with material private information can earn higher returns, which is why insider trading is regulated and prohibited in most jurisdictions.

[1] Technical analysis relies on the study of patterns in charts of prices, volumes, moving averages, and other indicators based solely on past price data.

[2] Fundamental analysis is an asset valuation technique based on public financial and economic information.

The Efficient Market Hypothesis has been the subject of extensive empirical research with mixed results. On the one hand, there is evidence consistent with efficiency: actively managed funds have difficulties in consistently outperforming market indices after fees and costs, and prices quickly adjust to important corporate announcements. On the other hand, research has also documented numerous anomalies, performance patterns that appear to violate Efficient Market Hypothesis. The size, value, and momentum effects already mentioned in the context of the Capital Asset Pricing Model are examples of these anomalies. Calendar anomalies have also been identified, such as the January effect, and phenomena such as the post-earnings announcement drift, where prices continue to adjust for days or weeks after earnings announcements.

The interpretation of these anomalies is controversial. From the perspective of the Efficient Market Hypothesis, they may represent compensation for risk factors not adequately captured by existing models. From the perspective of behavioral finance, they may reflect systematic cognitive biases or limits to arbitrage that prevent these inefficiencies from being completely eliminated. The debate continues, but the current dominant view is that markets are "mostly efficient", generally efficient, especially for liquid and widely followed assets, but with localized inefficiencies in certain segments, time horizons, or circumstances.

Market Microstructure

The theories and hypothesis presented thus far, Modern Portfolio Theory, the Capital Asset Pricing Model, and the Efficient Market Hypothesis, operate primarily at an aggregate level, examining how portfolios should be constructed, how assets are priced in equilibrium, and how quickly information is incorporated into prices. These frameworks typically abstract away from the specific mechanisms through which trading occurs and prices are actually formed. Market microstructure theory, in contrast, focuses precisely on these mechanisms: the institutional details of how orders are submitted, matched, and executed, and how these processes give rise to the prices we observe.

The study of market microstructure emerged as a distinctive field during the 1970s and 1980s, with seminal contributions from Garman (1976) [17], who modeled the market maker problem, Bagehot (1971) [1], who analyzed the role of information asymmetry in trading, and Glosten and Milgrom (1985) [18], who formalized concepts such as that bid-ask spreads compensate market makers for the risk of trading with informed participants. O'Hara (1995) [37] provided a comprehensive treatment of the theoretical foundations, while Harris (2003) [22] offered a practitioner-oriented perspective on the realities of trading venues and market structure.

The microstructure of markets recognizes that the idealized conditions assumed by aggregate models (frictionless markets, instantaneous price adjustment, information at no cost) are not met in reality. In contrast, real markets exhibit transaction costs, bid-ask spreads, delays in order execution, informational asymmetries, and strategic behavior on the part of different types of participants. These frictions and imperfections are not simply inconveniences to be ignored; are the basis for price dynamics, especially over short time horizons.

Price Discovery Mechanism

At the heart of the microstructure of markets is the process of price discovery, the mechanism by which the scattered information possessed by different market participants is aggregated and incorporated into prices through their trading decisions. Unlike the Efficient Market Hypothesis' implicit assumption of instant price adjustment, price discovery is a continuous, dynamic process that unfolds over time as orders arrive, execute, or cancel.

The limit order book is the primary data structure that organizes this process in modern electronic markets. The limit order book records all pending buy orders (bids) and sell orders (asks) at different price levels, typically sorted by price-time priority: orders with the best prices are executed first, and among orders at the same price, the oldest orders take precedence. The best bid and the best ask, the highest price at which someone is willing to buy and the lowest price at which someone is willing to sell, define the inside spread, a fundamental measure of liquidity and transaction costs.

The state of the limit order book at any given time reflects the current balance between supply and demand at different price levels and serves as a rich source of information about the market. The shape of the limit order book, the distribution of volumes across different price levels, provides information on the depth of available liquidity and potential support and resistance levels. Imbalances between bid and ask volumes can predict price pressure in the short term. The rate and pattern of arrivals, modifications, and cancellations of orders reveal the intentions and expectations of market participants. Large orders may indicate strong conviction; quick cancellations can indicate changes in sentiment or strategic positioning.

The Efficient Market Hypothesis in its semi-strong form suggests that prices should reflect all publicly available information, but this does not imply that prices are adjusted instantaneously every microsecond. The process of incorporating price information through trading requires time (however brief) during which orders must be placed, matched, and executed. During this price discovery process, which can unfold for seconds, milliseconds or even microseconds, transient inefficiencies can arise due to microstructural frictions: the provision of liquidity is costly due to inventory risk and the risk of adverse selection, information is processed with finite latency, and order flow reveals information gradually.

Several empirical studies have documented the predictive content of limit order book data for short-term price movements. Cont, Stoikov and Talreja (2010) [9] developed stochastic models of limit order book dynamics and showed how order flow imbalances predict price changes. Kercheval and Zhang (2015) [27] showed that machine learning models trained with limit order book features could predict short-term price movements with significantly higher accuracy than chance. These results suggest that the limit order book data contains information that is not yet fully reflected in the current price, information that is being processed and will be incorporated into prices over the next few seconds or fractions of a second.

High-Frequency Trading as a Microstructural Opportunity

High-frequency trading strategies are specifically designed to exploit these transient microstructural inefficiencies. Operating on timescales of milliseconds or microseconds, high-frequency trading strategies can detect patterns in limit order book dynamics, order flow, or execution activity that presage short-term price movements and can act on this information before it is fully incorporated into prices by slower market participants.

Importantly, these microstructural inefficiencies are fundamentally different in nature from the anomalies that challenge the Efficient Market Hypothesis over longer horizons. On the one hand, the size, value and momentum effects operate over weeks, months or years and are potentially attributable to risk premiums or behavioral biases. On the other hand, microstructural patterns arise from the mechanics of the trading process itself and typically persist only for very short durations, long enough to be detected and exploited by sufficiently fast strategies, but short enough to be consistent with markets that are "mostly efficient" on longer timescales.

The profitability of high-frequency trading does not necessarily contradict the Efficient Market Hypothesis. As Lo (2004) [32] argues in the Adaptive Markets Hypothesis, market efficiency is not a static property but varies with time, market conditions, and the composition of participants. On very short timescales, the frictions inherent in the trading process create opportunities for participants equipped with superior speed and technology. However, competition between high-frequency trading firms tends to erode these opportunities over time, reducing latencies, tightening spreads, and pushing the frontier of efficiency towards ever-shorter timescales. From this perspective, high-frequency trading can be seen as an extension of the arbitrage mechanism that drives markets towards efficiency: high-frequency trading strategies detect and eliminate microstructural inefficiencies, facilitating the price discovery process and contributing to market efficiency over longer horizons, even while profiting from fleeting inefficiencies over shorter horizons.

How to Measure Portfolio Performance

In the context of investing and trading, evaluating the performance of a strategy requires simultaneously considering both the returns obtained and the risk taken to achieve them. A strategy can generate high returns but take excessive risks, or it can produce modest but consistent gains with low volatility. Therefore, metrics are needed that capture the trade-off between return and risk, allowing strategies to be compared objectively.

Sharpe Ratio

The Sharpe ratio (SR), developed by William Sharpe (1966) [46], is one of the most widely used metrics to evaluate investment strategies. It measures the compensation an investor receives per unit of risk assumed, providing a standardized measure of risk-adjusted return. Formally, the Sharpe ratio is defined as:

$$\text{SR} = \frac{\mu - R_f}{\sigma},$$

where μ is the expected return of the strategy, R_f is the risk-free return, and σ is the standard deviation of the returns. The numerator $\mu - R_f$ represents the excess return above what could be obtained without assuming risk, while the denominator standardizes this excess by the total risk assumed.

A high Sharpe ratio indicates that the strategy generates considerable returns per unit of volatility, while a low ratio suggests that the returns do not adequately compensate for the risk taken.

In practice, expected returns and volatilities are not directly observable and must be estimated from historical data:

$$\hat{\mu} = \frac{1}{T}\sum_{t=1}^{T} r_t, \quad \hat{\sigma}^2 = \frac{1}{T}\sum_{t=1}^{T}(r_t - \hat{\mu})^2,$$

where r_t are the excess returns observed in each time step t, and T is the total number of time steps.

An important aspect of the Sharpe ratio is that it assumes that returns follow a normal distribution or that investors only care about the average and variance of returns. In real markets, returns often have heavy tails, asymmetry, and other characteristics that may limit the validity of the Sharpe ratio as the only evaluation metric [33].

Information Ratio

While the Sharpe ratio measures the total risk-adjusted performance of a strategy, the information ratio (IR) quantifies the specific skill of active management. The information ratio compares the returns of the strategy with those of a benchmark, typically a market index or a passive benchmark strategy. The information ratio is defined as:

$$\text{IR} = \frac{\alpha}{\text{TE}},$$

where α is the alpha of the portfolio (the excess performance over the benchmark) and TE (tracking error) is the standard deviation of this excess performance. Alpha measures whether the strategy generates value above the benchmark, while tracking error quantifies the consistency with which this added value occurs.

A high information ratio indicates that the strategy consistently generates alpha, while a low information ratio suggests that outperforms are volatile or perhaps attributable to random factors rather than the skill of the manager.

The distinction between the Sharpe ratio and the information ratio is fundamental: the former measures the absolute risk-adjusted performance of the strategy, while the latter measures the relative ability of the strategy with respect to an alternative. For active management strategies, IR is particularly relevant because it separates the value added by management decisions from the performance that could be obtained with a passive strategy.

Fundamental Law of Active Management

The Fundamental Law of Active Management (FLAM), developed by Richard Grinold and Ronald Kahn [21], offers a theoretical framework for understanding where superior performance in active management comes from. The law establishes an approximate relationship between the information ratio and two fundamental components:

$$\text{IR} \approx \text{IC} \times \sqrt{\text{breadth}},$$

where the information coefficient (IC) measures the manager's predictive ability, quantified as the correlation between predictions of future returns and actual observed returns. And the Breadth represents the independent number of bets or investment decisions that the manager makes in a given period. It is a measure of the opportunities available to apply the forecasting skills.

The Fundamental Law of Active Management reveals that there are two ways to improve the risk-adjusted performance of a strategy. The first is to improve the quality of predictions (increase IC), developing more accurate models or analyzes that better capture market dynamics. The second is to increase the number of investment opportunities (increase breadth), diversifying investment decisions or increasing the frequency of portfolio rebalancing.

This duality is particularly relevant: even with modest prediction skills, a strategy can generate a high information rato if it has a sufficiently large number of independent investment opportunities.

However, two critical aspects should be highlighted. First, breadth refers to independent decisions, not simply to numerous decisions. If the decisions are highly correlated with each other, the effective breadth is less than the nominal number of decisions. Second, increasing the number of decisions can lead to higher transaction costs and can reduce IC if predictions at shorter time horizons are inherently more uncertain or noisy.

The Fundamental Law of Active Management has been extended by Clarke et al. [8] to incorporate the transfer coefficient (TC), which reflects portfolio constraints and the efficiency with which the manager translates his predictions into portfolio positions. This extension recognizes that even with a high information coefficient and substantial breadth, constraints such as position limits or transaction costs can limit the manager's ability to fully implement their ideas.

The Fundamental Law of Active Management provides a valuable conceptual framework for understanding the factors that determine the performance of active management strategies and for identifying areas where managers can concentrate their efforts to improve results: improving the accuracy of their predictions, increasing the number of independent investment opportunities, or both simultaneously.

Machine Learning Background

This chapter presents the essential machine learning concepts and architectures employed in the trading strategy developed in this work. The exposition is deliberately focused on the specific models and algorithms used, providing the theoretical background necessary to understand the design choices and technical reasoning behind each component.

Readers already familiar with vector-quantized variational autoencoders, causal convolutional architectures, transformer-based sequence modeling, and proximal policy optimization may skip this chapter. For deeper mathematical derivations and broader theoretical context, Appendix A provides supplementary material on deep learning fundamentals and reinforcement learning theory.

The presentation is organized around two complementary pillars that address distinct aspects of the problem. The first pillar concerns representation learning and sequential data processing, focusing on how raw or low-level observations are transformed into structured representations that capture temporal patterns and contextual dependencies. This includes discrete representation learning through vector quantized-variational autoencoders, temporal modeling via autoregressive dilated causal convolutional networks, and context aggregation through transformer-based self-attention mechanisms.

The second pillar addresses sequential decision-making under uncertainty, where an agent learns to select actions that maximize long-term objectives in a dynamic environment. This setting is formalized as a Markov Decision Process, in which policies are learned through reinforcement learning methods such as proximal policy optimization, balancing exploration and exploitation while addressing delayed reward assignment.

Together, these two pillars enable a modular architecture in which learned representations and context-aware encodings define the state space on which decision-making policies operate. This separation allows representation learning, temporal modeling, and policy optimization to be developed and analyzed independently, while remaining tightly integrated within the overall system.

The following sections detail the specific components used in this work: vector quantized-variational autoencoders, autoregressive dilated causal convolutional networks, transformer encoders, and proximal policy optimization, presenting their mathematical formulation and operational role within the strategy.

Deep Learning

In relation to deep learning, the architectures employed in this work are best understood through their learning objectives and the type of knowledge they extract from data (Goodfellow et al., 2016) [19]. Discriminative models learn mappings from input to output data. Generative models, in contrast, aim to learn the underlying data distribution, enabling the synthesis of new samples that resemble the observed data.

Autoencoder-based architectures naturally lie at the intersection of these two paradigms. The encoder learns a compact representation that is useful for inference, while the decoder enforces a generative objective through data reconstruction. Vector-quantized variational autoencoders extend this framework by introducing discrete latent spaces, producing symbolic representations that are both learnable and generative. This discretization enables downstream components to operate on structured, categorical state spaces rather than continuous embeddings.

Beyond representation learning, sequential modeling architectures address temporal structure through different mechanisms and objectives. Causal convolutional networks are employed as purely generative models: their autoregressive and causal structure allows them to synthesize temporally consistent sequences by modeling local dependencies through hierarchical, dilated convolutions.

Transformer encoders, by contrast, are used in a discriminative way. Through self-attention, they construct context-aware representations that capture global dependencies across a sequence. These representations define a structured and informative state space, which is particularly well suited for downstream decision-making tasks such as RL.

Each of these architectures thus serves a distinct mathematical role: reconstruction-based learning, likelihood-driven sequence generation, or contextual representation learning, without conflating their purposes within the discriminative-generative dichotomy.

Vector Quantized-Variational AutoEncoder

In contrast to the continuous latent representation of the VAE (see Appendix A.1.4), subsequent approaches aimed to enhance the expressiveness and interpretability of learned latent spaces. Among these, the Vector Quantized-Variational AutoEncoder (VQ-VAE), proposed by van den Oord, Vinyals and Kavukcuoglu (2017) [38], introduced a discrete latent space by replacing the continuous variables with a codebook of learnable embeddings.

The VQ-VAE approach implements a quantization mechanism that learns richer and more structured representations, reducing the tendency toward overly smooth reconstructions typically produced by VAE models. In addition to this architectural change, the VQ-VAE modifies the standard VAE objective: the loss function not only accounts for the reconstruction loss but also optimizes the codebook and the commitment losses, both of which are contextualized below. Furthermore, unlike the standard VAE, where the prior distribution is predefined, the VQ-VAE requires learning a prior that better reflects the nature of its discrete latent variables.

Discrete Latent Variables. The quantization mechanism is applied immediately after the encoding stage, which generates continuous latent representations $\mathbf{z}_e(\mathbf{x}) = E_\phi(\mathbf{x}) \in \mathbb{R}^{M \times D}$. These representations are then quantized via nearest neighbor look-up in the learned codebook of embeddings $\mathbf{E} \in \mathbb{R}^{K \times D}$, where each embedding $\mathbf{e}_k \in \mathbb{R}^D$. Formally, the quantized latent representation is defined as:

$$\mathbf{z}_{q,i}(\mathbf{x}) = \mathbf{e}_{k_i}, \quad \text{where} \quad k_i = \arg\min_j ||\mathbf{z}_{e,i}(\mathbf{x}) - \mathbf{e}_j||_2^2.$$

It should be noted that this quantization process makes the posterior distribution categorical and deterministic, since each input sample is assigned to the closest learned codebook embedding in a fixed way:

$$q_\phi(\mathbf{z}_i = k_i \mid \mathbf{x}) = \begin{cases} 1, & \text{if } k_i = \arg\min_j \|\mathbf{z}_{e,i}(\mathbf{x}) - \mathbf{e}_j\|_2^2, \\ 0, & \text{otherwise.} \end{cases}$$

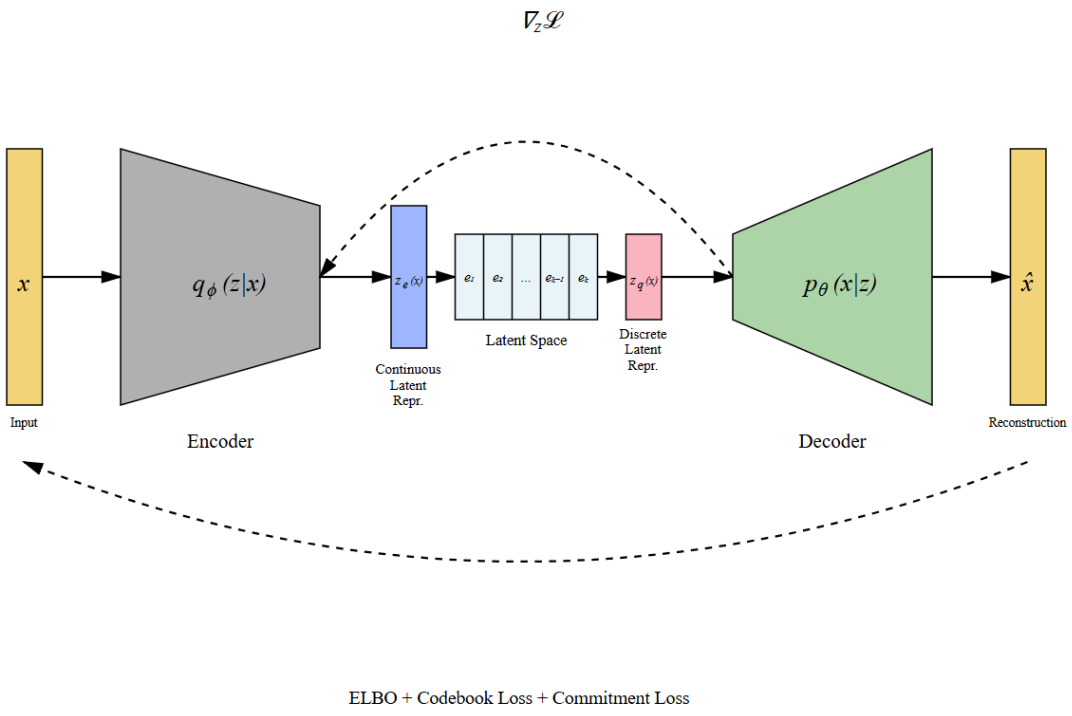


Figure 1: Vector-Quantized Variational Autoencoder architecture. The encoder $q_\phi(z|x)$ produces continuous latent representations $z_e(x)$, which are then quantized via nearest-neighbor lookup in the learned codebook to obtain discrete representations $z_q(x)$. The dashed line represents the straight-through gradient estimator, allowing gradients to flow back to the encoder. The model optimizes a composite loss including reconstruction, codebook, and commitment terms.

Learning. As already mentioned, the optimization objective in the case of VQ-VAE extends the standard VAE loss (detailed in Appendix A.1.4) by incorporating additional terms that take into account the discrete latent structure introduced by the quantization mechanism. Consequently, the total loss to be minimized over the whole dataset can be expressed as the relation among these three components:

$$\mathcal{L}(\mathbf{x}; \boldsymbol{\phi}, \boldsymbol{\theta}, \mathbf{E}) = \mathcal{L}_{\text{rec}}(\mathbf{x}; \boldsymbol{\phi}, \boldsymbol{\theta}) + \mathcal{L}_{\text{codebook}}(\mathbf{x}; \mathbf{E}) + \mathcal{L}_{\text{commit}}(\mathbf{x}; \boldsymbol{\phi}).$$

The reconstruction loss is identical to that of the VAE model and measures how well the decoder can reconstruct the input data from its latent representation:

$$\mathcal{L}_{\text{rec}}(\mathbf{x}; \boldsymbol{\phi}, \boldsymbol{\theta}) = -\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}_q(\mathbf{x})).$$

The codebook loss updates the embeddings in the codebook so that, in latent space, they remain close to the encoder outputs. Formally, it can be written following the standard form most used in dictionary learning algorithms:

$$\mathcal{L}_{\text{codebook}}(\mathbf{x}; \mathbf{E}) = \frac{1}{M} \sum_{i=1}^{M} \left\| \text{sg}[\mathbf{z}_{e,i}(\mathbf{x})] - \mathbf{e}_{k_i} \right\|_2^2,$$

where the stop-gradient operator sg prevents gradients from propagating through the encoder, ensuring that only the codebook embeddings are updated.

Finally, the commitment loss encourages the encoder outputs to remain close to the chosen codebook embeddings. Analogously to the previous case, gradients are not applied to embeddings here:

$$\mathcal{L}_{\text{commit}}(\mathbf{x};\phi) = \frac{\beta}{M}\sum_{i=1}^{M}\|\mathbf{z}_{e,i}(\mathbf{x}) - \text{sg}[\mathbf{e}_{k_i}]\|_2^2,$$

where β is a hyperparameter that controls the strength of this commitment.

Together, these three components of the loss ensure that the model jointly optimizes accurate reconstruction, the learning of meaningful discrete representations, and a consistent alignment between these and the codebook embeddings.

Prior. After having established the main differences in terms of architecture and optimization objective, as discussed previously, it is time to examine the last distinctive component: the prior distribution. While the standard VAE approach (see Appendix A.1.4) is based on the use of a continuous and predefined prior, typically Gaussian, the VQ-VAE replaces it with a categorical distribution over discrete latent codes. Since the latent space consists of a finite set of codebook embeddings, the prior must be explicitly learned to enable generation of new samples.

During training, however, the prior is assumed to be uniform over the K codebook embeddings. This assumption simplifies the optimization process, as the Kullback-Leibler divergence term present in the standard VAE objective becomes constant and can be omitted from the loss function:

$$p(\mathbf{z}) = \text{Cat}(\pi_1, \pi_2, \ldots, \pi_K), \quad \text{where} \quad \pi_k = \frac{1}{K}.$$

This fact implies that, in order to generate new samples of the underlying process being modeled, it is necessary to learn the prior distribution $p(\mathbf{z})$ over the discrete latent codes through an additional modeling process. This prior learning is addressed through autoregressive modeling, as detailed in the following subsection on causal convolutional networks.

Transformers and the Attention Mechanism

Transformers are neural network architectures that process sequential data through self-attention mechanisms rather than recurrence or convolution, enabling parallel computation across the entire sequence and direct modeling of long-range dependencies (Vaswani et al., 2017) [49]. In contrast to recurrent neural networks, which process sequences sequentially and struggle with long-range dependencies due to vanishing gradients, and convolutional architectures (see Appendix A.1.2), which require multiple layers to connect distant positions, transformers allow each position to directly attend to all other positions in a single operation. This fundamental difference enables more effective capture of global contextual relationships in sequential data.

The architecture consists of an encoder that maps input sequences to contextualized representations and a decoder for autoregressive generation, though many applications employ only the encoder component for representation learning, regression, and classification tasks. The encoder's ability to produce context-aware representations where each element incorporates information from the entire sequence makes it particularly well-suited for discriminative tasks requiring global understanding of sequential patterns.

Self-Attention Mechanism. The fundamental building block of the transformer is the self-attention mechanism, which computes a weighted representation of the input sequence where each element attends to all other elements based on their learned relevance. Formally, given

an input sequence $\mathbf{X} \in \mathbb{R}^{n \times d}$ with n tokens[3] (sequence length) and d (embedding dimension), the attention mechanism first projects the input into three representations through learned linear transformations:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V,$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d_k}$ are the query, key, and value matrices respectively, with d_k denoting the key/query dimension (typically $d_k = d/h$ where h is the number of attention heads), and $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d_k}$ are learnable weight matrices. The attention mechanism then computes a similarity score between each query and all keys, normalizes these scores through softmax, and uses them to compute a weighted sum of the values:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V},$$

where the scaling factor $1/\sqrt{d_k}$ prevents the dot products from growing too large in magnitude, which would push the softmax function into regions with extremely small gradients. This scaled dot-product attention allows each position to attend to all positions in the input sequence, with attention weights learned to focus on relevant context. The resulting attention weights form a matrix that explicitly represents the learned dependencies between all pairs of positions in the sequence, providing interpretability regarding which elements the model considers relevant for each output.

Multi-Head Attention. Transformers employ multi-head attention, which runs multiple attention mechanisms in parallel with different learned projections, allowing the model to jointly attend to information from different representation subspaces at different positions. For h attention heads (number of parallel attention mechanisms), each with dimension d_k (key/query dimension per head), the outputs are concatenated and projected:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}_O,$$

where $\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_Q^i, \mathbf{K}\mathbf{W}_K^i, \mathbf{V}\mathbf{W}_V^i)$ represents the i-th attention head with its own weight matrices $\mathbf{W}_Q^i, \mathbf{W}_K^i, \mathbf{W}_V^i \in \mathbb{R}^{d \times d_k}$, and $\mathbf{W}_O \in \mathbb{R}^{hd_k \times d}$ is the output projection matrix that combines all heads back to the original embedding dimension d. This multi-head design enables the model to capture different types of relationships simultaneously: some heads may focus on local context, others on global patterns, with the relative importance of each learned from data.

Encoder. The transformer encoder consists of a stack of L identical layers (encoder depth), each containing two sub-layers: a multi-head self-attention mechanism and a position-wise FFN (detailed in Appendix A.1.1). Each sub-layer is wrapped with a residual connection followed by layer normalization:

$$\mathbf{h}^{(l)} = \text{LayerNorm}(\mathbf{h}^{(l-1)} + \text{Sublayer}(\mathbf{h}^{(l-1)})),$$

where $l \in \{1, \ldots, L\}$ indexes the encoder layer, $\mathbf{h}^{(l)}$ denotes the hidden representation at layer l, and $\text{Sublayer}(\cdot)$ represents either the multi-head attention or the FFN. The position-wise FFN consists of two linear transformations with a ReLU activation:

$$\text{FFN}(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2,$$

[3] In the context of transformers and the attention mechanism, a token denotes a discrete unit in the input sequence, such as a word, subword, character, or encoded feature, mapped to a vector representation through an embedding layer.

where $\mathbf{W}_1 \in \mathbb{R}^{d \times d_{\text{ff}}}$, $\mathbf{b}_1 \in \mathbb{R}^{d_{\text{ff}}}$, $\mathbf{W}_2 \in \mathbb{R}^{d_{\text{ff}} \times d}$, $\mathbf{b}_2 \in \mathbb{R}^{d}$, with d_{ff} denoting the feedforward layer dimension (typically $d_{\text{ff}} = 4d$), applied identically to each position separately, effectively implementing two 1×1 convolutions. The combination of self-attention and feedforward layers allows the model to first aggregate information across the sequence through attention, then process each position's aggregated representation through a shared nonlinear transformation.

Positional Encoding. Since the attention mechanism itself has no notion of sequence order, transformers incorporate positional encodings that are added to the input embeddings to inject information about token positions. These can be fixed sinusoidal functions of different frequencies or learned embeddings, enabling the model to leverage sequential ordering while maintaining the benefits of parallel computation. Without positional information, the transformer would be invariant to permutations of the input sequence, losing the temporal or spatial structure that is often essential for sequential data understanding.

The transformer encoder processes the entire input sequence in parallel, producing contextualized representations where each token's representation incorporates information from all other tokens weighted by their learned relevance. This architecture has become dominant for sequential data modeling tasks, offering superior performance on long sequences compared to recurrent networks while enabling efficient parallelization during training. The decoder component, used in sequence-to-sequence tasks, incorporates an additional masked self-attention layer that prevents positions from attending to subsequent positions, maintaining the autoregressive property necessary for generation tasks.

Autoregressive Dilated Causal Convolutional Network

Originally introduced for raw audio generation, the WaveNet architecture (van den Oord et al., 2016) [39] demonstrates that temporal dependencies in sequential data can be effectively modeled through autoregressive dilated causal convolutional networks. The architectural principles developed for audio waveforms extend naturally to other sequential domains, including the modeling of prior distributions over discrete latent codes. Similar architectures, such as PixelCNN, have been employed in related contexts for learning distributions over discrete representations, establishing this approach as a general framework for autoregressive modeling of categorical sequences.

Autoregressive Factorization. The fundamental principle underlying this approach is the factorization of the joint probability distribution over a sequence of discrete codes into a product of conditional distributions. Given a sequence of latent codes $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N)$ where each $\mathbf{z}_i \in \{1, 2, \ldots, K\}$ represents a codebook index, the joint distribution can be expressed as:

$$p_\psi(\mathbf{z}) = \prod_{i=1}^{N} p_\psi(\mathbf{z}_i \mid \mathbf{z}_{<i}),$$

where $\mathbf{z}_{<i} = (\mathbf{z}_1, \ldots, \mathbf{z}_{i-1})$ denotes all previous codes in the sequence, and ψ represents the model parameters. This factorization allows the model to capture complex temporal dependencies by conditioning each code on the complete history of preceding codes. The model is trained to maximize the likelihood of observed sequences, equivalently minimizing the negative log-likelihood:

$$\mathcal{L}(\psi) = -\mathbb{E}\left[\sum_{i=1}^{N} \log p_\psi(\mathbf{z}_i \mid \mathbf{z}_{<i})\right].$$

This objective corresponds to the cross-entropy loss between the predicted distribution over the K possible codes and the actual code at each position, effectively treating each step as a

K-way classification problem conditioned on the sequence history.

Dilated Causal Convolutions. A critical requirement for autoregressive modeling is that the prediction at position i must depend only on positions strictly before i, ensuring that no information from the future leaks into the conditioning context. Standard convolutional operations (see Appendix A.1.2) violate this causality constraint, as they apply filters symmetrically around each position, incorporating information from both past and future timesteps.

Causal convolutions enforce the temporal ordering constraint by modifying the convolution operation to access only historical information. For a one-dimensional convolution with kernel weights $\mathbf{w} = (w_0, w_1, \ldots, w_{k-1})$ of size k, the causal convolution at position i is defined as:

$$h_i = f\left(\sum_{j=0}^{k-1} w_j \cdot x_{i-j}\right),$$

where f is a nonlinear activation function and x_{i-j} denotes the input at position $i-j$. This formulation ensures that h_i depends only on $x_i, x_{i-1}, \ldots, x_{i-k+1}$, satisfying the causality constraint. In practice, this is achieved through left-padding the input sequence by $k-1$ positions before applying the convolution, such that the receptive field of position i extends only into the past.

While causal convolutions guarantee valid autoregressive modeling, they present a fundamental limitation: the receptive field grows linearly with network depth. Capturing long-range temporal dependencies would thus require either very large kernels or very deep networks, both of which are computationally expensive and difficult to train. Dilated convolutions address this limitation by introducing gaps in the convolution kernel, allowing the receptive field to grow exponentially with depth while maintaining parameter efficiency.

A dilated convolution with dilation rate d is defined as:

$$h_i = f\left(\sum_{j=0}^{k-1} w_j \cdot x_{i-j\cdot d}\right),$$

where the kernel samples the input at intervals of d positions. When $d = 1$, this reduces to the standard causal convolution. By exponentially increasing the dilation rate across layers (e.g., $d \in \{1, 2, 4, 8, 16, \ldots\}$), the receptive field grows exponentially while the number of parameters remains linear in depth. This exponential growth enables the model to condition predictions on long temporal contexts without the computational burden of extremely deep architectures or large kernels, making it feasible to capture dependencies spanning hundreds or thousands of timesteps.

Gated Activation Units. In addition to dilated causal convolutions, the architecture employs gated activation units inspired by LSTM gating mechanisms. Standard activation functions such as ReLU apply a fixed nonlinearity to the pre-activation values. Gated activations, in contrast, use a multiplicative mechanism that allows the network to dynamically control information flow based on the input.

For each layer, the gated activation computes two parallel pathways through separate convolutions, then combines them through element-wise multiplication:

$$\mathbf{z} = \tanh(\mathbf{W}_f * \mathbf{x}) \odot \sigma(\mathbf{W}_g * \mathbf{x}),$$

where $\mathbf{W}_f$ and $\mathbf{W}_g$ are convolution kernels for the filter and gate pathways respectively, $*$ denotes the (causal, dilated) convolution operation, $\tanh$ is the hyperbolic tangent function,

σ is the sigmoid function, and $\odot$ represents element-wise multiplication. The tanh pathway produces candidate features, while the sigmoid pathway produces gating values in $(0, 1)$ that determine which features to propagate. This multiplicative gating allows the model to learn which temporal patterns are relevant for prediction at each position, providing greater representational flexibility than fixed activation functions.

Residual and Skip Connections. The complete model architecture begins with an embedding layer that maps each discrete code $\mathbf{z}_i \in \{1, \ldots, K\}$ to a continuous vector representation $\mathbf{e}_i \in \mathbb{R}^d$ through a learned embedding matrix $\mathbf{E} \in \mathbb{R}^{K \times d}$. These embedded representations are then processed through a stack of residual blocks, each containing a dilated causal convolution with gated activations followed by a residual connection:

$$\mathbf{h}^{(l)} = \mathbf{h}^{(l-1)} + \text{GatedConv}^{(l)}(\mathbf{h}^{(l-1)}),$$

where $\mathbf{h}^{(0)}$ corresponds to the embedded input sequence. The residual connections facilitate gradient flow through the deep network, enabling stable training of models with many layers.

The final layer projects the hidden representations to logits over the K possible codes:

$$\boldsymbol{\ell}_i = \mathbf{W}_{\text{out}} \mathbf{h}_i^{(L)} + \mathbf{b}_{\text{out}},$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{K \times d}$ and $\mathbf{b}_{\text{out}} \in \mathbb{R}^K$ are learned parameters. These logits are converted to a probability distribution through the softmax function:

$$p_\psi(\mathbf{z}_i = k \mid \mathbf{z}_{<i}) = \frac{\exp(\ell_{i,k})}{\sum_{k'=1}^{K} \exp(\ell_{i,k'})},$$

providing the conditional distribution required for the autoregressive factorization. Through this architecture, the model learns to capture complex temporal patterns in discrete sequences while maintaining computational efficiency through dilated convolutions and the causal structure necessary for valid sequential generation.

Reinforcement Learning

Reinforcement learning addresses sequential decision-making under uncertainty, where an agent learns to optimize long-term objectives through interaction with a dynamic environment (Sutton and Barto, 2018) [47]. Unlike supervised learning, where correct actions are provided as labeled examples, reinforcement learning agents discover effective strategies through trial and error, receiving evaluative feedback in the form of rewards that may be delayed in time.

The fundamental challenge lies in the credit assignment problem: determining which actions contributed to eventual outcomes when rewards appear long after decisions are made. The agent's actions not only determine immediate outcomes but also influence future states and opportunities, requiring reasoning about long-term consequences rather than isolated predictions.

Among the various approaches to reinforcement learning, policy gradient methods optimize decision-making policies directly by performing gradient ascent on expected cumulative reward. Proximal policy optimization represents an advancement in this family, introducing constrained policy updates that maintain learning stability while ensuring consistent improvement. This constraint mechanism addresses a critical challenge in policy gradient methods: preventing destructive updates that can destabilize training, particularly important when agents operate on learned, structured state representations.

Proximal Policy Optimization

Building upon the policy gradient framework (see Appendix A.2.2), Proximal Policy Optimization (PPO), introduced by Schulman et al. (2017) [43], represents a significant advancement in policy gradient methods, offering a practical and robust approach to policy optimization. PPO was developed as a more accessible alternative to Trust Region Policy Optimization (TRPO) [44], which introduced the concept of constraining policy updates to ensure stable learning. While TRPO achieves this through a computationally expensive constrained optimization problem that limits the Kullback-Leibler divergence between consecutive policies, PPO accomplishes similar stability guarantees through a simpler clipped objective function.

Trust Region Constraint. The fundamental challenge that both TRPO and PPO address is the instability that can arise from large policy updates. When the policy changes too drastically between iterations, the agent may enter regions of the state-action space where the value estimates are inaccurate, leading to destructive updates and performance collapse. TRPO formalized this concern by constraining updates such that:

$$\mathbb{E}\left[D_{\mathrm{KL}}(\pi_{\boldsymbol{\theta}_{\mathrm{old}}}(A_t|S_t)\|\pi_{\boldsymbol{\theta}}(A_t|S_t))\right] \leq \delta,$$

where δ is a small constant and D_{KL} denotes the Kullback-Leibler divergence. This constraint ensures that the new policy remains close to the old policy in terms of probability distributions over actions.

PPO simplifies this approach by directly modifying the objective function rather than adding an explicit constraint. The algorithm introduces a probability ratio:

$$r_t(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(A_t|S_t)}{\pi_{\boldsymbol{\theta}_{\mathrm{old}}}(A_t|S_t)},$$

which measures how much the new policy differs from the old policy for the action actually taken. The clipped surrogate objective is then defined as:

$$\mathcal{L}_{\mathrm{clip}}(\boldsymbol{\theta}) = \mathbb{E}\left[\min\left(r_t(\boldsymbol{\theta})\hat{A}_t, \mathrm{clip}(r_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right],$$

where $\hat{A}_t$ is the advantage estimate at time t, and ϵ is a hyperparameter that controls the clipping range. The clipping operation ensures that the probability ratio remains within $[1-\epsilon, 1+\epsilon]$, effectively limiting how much the policy can change in a single update.

Advantage Estimation. PPO typically employs Generalized Advantage Estimation (GAE) [42] to compute $\hat{A}_t$, which balances bias and variance through an exponentially weighted average of temporal-difference residuals:

$$\hat{A}_t = \sum_{l=0}^{T-t} (\gamma\lambda)^l \delta_{t+l},$$

where $\delta_t = R_{t+1} + \gamma \hat{v}_{\mathbf{w}}(S_{t+1}) - \hat{v}_{\mathbf{w}}(S_t)$ is the TD error, and $\lambda \in [0,1]$ is the GAE parameter that controls the bias-variance trade-off. When $\lambda = 0$, this reduces to the one-step TD error, while $\lambda = 1$ corresponds to the Monte Carlo return.

Learning. The PPO algorithm optimizes a combined objective function that integrates three key components: the clipped policy loss, the state-value function loss, and an entropy regularization term.

The critic is trained by minimizing the state-value function loss, typically the mean squared error between estimated and target values:

$$\mathcal{L}_{\mathsf{vf}}(\mathbf{w}) = \mathbb{E}\left[(V_t^{\mathsf{targ}} - \hat{v}_{\mathbf{w}}(S_t))^2\right],$$

where $V_t^{\mathsf{targ}} = \hat{A}_t + \hat{v}_{\mathbf{w}_{\mathsf{old}}}(S_t)$ is the target value. Some implementations also include value function clipping to prevent excessively large updates.

To encourage exploration and prevent premature convergence to deterministic policies, an entropy bonus term is added. For discrete action spaces:

$$\mathcal{L}_H(\boldsymbol{\theta}) = \mathbb{E}\Big[H[\pi_{\boldsymbol{\theta}}(\cdot|S_t)]\Big] = \mathbb{E}\left[-\sum_{a\in\mathcal{A}(S_t)} \pi_{\boldsymbol{\theta}}(a|S_t)\log\pi_{\boldsymbol{\theta}}(a|S_t)\right],$$

and for continuous actions:

$$\mathcal{L}_H(\boldsymbol{\theta}) = \mathbb{E}\Big[H[\pi_{\boldsymbol{\theta}}(\cdot|S_t)]\Big] = \mathbb{E}\left[\frac{1}{2}\log(2\pi e\sigma(S_t,\boldsymbol{\theta})^2)\right].$$

The complete objective function combines all components:

$$\mathcal{L}(\boldsymbol{\theta}, \mathbf{w}) = -\mathcal{L}_{\mathsf{clip}}(\boldsymbol{\theta}) + c_1\mathcal{L}_{\mathsf{vf}}(\mathbf{w}) - c_2\mathcal{L}_H(\boldsymbol{\theta}),$$

where c_1 and c_2 are hyperparameters that balance the state-value function loss and entropy bonus relative to the clipped policy loss. This unified objective allows joint optimization of both the actor and critic components while maintaining exploration through entropy regularization.

Training. PPO operates in an on-policy manner with experience replay. The algorithm collects trajectories using the behavior policy $\pi_{\boldsymbol{\theta}_{\mathsf{old}}}$ for a fixed number of time steps T, stores them in a buffer, and then performs multiple epochs of minibatch updates on this data. This reuse of data improves sample efficiency while the clipped objective prevents the policy from deviating too far from the behavior policy that generated the samples. The typical PPO algorithm proceeds as follows:

1. Collect trajectories $\{(S_t, A_t, R_{t+1})\}$ using behavior policy $\pi_{\boldsymbol{\theta}_{\mathsf{old}}}$.

2. Compute advantage estimates $\hat{A}_t$ using GAE.
3. For K epochs:
    - Sample minibatches from the collected data.
    - Update $\boldsymbol{\theta}$ and $\mathbf{w}$ by minimizing $\mathcal{L}(\boldsymbol{\theta}, \mathbf{w})$.
4. Set $\boldsymbol{\theta}_{\text{old}} = \boldsymbol{\theta}$ and repeat.

This design makes PPO significantly simpler to implement than TRPO while maintaining comparable or superior performance across a wide range of continuous control tasks, making it one of the most widely adopted algorithms in modern reinforcement learning applications.

Designing a ML-driven Trading Strategy

After the formulation of the problem at hand and the research hypothesis, this chapter introduces the process of designing, implementing, training, validating and evaluating the proposed HFT strategy. The aim is to move from theoretical approaches on the possible existence of market structures, whether derived from anomalies or exposures to risk factors, to an experimental environment that allows their practical exploitability to be assessed.

The starting point is the definition of a high-frequency algorithmic trading agent that interacts with the market through the LOB. This agent, inspired by the RL approach, makes buying or selling decisions, as well as sizing, based on available liquidity and price information, with the aim of optimizing its behavior according to a reward function that reflects a trade-off between profitability, risk and trading costs.

The set of experiments presented below is based on a completely empirical and data-driven approach. This implies that design decisions, both in the construction of features, in the choice of architectures or in the definition of validation strategies, have been made based on observed and verifiable results. In this context, an experimental framework has been designed that guarantees the reproducibility, statistical robustness and generalization capacity of the models.

As a brief technical contextualization, the experimentation environment has been implemented using:

- MongoDB, as a database for structured storage of LOB and operations data.
- Apache Spark, for efficient processing and grouping of data in large volumes.
- PyTorch, as the main framework for the development of DL models and RL.
- MLflow, for tracking and comparing different experiments and model configurations.

The following sections describe, first, the experimental framework, which defines the overall structure of the workflow, the validation method employed, and the criteria for evaluating the strategy. Next, we delve into the different stages of the iterative implementation process, detailing the components of the pipeline, the intermediate results and the design decisions derived from the empirical analysis.

Experimental Framework

This section establishes the methodological foundation for the HFT strategy development and evaluation. The framework comprises three components: a workflow overview describing the sequential pipeline from raw data to evaluated trading policies, a validation methodology ensuring statistical robustness and generalization, and assessment criteria defining performance metrics across trading scenarios.

Workflow

Strategy development follows a sequential pipeline organized into four stages: dataset construction, preprocessing, model development, and strategy evaluation. Each stage builds on previous stage results, with MongoDB and Apache Spark as primary data processing tools enabling modular operations through hourly batch processing and ensuring straightforward data flow.

The dataset construction stage transforms raw LOB data into structured, temporally-partitioned collections suitable for preprocessing and subsequent modeling. Raw market data is initially recorded through continuous 30-second LOB snapshots, capturing bid-ask price levels and liquidity volumes. This data is stored in a unified MongoDB collection where each document represents a complete temporal sample, providing the foundation for all subsequent processing stages.

Temporal partitioning divides this collection into training, validation, and test sets following the validation methodology specifically designed for time series data. Unlike standard cross-validation that assumes independent samples, this methodology accounts for temporal dependencies by separating training and validation folds with protection zones, preventing information leakage across temporal boundaries. The validation method, in order to assess statistical robustness and generalization, materializes different fold configurations of the same training and validation dataset as separate MongoDB collections, ensuring subsequent preprocessing and modeling stages operate on originally identical data. Dataset construction procedures and implementation are presented in the Dataset section.

The preprocessing stage prepares data for model consumption through two parallel branches. The first branch processes raw LOB snapshots through standardization, logarithmic quantization, and liquidity aggregation procedures, adapting the microstructural information into representations suitable for ML components. The second branch derives manual features including historical returns, volatility proxies, and LOB depth indicators. This last branch undergoes stylized facts analysis examining distributional properties, temporal dependencies, and stationarity characteristics to guide transformation and standardization procedures that address asymmetries and ensure temporal comparability across market regimes. Preprocessing procedures, stylized facts analysis, feature selection criteria, transformation and standardization methods are detailed in the Preprocessing section.

The modeling stage develops ML components to extract LOB and price formation insights from preprocessed features. The first component, an automatic feature extractor based on VQ-VAE architecture, learns discrete market regime representations from standardized LOB snapshots. The second component models the VQ-VAE prior distribution, learning temporal dynamics in the discrete latent space; through autoregressive generation, this component enables synthesis of artificial market sequences that constitute an additional feature source for the downstream trading strategy. The third component consists of a Transformer-based PPO algorithm trained on four feature sources: VQ-VAE representations combined with manually

derived features, manually derived features alone, VQ-VAE representations alone, and VQ-VAE synthetic representations sampled from the learned prior. Model architecture implementations, training procedures, and validation results are presented in the Training and Validation section.

The strategy evaluation stage assesses learned trading policies across four transaction cost scenarios representing different agent-market interaction modes: buy-and-hold (passive baseline), taker (aggressive market orders with 10 bps fees), maker neutral (limit orders with zero fees), and maker rebate (limit orders with 2.5 bps rebates). Performance is measured through Sharpe ratios without baseline (self-contained risk-adjusted returns), Sharpe ratios with buy-and-hold baseline (excess returns beyond passive exposure), and alpha-beta decomposition (separating skill-based returns from systematic market exposure). The synthetic data experiment provides controlled comparison to assess whether learned patterns reflect genuine market structure or artifacts of data generation. Evaluation results and trading policy analysis under the validation method are presented first in the Training and Validation section, followed by hold-out test set assessment in the Final Evaluation section.

Validation Method

The validation method on which the main design decisions and modeling components developed in this work are based follows a structured strategy that combines temporal cross-validation with a separate test set. This approach makes it possible to guarantee an honest assessment in an intrinsically non-stationary environment such as that of the financial markets.

First, the dataset is divided into two distinct parts: a majority set intended for cross-validation and a final test set that is kept completely isolated throughout the entire development process. This test set does not intervene in any decisions related to the preprocessing of the data or the design or training of the models, and is used exclusively as an unbiased measure of performance once the complete pipeline has been validated.

The cross-validation applied to the majority set follows the scheme of combinatorial purged cross-validation with embargo proposed by Marcos López de Prado [34], especially suitable for temporary data. Each partition (split) constructs training and validation periods (folds), separated by purge and embargo zones designed to prevent any leakage of information between periods of different roles. The length of these zones is defined according to the temporal requirements of the features used, ensuring that the validation samples do not depend, either directly or indirectly, on the training data.

This scheme generates multiple partitions that cover different market regimes and allow the evaluation of the temporal robustness of both the models and the artifacts associated with the preprocessing process. For each design decision, whether in the training of pipe artifacts (such as scaling or transformation tools) or in the architectural configuration of the main models of the strategy, the corresponding components are trained and validated independently on each of the partitions. From this process, a performance measure is obtained for each partition, and the aggregation of the metrics, both in terms of average and variability, allows the selection of decisions that show a better capacity for generalization under heterogeneous market conditions.

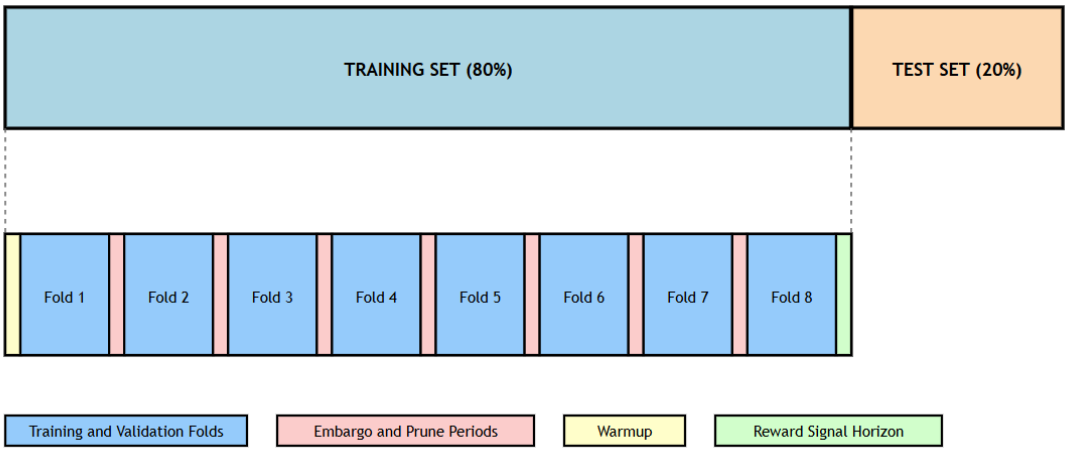


Figure 2: Schematic overview of the validation method used in this work, illustrating in one split the separation between training and validation folds, the test set, as well as the purge and embargo zones applied at fold boundaries, and dataset-level constraints such as the warmup period and reward signal horizon enforced to prevent information leakage.

Once the optimal configurations and architectures have been identified by cross-validation, the specific artifacts and production models for each partition are trained, which are frozen before being applied to the validation data or generating new representations of these required for later stages of the pipeline. This procedure preserves the statistical integrity of the process and ensures a strict separation between training and validation in all phases.

Finally, with the complete pipeline architecture already validated, a final strategy is trained using all available data (previously intended for training and validation), excluding the test set. This strategy is evaluated on completely unseen data, providing a final check of consistency between the behavior observed during cross-validation and actual performance on new data.

In conclusion, this validation scheme, based on multiple partitions of time periods separated by purge and embargo zones, offers a reliable estimation of performance, allows quantification of the uncertainty associated with design decisions and reduces the risk of overfitting both in the selection of architectures and adjustment of artifacts and in the final evaluation of the strategy.

Assessment Criteria

The trading strategy is evaluated across four transaction cost scenarios representing different modes of agent-market interaction through distinct trading operationalities:

- **Buy-and-Hold:** Establishes a passive market exposure baseline by maintaining a constant position with zero transaction costs, capturing pure directional market risk without active trading decisions.
- **Taker:** Represents aggressive execution through market orders that consume liquidity immediately upon submission, incurring transaction costs of 10 basis points (taker fee) plus half the bid-ask spread. This fee structure, deliberately set near realistic exchange values but at the upper range, stress-tests whether the algorithm identifies genuine predictive edges justifying immediate execution costs.

- **Maker Neutral:** Evaluates passive limit order placement where the agent provides liquidity by posting orders at best bid and ask prices, assuming zero transaction costs.
- **Maker Rebate:** Applies the same limit order logic as maker neutral but incorporates a 2.5 basis point rebate per executed trade, reflecting economic incentives exchanges provide to liquidity suppliers.

The maker scenarios abstract from adverse selection risk by assuming limit order fills are guaranteed upon price contact, a simplification that warrants examination in future work.

The RL algorithm's reward function optimizes exclusively for the taker scenario. By penalizing each trade with the full 10 basis point cost plus spread, the training process forces the agent to learn patterns justifying aggressive execution despite substantial fees. This design prioritizes robustness, ensuring any discovered trading edge overcomes harsh execution conditions. Performance under the three alternative scenarios is evaluated post-training without exposure during optimization, decomposing whether returns originate from directional prediction, optimal execution timing, or exploitation of exchange fee structures.

Strategy performance is assessed through three complementary metrics:

- **Sharpe ratio without baseline:** Measures risk-adjusted returns using each scenario's own mean and standard deviation:

  $$\text{Sharpe} = \frac{\mathbb{E}[R]}{\sigma(R)},$$

  where R represents the return series. This provides a self-contained evaluation of performance consistency.

- **Sharpe ratio with baseline:** Uses the buy-and-hold scenario as the baseline:

  $$\text{Sharpe}_{\text{BH}} = \frac{\mathbb{E}[R - R_{\text{BH}}]}{\sigma(R - R_{\text{BH}})},$$

  where R_{BH} denotes buy-and-hold returns. This quantifies value added by active trading beyond passive market exposure, corresponding to the information ratio when buy-and-hold represents the benchmark strategy.

- **Alpha-beta decomposition:** For the best-performing scenario, a regression-based decomposition separates systematic and idiosyncratic return components following the framework presented in the Financial Background chapter. Strategy returns are modeled as:

  $$R_{\text{strategy}} = \alpha + \beta R_{\text{BH}} + \epsilon,$$

  where α represents the constant return component independent of market direction (abnormal returns), and β captures the sensitivity to systematic market movements (systematic risk exposure).

Dataset

The construction of the dataset used in this work is described on the basis of three closely related key aspects: the source of the data, corresponding to the snapshots of the LOB; the temporal coverage and the volume of data available; and the splitting method of the training, validation and final test set, defined in coherence with the validation method established in the section of the experimental framework.

Data Source and Notation

The unique data source for this work is the BitMEX exchange, selected because it offers access to the aggregate L2 level of the LOB, which provides sufficiently detailed information on price levels and liquidity volumes, and has comprehensive technical documentation that facilitates the development of an Information Retrieval System (IRS). Through its public API [5], the complete state of the LOB is periodically reconstructed every 30 seconds, thus generating a sequence of snapshots that represent the continuous evolution of market liquidity.

Each snapshot includes the bid and ask price levels and the associated volumes in dollars, along with a precise timestamp. This structure is used throughout the pipeline and allows the dataset to be formally defined as a temporally indexed collection of LOB snapshots. The data is persisted in Parquet format, organized in daily files, which facilitates its subsequent integration into the feature flow implemented with Spark and the iterative collection system in MongoDB.

To formalize its representation, each state of the LOB retrieved at a time t is expressed as:

$$L_t = \left\{ (p_i^b, v_i^b)_{i=1}^{N_b},\ (p_j^a, v_j^a)_{j=1}^{N_a} \right\},$$

where (p_i^b, v_i^b) denote the price and aggregated volume at the i-th bid level, ordered such that $p_1^b > p_2^b > \cdots > p_{N_b}^b$, and (p_j^a, v_j^a) denote the price and aggregated volume at the j-th ask level, ordered such that $p_1^a < p_2^a < \cdots < p_{N_a}^a$. The full depth of the aggregated L2 level LOB is retrieved, that is, the entire set of available bid and ask levels.

Hence, the complete dataset is defined by the temporally indexed set as follows:

$$\mathcal{D} = \{L_t\}_{t=1}^{T},$$

where T is the total number of snapshots recorded.

Data Coverage and Availability

The temporal coverage of the dataset spans from July 11 to September 25, 2025, encompassing 77 consecutive days of continuous market observation. This period provides approximately 2.5 months of high-frequency trading data, retrieved without major interruptions in the daily collection schedule, ensuring temporal continuity throughout the entire observation window.

In terms of the data volume available, the dataset comprises 210.565 LOB snapshots distributed across 77 daily files. The collection maintains remarkable consistency, with an average of approximately 2.735 snapshots per day and a tight distribution ranging from 2.310 to 2.761 snapshots. Each daily file contains approximately 100 MB of data, reflecting the detailed structure of the LOB snapshots with their associated price levels and volumes.

Regarding sampling characteristics, the 30-second reconstruction frequency theoretically yields 2,880 snapshots per day. The observed retrieval rate of approximately 95% reflects the real-world conditions of the data collection process. Minor gaps in the expected sample count can be attributed to logged connection errors during snapshot retrieval.

Splitting Method

The splitting of the dataset follows a hierarchical structure aligned with the previously defined validation method. At the highest level, the data is divided into a set intended for cross-validation and a test set with an 80-20 ratio. Specifically, the test set is temporarily placed after the cross-validation period, ensuring a strict forward-testing scenario that reflects real deployment conditions.

To prevent any temporal leakage of information, the splitting incorporates several protection mechanisms. Among these, a warmup period equivalent to one hour of observed data stands out that precedes the cross-validation data and allows the initialization of the features derived from past observations. This warmup accommodates both the half-life decay windows used in the standardization stage and the lookback windows required for manually derived feature calculation, ensuring that all past dependent operations are fully initialized before the first sample used in the validation process.

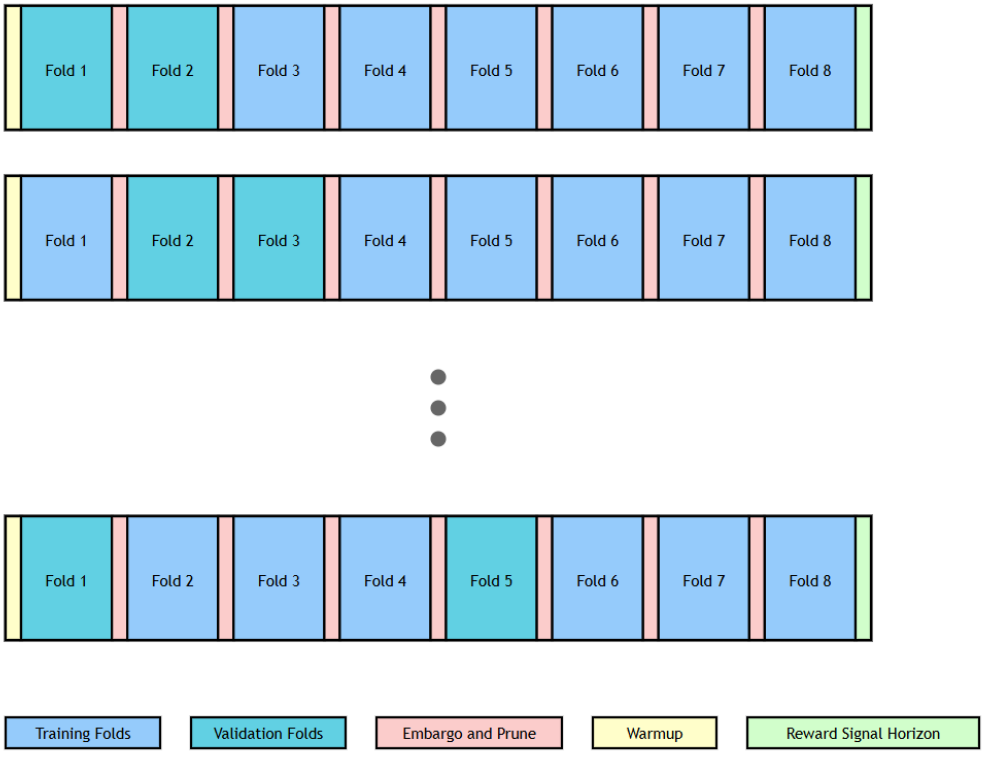


Figure 3: Schematic illustration of the combinatorial purged cross-validation with embargo. Each row represents a different split, with validation folds highlighted, purge and embargo zones applied at fold boundaries, and dataset-level constraints such as the warmup period and reward signal horizon enforced to prevent information leakage.

Between the cross-validation and test sets, an embargo period of the same length as the previous one is applied which creates a temporary cushion, while purge zones of that amount of samples are applied to the boundaries of the folds during cross-validation. These mechanisms prevent the leakage of information in both directions: backwards, through features calculated from past observations, since all standardization half-lives and lookback windows are limited to being significantly shorter than the one-hour time cushion, and forward, through the reward signal, which uses only the immediate return within the protected time frame, with enough margin to be extended if necessary.

The set intended for cross-validation is then subdivided into 8 time folds of equal size, each corresponding to approximately one week of continuous observations. The combinatorial cross-validation scheme with purge and embargo generates 28 different training-validation configurations, also called splits, by systematically selecting 2 validation folds from the 8 available folds. Each configuration applies the purge and embargo mechanisms to the corresponding limits to maintain temporal independence between the training and validation samples.

For stylized facts analysis performed as part of the data characterization process, windows of 10,000 samples are randomly positioned within the safe zone of each fold, defined as the region of the fold that excludes the one-hour side margins. This random placement ensures that statistical evidence is not biased by specific temporal patterns or boundary effects, providing robust characterization of the statistical properties of data across different market regimes.

Finally, the complete configuration of the partitions is preserved in structured metadata that accurately records the time limits, sample indexes, and configuration parameters of all folds and splits used in the validation process. This metadata ensures exact reproducibility of the splitting method, ensuring that any subsequent analysis or model training can faithfully replicate the organization of the dataset, regardless of the implementation details.

Preprocessing

This section presents the preprocessing pipeline that transforms raw LOB snapshots and derives manually engineered features into model-ready data representations. The workflow progresses through three stages: manual feature derivation, feature engineering and selection, and feature transformation and standardization.

The first stage derives 37 features across five categories: price features, volatility proxies, LOB depth and liquidity descriptors, historical returns, and forward returns. The second stage analyzes the stylized facts of these features to assess their statistical properties and justify the reduction to 18 final features based on stationarity, autocorrelation, heteroskedasticity, and cross-correlation patterns. Finally, the third stage applies mathematical transformations to address distributional asymmetries followed by exponentially weighted moving average (EWMA) based standardization to ensure temporal comparability across market regimes.

Additionally, a parallel standardization procedure processes raw LOB snapshots through price standardization, logarithmic quantization, and liquidity aggregation to generate structured representations suitable for DL architectures.

Manual Feature Derivation

Before the incorporation of main ML strategy components, HFT systems relied on manually derived features to capture the dynamics of liquidity and price formation. Indicators related to order book depth, volume imbalances, liquidity concentration, volatility proxies, and historical returns have been widely used in both quantitative practice and microstructure literature [10, 6, 20].

The work incorporates a branch of manually derived features with two objectives: complement ML components learned representations with microstructural descriptors that have proven useful in previous studies, and empirically compare the effectiveness of classical approaches with modern representation learning, evaluating their respective contributions to the strategy's decision quality.

Following the already established microstructure literature, 26 features are derived, organized into five conceptual families, each capturing a relevant dimension of market microstructure or price dynamics. Below, all features are formally stated, grouped according to their theoretical motivation.

- **Price features:** Summarize the immediate state of the top of the order book and provide basic references on price level and variation between snapshots.

$$m_t = \frac{p_1^b(t) + p_1^a(t)}{2} \quad \text{(mid-price)}, \qquad r_t = \log m_t - \log m_{t-1} \quad \text{(log-return)},$$

$$s_t = p_1^a(t) - p_1^b(t) \quad \text{(spread)}, \qquad \mu_t = \frac{p_1^a(t) v_1^b(t) + p_1^b(t) v_1^a(t)}{v_1^b(t) + v_1^a(t)} \quad \text{(microprice)}.$$

- **Volatility proxies:** Quantify the magnitude of recent price fluctuations and act as indicators of short-term risk.

$$\hat{\sigma}_t^2(n) = \frac{1}{n} \sum_{k=1}^{n} r_{t-k}^2 \quad \text{(variance proxy)},$$

$$\hat{\sigma}_t(n) = \sqrt{\hat{\sigma}_t^2(n)} \quad \text{(volatility)}.$$

- **Limit order book depth and liquidity descriptors:** Capture the distribution of volume at different depths and quantify imbalance, concentration, and effective depth. These measures reflect the market's ability to absorb orders and provide information on potential price movements conditioned by the state of liquidity.

  For index sets $K_5 = \{1, 2, 3, 4, 5\}$, $K_{15} = \{1, \dots, 15\}$, $K_{50} = \{1, \dots, 50\}$, and K_{all} representing all available depth:

$$
\begin{aligned}
d_t(K) &= \sum_{i \in K} \left(v_i^b(t) + v_i^a(t) \right) && \text{(depth)}, \\
\delta_t(K) &= \frac{\sum_{i \in K} v_i^b(t) - \sum_{i \in K} v_i^a(t)}{\sum_{i \in K} v_i^b(t) + \sum_{i \in K} v_i^a(t)} && \text{(depth imbalance)}, \\
\lambda_t(K) &= \frac{\sum_{i \in K} w_i \left(v_i^b(t) + v_i^a(t) \right)}{\sum_{i \in K} \left(v_i^b(t) + v_i^a(t) \right)} && \text{(liquidity concentration)}, \\
\pi_t(K) &= \frac{\sum_{i \in K} \left(p_i(t) - m_t \right) v_i(t)}{\sum_{i \in K} v_i(t)} && \text{(price impact proxy)}, \\
\ell_t(K) &= \frac{p^a_{\max(K)}(t) - p^b_{\max(K)}(t)}{m_t} && \text{(liquidity spread)},
\end{aligned}
$$

  where w_i denotes distance-based weights in liquidity concentration, and $K \in \{K_5, K_{15}, K_{50}, K_{\text{all}}\}$ represents the aggregation levels.

- **Historical returns:** Provide temporal information on recent price evolution through multiple lags.

$$
r_{t-k} = \log m_t - \log m_{t-k}, \quad k \in \{1, 2, 3, 4, 5, 10, 20, 40, 60, 120, 240\},
$$

  where each k value represents a time window expressed in number of consecutive snapshots, covering horizons from 30 seconds to 2 hours with a sampling interval of 30 seconds.

- **Forward returns:** Act as the main component of the reward signal, keeping in mind the RL framework, and define the target variable that the models will use to learn patterns related to price formation.

$$
r_t^+ = \log m_{t+1} - \log m_t.
$$

  Only the immediate future return is adopted, corresponding to the next available snapshot. Although this choice simplifies reality by ignoring the minimum delay between observation, information processing, and agent reaction inherent to market environments, it is sufficient for the development and simulation scope.

Taken together, this set of manually derived features provides a rich and interpretable representation of market microstructure, capturing essential dimensions such as price dynamics, volatility, order book depth, and temporal behavior. However, the effective usefulness of each feature cannot be assumed: their statistical properties, potential redundancy, and relationship with known patterns of high-frequency financial series must be analyzed.

For this reason, a stylized facts analysis and feature engineering process is necessary to assess the stability and informativeness of the variables, and determine which features contribute to explaining the underlying process of price formation through liquidity dynamics.

Feature Engineering

This section analyzes the statistical properties of manually derived features to assess their relevance, identify redundancies, and justify the final feature set used in the trading strategy. The analysis focuses on two subsets: historical returns, which have the same properties as forward returns, serving as the reward signal in the RL framework, and non-return features derived from LOB microstructure.

Initial Feature Set. The initial feature derivation process generates 37 features across multiple categories:

- **Price features** (4): mid-price, log-return, spread and microprice.
- **Volatility proxies** (1): volatility.
- **Limit order book depth and liquidity descriptors** (20): depth, depth imbalance, liquidity concentration, price impact proxy, and liquidity spread, each calculated at four aggregation levels ($\{K_5, K_{15}, K_{50}, K_{\text{all}}\}$).
- **Historical returns** (11): past log-return with lags $k \in \{1, 2, 3, 4, 5, 10, 20, 40, 60, 120, 240\}$.
- **Forward returns** (1): forward log-return as the reward signal.

The stylized facts analysis is performed on all 37 features. The detailed discussion and visualizations focus on the final selected features that feed the models, providing statistical justification for retention and exclusion decisions.

Statistical Testing Framework. The analysis employs standard statistical tests to characterize high-frequency financial data properties:

- **Normality**: The Jarque–Bera (JB) test [24] evaluates deviations from normality, identifying the presence of heavy tails and asymmetries.
- **Stationarity**: The Augmented Dickey–Fuller (ADF) and Kwiatkowski–Phillips–Schmidt –Shin (KPSS) tests [11], [29], verify the stability of the mean and variance over time.
- **Linear autocorrelation**: The Ljung–Box (LB) test [31] detects serial correlation in the series.
- **Nonlinear autocorrelation**: The Tsay test [48] captures higher-order correlations and nonlinear dependencies.
- **Conditional heteroskedasticity**: The Lagrange Multiplier test for autoregressive conditional heteroscedasticity (ARCHLM) [12] and the LB test applied to squared returns measure the autocorrelation of variance, indicative of heteroskedasticity.

This testing framework is applied consistently across all feature categories analyzed in the following subsections.

Stylized Facts of Historical Returns. Historical returns are calculated using overlapping rolling windows with appropriate strides applied to each horizon to reduce artificial serial correlation and ensure near-independence between observations.

- **Normality:** The JB test rejects normality at $\alpha = 0.05$ for all horizons $h \in \{1, 2, 3, 5, 10, 20\}$, with mean p-values effectively at zero ($\bar{p} \approx 0.0$) and narrow confidence intervals (CI width $= 0.022$) across test windows.

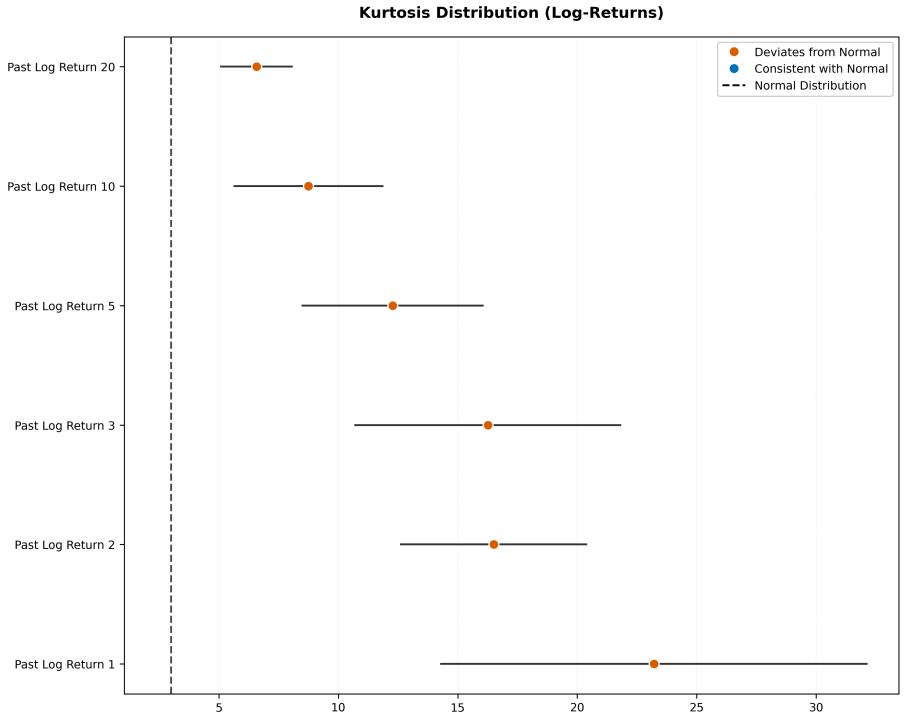


Figure 4: Excess kurtosis of returns across different time horizons. Error lines represent 95% confidence intervals computed across test windows, reflecting temporal variability.

Excess kurtosis declines from $\text{Kurt}(X_t) = 23.21$ at $h = 1$ to 6.57 at $h = 20$, but all values remain substantially above zero (Figure 4). Skewness ranges from $\text{Skew}(X_t) = -0.299$ at $h = 1$ to 0.091 at $h = 20$, remaining close to zero across all horizons. The combination of high kurtosis and near-zero skewness indicates symmetric heavy tails driven by extreme events in both directions.

For horizons beyond $h = 20$, normality by aggregation emerges as predicted by the central limit theorem, with preliminary evidence suggesting convergence around $h = 40$ (20 minutes) and more pronounced Gaussianity at $h \geq 60$ (30 minutes). These longer horizons were excluded from the final feature set due to diminished predictability.

- **Stationarity:** Both ADF and KPSS tests confirm stationarity at $\alpha = 0.05$ for all horizons, with a proportion of 1.0 across test windows and mean p-values ranging from 3.07×10^{-24} for $h = 1$ to 1.24×10^{-4} for $h = 20$. Narrow confidence intervals (CI width $= 0.022$) indicate temporal stability throughout the time series.
- **Linear Autocorrelation:** The LB test reveals decreasing autocorrelation with increasing horizon, exhibiting notable temporal variability (Figure 5). At $h = 1$, the proportion of test windows with significant autocorrelation is 0.774, declining progressively to 0.655 at $h = 2$, 0.500 at $h = 3$, 0.583 at $h = 5$, 0.429 at $h = 10$, and 0.268 at $h = 20$. Beyond $h = 3$, the proportion falls below the majority threshold.

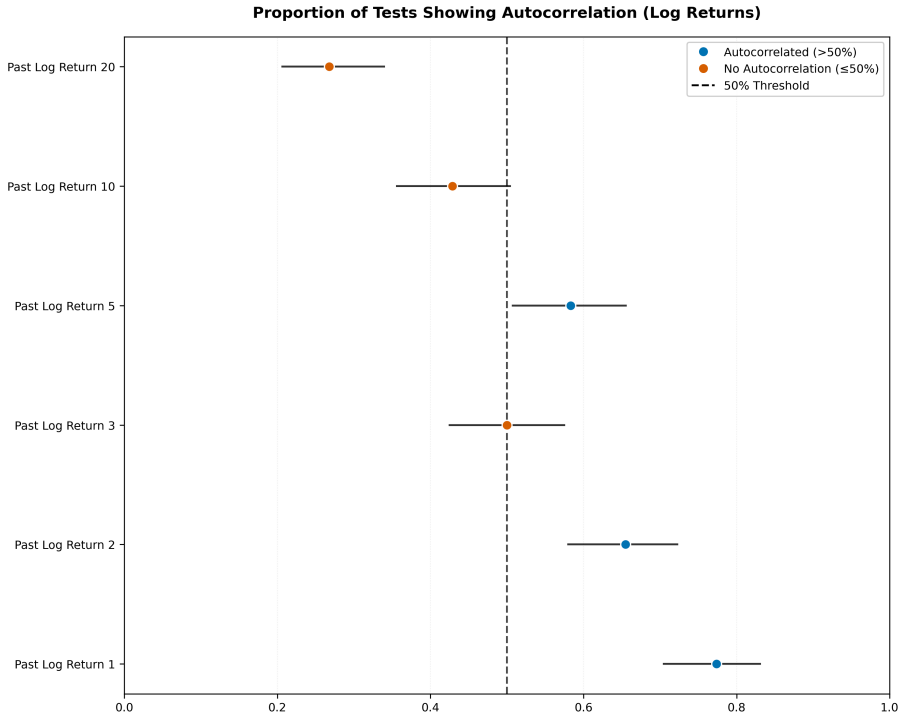


Figure 5: Proportion of test windows exhibiting significant linear autocorrelation across return horizons. Error lines represent 95% confidence intervals, and the dashed line indicates the majority threshold of 50%.

Wide confidence intervals (ranging from 0.126 to 0.150) indicate substantial temporal variability in autocorrelation behavior, contrasting with the temporal stability observed in normality and stationarity tests. This suggests that linear predictability is sensitive to local market conditions.

- **Nonlinear Autocorrelation:** The Tsay test detects significant nonlinear autocorrelation at $h = 3$ (1.5 minutes), indicating that nonlinear relationships persist even when linear autocorrelation is weak.
- **Conditional Heteroskedasticity:** ARCH-LM and LB tests on squared returns reveal strong volatility clustering that decreases monotonically with horizon. At $h = 1$, the proportion of test windows exhibiting significant heteroskedasticity is 0.897 (CI width $= 0.053$), decreasing to 0.857 at $h = 2$ (CI width $= 0.061$), 0.823 at $h = 3$ (CI width $= 0.067$), 0.788 at $h = 5$ (CI width $= 0.071$), 0.661 at $h = 10$ (CI width $= 0.083$), and 0.566 at $h = 20$ (CI width $= 0.087$). The relatively narrow confidence intervals indicate temporal stability of heteroskedasticity.

  Even at $h = 20$, more than half of test windows exhibit significant heteroskedasticity, confirming that volatility clustering remains relevant beyond ultra-short horizons.

At short horizons ($h < 20$), returns exhibit consistent stationarity, rejection of normality with heavy tails, decreasing but temporally variable linear autocorrelation, and strong conditional heteroskedasticity. At $h = 20$ (10 minutes), stationarity persists while autocorrelation becomes minimal (proportion $= 0.268$), with normality still rejected due to heavy tails. Beyond $h = 20$, normality by aggregation emerges and predictability diminishes.

These findings establish a boundary between short-term horizons with exploitable predictive structure and longer horizons converging toward weak-form market efficiency. Consequently, only historical returns with horizons $h \in \{1, 2, 3, 5, 10, 20\}$ are included in the final feature set.

Stylized Facts of Non-Return Features. This subsection analyzes the statistical properties of non-return features derived from the LOB, encompassing volatility proxies and structural indicators such as depth, depth imbalances, liquidity concentration, and price impact proxies calculated at multiple aggregation levels.

- **Normality:** The JB test rejects normality at $\alpha = 0.05$ for all features, with a rejection proportion of 1.0 and mean p-values effectively at zero ($\bar{p} \approx 2.43 \times 10^{-8}$).

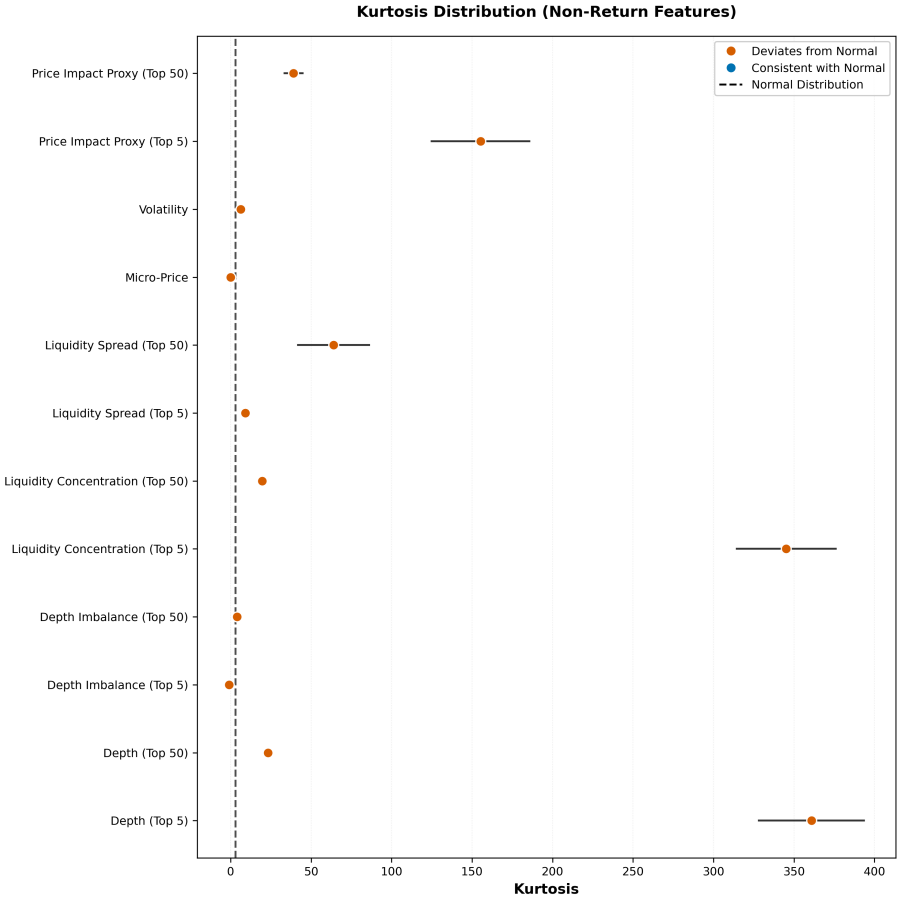


Figure 6: Excess kurtosis of non-return features. Error lines represent 95% confidence intervals computed across test windows, reflecting temporal variability.

Excess kurtosis exhibits a bimodal pattern (Figure 6). Top 5 depth indicators show extreme values: Kurt(depth_top_5) $= 360.89$, Kurt(liquidity_concentration_top_5) $= 345.24$, Kurt(price_impact_proxy_top_5) $= 155.29$, and Kurt(liquidity_spread_top_50) $= 63.94$. In contrast, depth imbalance features exhibit near-zero or slightly negative kurtosis: Kurt(depth_imbalance_top_5) $= -0.93$ and Kurt(depth_imbalance_top_50) $= 3.99$.

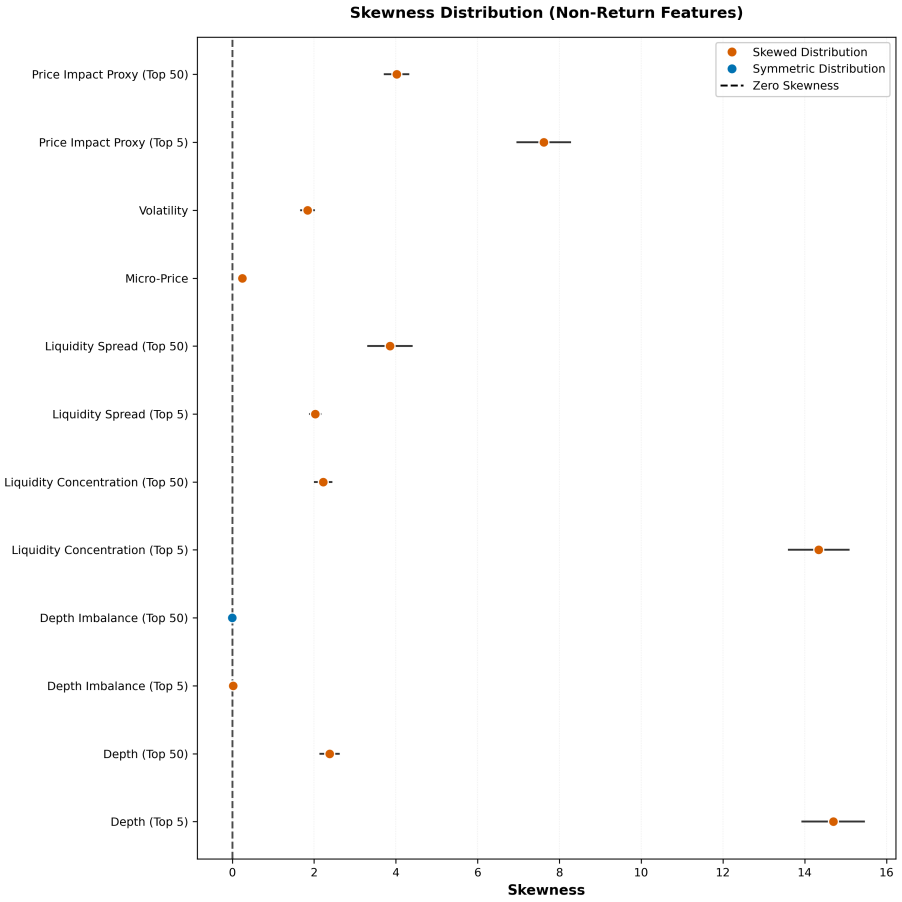


Figure 7: Skewness of non-return features. Error lines represent 95% confidence intervals. Top 5 depth features show strong positive skewness, while other features exhibit low to medium positive values, indicating right-tailed distributions.

Positive skewness is observed across features (Figure 7), with particularly strong values in top 5 depth indicators: Skew(depth_top_5) $= 14.70$, Skew(liquidity_concentration_top_5) $= 14.34$, and Skew(price_impact_proxy_top_5) $= 7.62$. Other features exhibit low to medium positive skewness.

- **Stationarity:** The vast majority of features pass stationarity tests at $\alpha = 0.05$. The notable exception is microprice, which exhibits clear non-stationarity with a stationarity proportion of only 0.060 (95% CI width $= 0.073$) (Figure 8). In contrast, relative measures such as depth imbalances and liquidity concentrations retain full stationarity (proportion $= 1.0$).
- **Autocorrelation:** All non-return features exhibit universal autocorrelation. The LB test at $\alpha = 0.05$ yields a proportion of 1.0 across all features, with mean p-values extremely close to zero ($\bar{p} \approx 1.06 \times 10^{-43}$) and narrow confidence intervals (CI width $= 0.022$). This universal autocorrelation arises from EWMA construction in volatility features, order flow inertia in structural indicators, and overlapping information across consecutive 30-second snapshots.

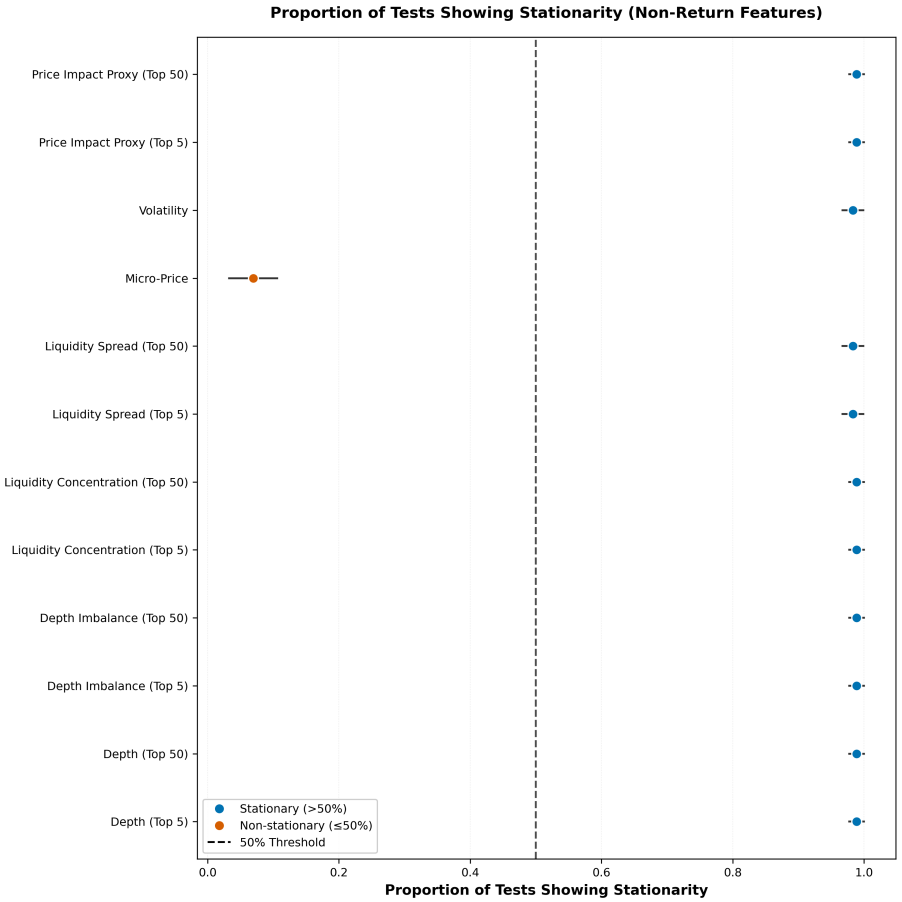


Figure 8: Proportion of windows exhibiting stationarity for non-return features. Error lines represent 95% confidence intervals.

- **Conditional Heteroskedasticity:** Pervasive heteroskedasticity is observed across features at $\alpha = 0.05$. Microprice exhibits the highest proportion at 0.986 (CI width $= 0.020$), followed by volatility at 0.972 (CI width $= 0.029$). Liquidity spread features show proportions of 0.899 for top 5 (CI width $= 0.053$) and 0.817 for top 50 (CI width $= 0.067$). Depth imbalance features show 0.899 for top 50 (CI width $= 0.053$) and 0.845 for top 5 (CI width $= 0.063$). Price impact proxy features exhibit 0.873 for top 50 (CI width $= 0.058$) and 0.756 for top 5 (CI width $= 0.075$). Depth features show 0.845 for top 50 (CI width $= 0.063$) and 0.589 for top 5 (CI width $= 0.086$). Liquidity concentration features show 0.865 for top 50 (CI width $= 0.060$) and 0.591 for top 5 (CI width $= 0.086$).

The analysis confirms that manually derived features reproduce characteristic stylized facts of financial time series: universal rejection of normality with bimodal kurtosis patterns (extreme for top 5 depth features, near-Gaussian for depth imbalances), full stationarity except microprice, universal autocorrelation, and pervasive heteroskedasticity. The differential intensity across feature types justifies targeted transformation and standardization procedures.

Furthermore, the presence of high cross-correlation among depth variants indicates that defining indicators at multiple depth levels introduces unnecessary collinearity without substantial informational gain. A configuration combining shallow depth (K_5), centered around the mid-price to capture local microstructural detail, and broad depth (K_{50}), reflecting a more insightful liquidity state, proves sufficient to obtain an informationally rich yet efficient representation while eliminating the redundant intermediate levels (K_{15} and K_{all}).

Final Feature Set. Based on the stylized facts analysis, the final feature set comprises 18 features:

- **Price features** (1): microprice.
- **Volatility proxies** (1): volatility.
- **Limit order book depth and liquidity descriptors** (10): depth, depth imbalance, liquidity concentration, price impact proxy, and liquidity spread, each calculated at two aggregation levels ($\{K_5, K_{50}\}$).
- **Historical returns** (6): past log-return with lags $k \in \{1, 2, 3, 5, 10, 20\}$.
- **Forward returns** (1): forward log-return as the reward signal.

This reduction from 37 to 18 features removes redundancy while preserving essential market microstructure information.

Feature Transformation and Standardization

This stage applies mathematical transformations to address distributional asymmetries identified across features in the stylized facts analysis, followed by EWMA-based standardization to ensure temporal comparability across market regimes.

Feature Transformation. The transformation of manually derived features addresses the distributional challenges documented in the stylized facts analysis, where substantial asymmetries and heavy tails were observed across multiple feature types. These distributional characteristics can impede numerical stability in optimization algorithms and adversely affect learning dynamics in neural network architectures.

The transformation selection process evaluates standard candidates (identity, logarithmic, square root, inverse hyperbolic sine, Box–Cox, and Yeo–Johnson), each suited to different distributional pathologies. For the readers unfamiliar to this kind of transformations just have to know that the Box–Cox transformation applies a parametric power transformation:

$$y(\lambda) = \begin{cases} \frac{x^\lambda - 1}{\lambda}, & \text{if } \lambda \neq 0, \\ \log(x), & \text{if } \lambda = 0, \end{cases}$$

where the parameter λ is estimated via maximum likelihood on strictly positive data. The Yeo–Johnson transformation extends this framework to handle the full real line through a piecewise power transformation with an analogous maximum likelihood estimated parameter.

The selection method employs the Pearson P/DF statistic as an evaluation metric, defined as the chi-square goodness-of-fit statistic divided by its degrees of freedom:

$$\text{P/DF} = \frac{\chi^2}{df} = \frac{1}{k-3} \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i},$$

where lower values indicate reduced asymmetries and tail heaviness [40].

Transformation selection is embedded within the validation framework. Parameters are fitted on training fold data (10% randomly sampled for computational efficiency) and evaluated on validation fold data to ensure generalization across temporal regimes. For each feature, all candidate transformations are evaluated across all splits, generating a distribution of P/DF scores. The final selection prioritizes either the most frequently selected transformation across splits or the transformation with the best average P/DF performance, depending on result consistency.

Forward log-returns r_t^+ are excluded from transformation to preserve their original scale for use as reward signals in the RL framework. All other manually derived features undergo the transformation selection and application process.

Feature Transformation Impact. Figure 9 presents the normalized reduction in average validation Peterson statistic achieved through transformation across all features. The metric represents the proportional improvement, where values closer to 1.0 indicate near-complete correction of distributional artifacts.

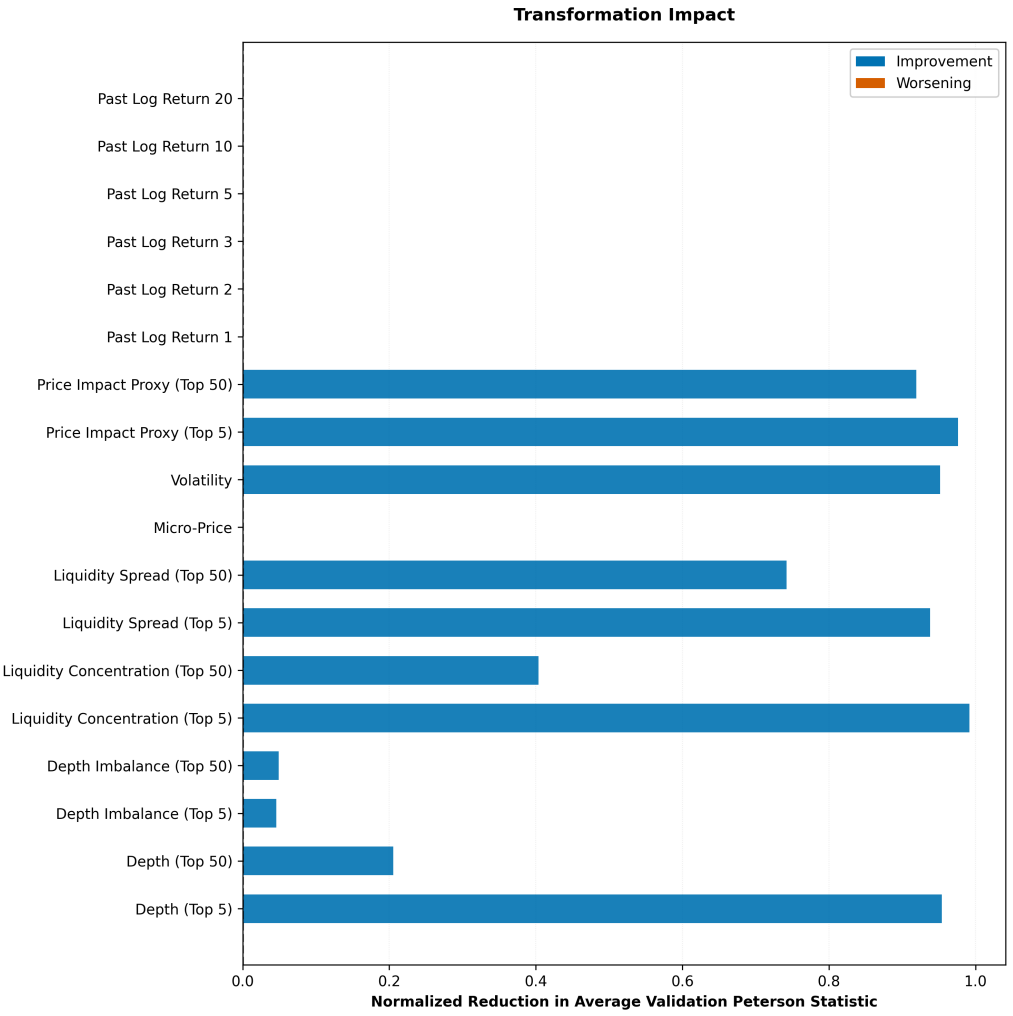


Figure 9: Normalized reduction in average validation Peterson statistic through transformation for all manually derived features. Higher values indicate greater proportional improvement toward reducing asymmetries and heavy tails.

Features exhibiting extreme heavy tails and strong positive skewness demonstrate the most dramatic improvements. Liquidity concentration top 5, with extreme distributional properties (Kurt = 345.24, Skew = 14.34), achieves the highest normalized reduction of approximately 0.99 via Yeo–Johnson transformation, representing near-complete correction from a Peterson statistic of 2402.6 to 19.6. Price impact proxy features, which exhibited extraordinarily high kurtosis (Kurt = 155.29 for top 5, Kurt = 39.12 for top 50), show normalized reductions of approximately 0.98 for top 5 (Box–Cox transformation from 1806.6 to 42.7) and 0.92 for top 50 (Box–Cox transformation from 3217.1 to 259.1). Depth top 5, with extreme kurtosis (Kurt = 360.89) and strong skewness (Skew = 14.70), achieves a normalized reduction of approximately 0.95 via Yeo–Johnson transformation (2470.6 to 113.4).

Liquidity spread and volatility features also show substantial improvements. Liquidity spread top 5 achieves a normalized reduction of approximately 0.94 via Box–Cox transformation (1003.5 to 62.3), while top 50 shows 0.74 via Yeo–Johnson (1366.0 to 351.9). Volatility achieves approximately 0.95 normalized reduction through logarithmic transformation (592.8 to 28.6), reflecting the effectiveness of log transformation for variance-based measures. Depth top 50 shows a more modest but meaningful improvement of approximately 0.21 via logarithmic transformation.

Features with moderate to low initial distributional deviations show minimal benefit from transformation. Historical returns, which displayed moderate excess kurtosis (ranging from 6.57 to 23.21) and near-zero skewness, show essentially no improvement: all past log-return features exhibit normalized reductions below 0.001. Depth imbalance features, which exhibited near-zero or slightly negative excess kurtosis (Kurt = -0.93 for top 5, Kurt = 3.99 for top 50) and minimal skewness, show normalized reductions of approximately 0.05, confirming near-Gaussian properties in their original form. Microprice similarly shows a normalized reduction effectively at zero, as its distributional issues stem primarily from non-stationarity rather than far from normality.

The heterogeneity in transformation effectiveness validates the data-driven selection methodology. Features with extreme distributional pathologies (kurtosis > 100, skewness > 7) benefit dramatically with normalized reductions exceeding 0.90, while those with near-Gaussian properties (kurtosis < 5, skewness < 1) require minimal intervention with normalized reductions below 0.05.

Feature Standardization. After transformation addresses distributional asymmetries, standardization ensures temporal comparability across market regimes through adaptive scaling. The standardization employs EWMA to compute time-varying location and scale parameters:

$$\mu_t = (1-\alpha)\mu_{t-1} + \alpha x_t, \quad \sigma_t^2 = (1-\alpha)\sigma_{t-1}^2 + \alpha(x_t - \mu_t)^2,$$

where $\alpha = 1 - \exp(-\log(2)/h)$ defines the decay rate from the half-life h. The standardized feature is computed as:

$$z_t = \frac{x_t - \mu_t}{\sigma_t}.$$

The half-life selection procedure follows the same validation methodology as transformation selection. For each feature, candidate half-lives ranging from 5 to 80 snapshots (2.5 to 40 minutes) are evaluated across all splits. The optimal half-life balances responsiveness to local regime changes against over-fitting to short-term fluctuations.

Feature Standardization Impact. Figure 10 presents the impact of EWMA standardization on Peterson statistics across all features. The impact differs fundamentally from transformation: while transformation reduced Peterson statistics by correcting distributional asymmetries,

standardization systematically increases Peterson statistics across most features. This apparent degradation reflects EWMA's operational mechanism: exponential smoothing introduces serial autocorrelation by construction, as each standardized value depends on smoothed statistics of previous observations. The Peterson statistic, sensitive to autocorrelation structure, increases even when marginal distributions remain approximately Gaussian.

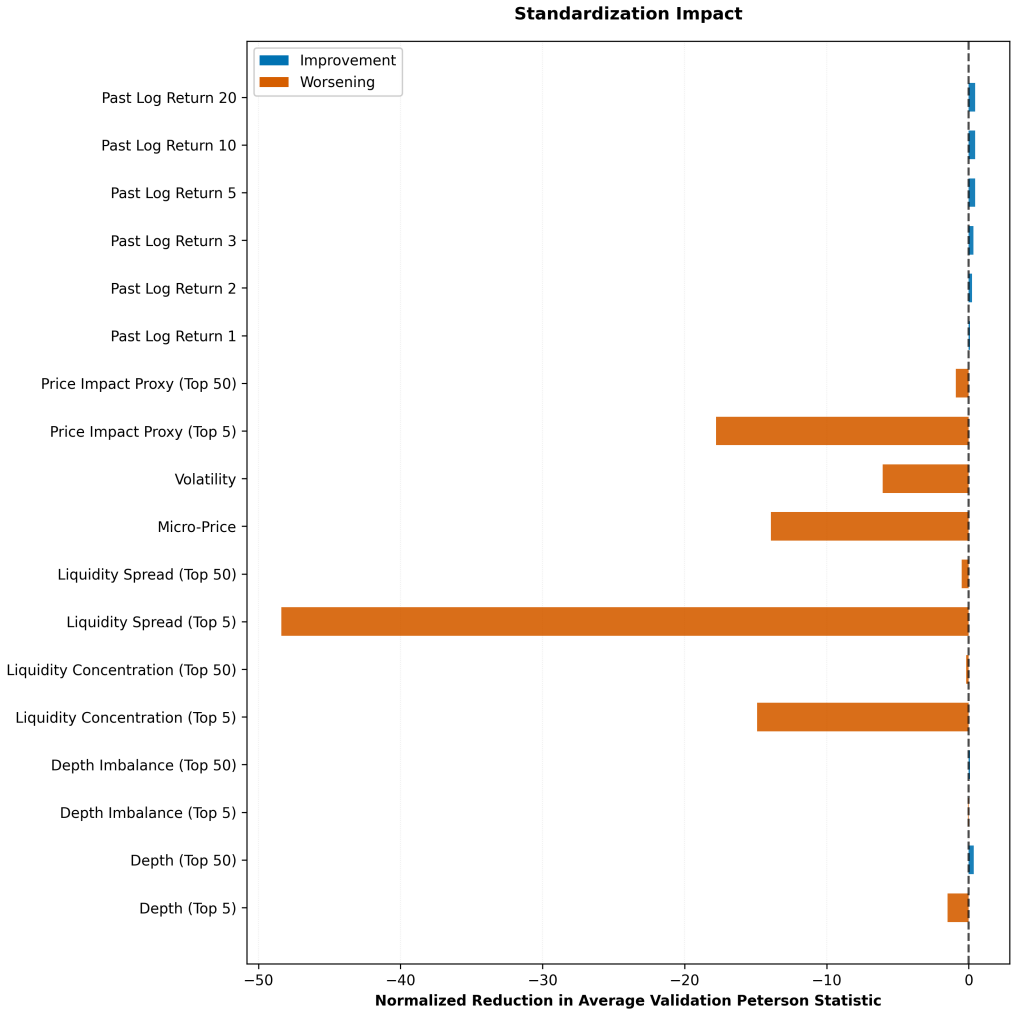


Figure 10: Impact of EWMA standardization on Peterson statistics. The systematic increase across most features reflects the introduction of controlled temporal dependence through exponential smoothing, accepting autocorrelation in exchange for adaptive scaling across market regimes.

Features with strong conditional heteroskedasticity exhibit the largest Peterson statistic increases under standardization. Microprice, volatility, and liquidity spread features show substantial increases, reflecting the introduction of controlled temporal dependence. These features benefit most from adaptive scaling, as their variance instability across time would otherwise create numerical challenges in model training.

Historical returns and depth features show minimal impact under EWMA standardization. These features already possessed moderate kurtosis and the mild adaptive scaling provides beneficial variance stabilization without introducing substantial additional autocorrelation beyond what already exists from their construction.

The divergent behavior validates the dual-stage preprocessing design. Transformation addresses shape artifacts through optimized transformations without introducing temporal dependencies. Standardization addresses scale heterogeneity and regime shifts through adaptive EWMA statistics, accepting controlled autocorrelation in exchange for numerical stability across varying market conditions. The Peterson statistic worsening quantifies the cost of this adaptation, while the benefits manifest in downstream model performance.

The half-life selection procedure, reflects feature-specific temporal characteristics: features with high-frequency dynamics receive shorter half-lives for rapid adaptation, while those with slower regime evolution use longer half-lives for stability.

Stylized Facts on Final Feature Set. To assess the effectiveness of the complete preprocessing pipeline, a final stylized facts analysis is conducted on the fully preprocessed features after both transformation and standardization.

- **Normality - Return features:** The preprocessed return features exhibit mean excess kurtosis of -0.35 (range: -0.71 to 0.83), representing a dramatic reduction from raw data mean kurtosis of approximately 13.8 (range: 6.57 to 23.21). Mean skewness reduces to 0.011 (range: -0.006 to 0.025). All return features now exhibit kurtosis and skewness within narrow ranges around zero.
- **Normality - Non-return features:** The preprocessed non-return features exhibit mean excess kurtosis of 0.09 (range: -0.91 to 1.71), representing a dramatic reduction from raw data mean kurtosis of approximately 85.5, which included extreme values reaching 360.89. Mean skewness reduces to 0.36 (range: -0.05 to 0.98), compared to raw values with mean 2.95 and extreme outliers reaching 14.70.
- **Stationarity:** All preprocessed features maintain stationary properties across the analyzed temporal regimes, correcting the non-stationarity observed in raw microprice data (proportion $= 0.060$). The adaptive EWMA standardization operates on deviations from local regime means and variances, ensuring that standardized features revert to zero mean and unit variance within each local regime.
- **Autocorrelation:** All preprocessed features exhibit universal autocorrelation as an expected consequence of EWMA-based standardization. The autocorrelation structure is controlled and predictable, governed by the selected half-life parameters. This trade-off, accepting controlled autocorrelation in exchange for adaptive scaling, is necessary for stable learning dynamics in sequential neural network architectures.
- **Conditional Heteroskedasticity:** The preprocessed features continue to exhibit heteroskedasticity consistent with EWMA smoothing behavior. In standardized features, heteroskedasticity reflects the adaptive nature of EWMA statistics: as exponentially weighted variance estimates evolve to track changing market conditions, standardized values exhibit time-varying conditional variance. This EWMA-induced heteroskedasticity represents successful tracking of regime shifts.

The reduction in both mean values and standard deviations of window-level statistics confirms that the preprocessing pipeline successfully addresses both point-wise asymmetries and temporal heterogeneity. Features that originally exhibited aggregated mean kurtosis of 13.8 (returns) and 85.5 (non-returns) now show values near zero with tight cross-window consistency. The final preprocessed feature set exhibits symmetric distributions with controlled tails, regime-adaptive stationarity, controlled autocorrelation structure suitable for sequential models, and adaptive heteroskedasticity that tracks market regime shifts.

Limit Order Book Standardization. This preprocessing stage standardizes LOB snapshots into a representation suitable for ML models while preserving the essential information content of the original data.

Liquidity volume exhibits exponential decay from the mid-price, with the vast majority of trading activity concentrated near the best bid and ask. A uniform quantization scheme would inefficiently allocate representation capacity, either over-representing sparse distant regions or under-representing information-rich areas near the mid-price.

A logarithmic quantization scheme provides high resolution close to the mid-price and progressively coarser resolution at distant levels, aligning representational capacity with the actual information content and dynamics of the order book structure.

Using the empirical cumulative distribution of liquidity, which is found to remain relatively stable over time, a threshold Δ can be established that restricts the modeled domain to the most relevant liquidity region. Specifically, the coverage is set to include approximately 70% of total liquidity volume, corresponding to a standardized price range of $\pm 1000\sigma$. This selective representation focuses exclusively on the liquidity most relevant to price formation dynamics.

The preprocessing consists of the following steps applied independently to each LOB snapshot L_t:

- **Price Standardization:** Price levels are transformed to values relative to the mid-price m_t, adjusted by the estimated local volatility $\hat{\sigma}_t$ derived in the previous subsections. The standardized i-th price level is defined as:

$$p_i^{\text{std}} = \frac{p_i - m_t}{\hat{\sigma}_t m_t},$$

  where ask prices yield $p_i^{\text{std}} > 0$ and bid prices yield $p_i^{\text{std}} < 0$.

- **Liquidity Clipping:** The standardized price levels are clipped using a consistent threshold $\Delta = 1000\sigma$ across all snapshots, modeling only a representative fraction of the total liquidity. The index set for standardized prices is redefined as:

$$I(\Delta) = \{\, i \in \{1, \dots, N_b + N_a\} : |p_i^{\text{std}}| \leq \Delta \,\}.$$

- **Logarithmic Quantization:** The standardized price levels within the truncated range $[-\Delta, \Delta]$ are quantized using a logarithmic scheme. Let $2B + 1$ denote the total number of bins and ε a small lower bound that avoids singularity at zero. The edges of the positive-side bins are defined as:

$$e_k = \varepsilon \left(\frac{\Delta}{\varepsilon}\right)^{\frac{k}{B}}, \quad k = 0, 1, \dots, B,$$

  while the negative-side bins are symmetrically defined as $-e_k$.

  The bin assignment function assigns each standardized price to its corresponding bin:

$$B(p_i^{\text{std}}) = k \quad \text{such that} \quad p_i^{\text{std}} \in [e_k, e_{k+1}).$$

  The aggregated liquidity volumes per bin are expressed as:

$$v_k = \sum_{\substack{i \in I(\Delta) \\ B(p_i^{\text{std}}) = k}} v_i,$$

  where $v_i = v_i^b + v_i^a$ denotes the total liquidity at level i.

- **Liquidity Normalization:** Aggregated liquidity volumes within each bin v_k are normalized so that their total sum equals one, transforming the aggregated liquidity into a relative measure of liquidity concentration across the quantized price levels. The normalized liquidity volume in bin k is given by:

$$v_k^{\text{norm}} = \frac{v_k}{\sum_{j=-B}^{B} v_j}.$$

The normalized liquidity vector for each time step t is defined as:

$$\mathbf{v}_t = \left[v_{-B}^{\text{norm}}, \ldots, v_{-1}^{\text{norm}}, v_0^{\text{norm}}, v_1^{\text{norm}}, \ldots, v_B^{\text{norm}}\right]^\top,$$

which represents the standardized and logarithmically quantized distribution of relative liquidity around the mid-price, satisfying the normalization condition:

$$\sum_{k=-B}^{B} v_k^{\text{norm}} = 1.$$

This preprocessing enables temporal and cross-regime comparison of LOB snapshots, producing a standardized representation suitable for ML models.

Training and Validation

This section presents the training, validation, and empirical assessment of the ML components that constitute the HFT strategy. The workflow progresses through three interconnected stages: discrete representation learning, temporal state modeling for decision-making, and synthetic data generation for augmentation.

The first component, based on VQ-VAE architecture, learns discrete representations of LOB snapshots that compress the liquidity distribution into compact latent embeddings, each interpreted an specific market regime. The second component employs a transformer-based PPO agent that processes temporal sequences of market states to generate trading decisions. Additionally, a third component, learns autoregressive dynamics present in discrete representations and BTC returns, in order to generate synthetic LOB data for enhanced strategy training.

For each component, the architectural design, training methodology, hyperparameter optimization, and generalization performance are detailed. The experimental framework employs the aforementioned validation method to ensure statistical robustness and prevent information leakage across temporal partitions.

The section concludes with the Trading Policies validation, where the complete strategy is assessed through simulated trading scenarios under consistent backtesting criteria. This unified framework enables comparison across the three feature sources, manually derived features alone, VQ-VAE latent representations alone, and their combination, measuring how discrete representation learning and synthetic data augmentation contribute to final strategy performance.

Discrete Representation Learning for Limit Order Books

After the feature engineering stage, the dataset contains manually derived features that constitute a relevant source of information for the implemented strategy. However, having state-of-the-art tools specifically designed for information compression and automatic feature extraction, such as autoencoders, the interest arises in applying them to the dataset with the aim of learning representations that encapsulate the liquidity behavior of each snapshot.

In accordance with the literature presented in the theoretical background chapter, specifically, in the ML section, an alternative approach to the aforementioned model is introduced, which considers discrete representation, as opposed to conventional AE of continuous latent nature. This design choice is not arbitrary: the LOB does not exhibit continuous behavior, but rather the distribution of liquidity is highly dispersed across price levels, with significant concentrations at specific levels, often with some symmetry around the mid-price, as well as mismatches that reflect the immediate intentionality of market participants.

Hence, the discrete nature of the problem has direct implications for the model design, both in terms of the representation space and the loss function used during reconstruction. In particular, the loss function based on the MSE, common in image reconstruction tasks, implies assuming Gaussian-distributed errors and is ill-suited to capture differences in liquidity distributions. Since the LOB is encoded as density profiles along standardized price levels, a metric capable of reflecting structural discrepancies between mass allocation is required. For this reason, a loss function based on the 1-Wasserstein distance is chosen:

$$W_1(u, v) = \min_{\gamma \in \Pi(u,v)} \sum_{i,j} \gamma_{ij} |i - j|,$$

where u and v denote the original and reconstructed liquidity distributions across standardized price levels, γ_{ij} denotes the amount of liquidity mass transported from level i to level j, $|i-j|$ corresponds to the cost of moving mass between these levels, and $\Pi(u, v)$ is the set of all valid transport plans (joint distributions with marginals u and v). In fact, this metric quantifies the minimum work required to transform the reconstructed liquidity distribution into the original one, more aligned with the implemented representation of the limit order book snapshots and its interpretation as a distributive object.

Thus, the first ML component of the strategy consists of an automatic feature extractor based on a VQ-VAE approach, explicitly conceived as a representation learning model aimed at LOB snapshots. In this way, the model learns latent embeddings that compactly represent each snapshot of the LOB, allowing the standardized representation to be dispensed with. These embeddings fulfill a double function within the work: on the one hand, they enable the generation of synthetic data by modeling the temporal interactions between representations, and on the other hand, they feed the final strategy with highly informative features.

The implemented architecture follows the canonical encoder-quantizer-decoder structure of VQ-VAE. The decoder employs a softmax activation in the final layer to ensure the reconstruction represents a proper liquidity distribution where each price level has non-negative liquidity and the normalization property holds. The total loss combines reconstruction, commitment, and codebook loss components standard to VQ-VAE architectures. The main component is the W_1 distance, normalized by the maximum possible distance that liquidity can be moved in the density profile. This normalization proves critical for balancing the multi-component loss function: without it, the reconstruction term would completely dominate other components, causing the model to ignore codebook learning mechanisms and collapse to using only a few discrete codes. In addition, a usage penalty is incorporated to encourage complete codebook utilization, avoiding collapse to using only a small subset of embeddings.

Training progress is monitored through perplexity and usage metrics. Perplexity measures the diversity of the distribution of assignments to the codebook embeddings, with higher values indicating more uniform utilization of the discrete representations. Usage reports the fraction of embeddings used at least once in the evaluated batch, providing a direct measure of codebook coverage. The learning rate is adjusted through a scheduler that reduces it upon validation loss plateaus, facilitating convergence to more refined solutions. Early stopping prevents overfitting by selecting the best model before performance degrades on validation data.

The modeling workflow follows a two-phase approach that separates hyperparameter exploration from final production training and validation. The search space encompasses the parameters detailed in the next table.

Hyperparameter	**Search Space**
Codebook size (K)	$\{64, 128, 256, 512\}$
Embedding dimension (D)	$\{64, 128, 256\}$
Convolutional layers	$\{3, 4\}$
Commitment coefficient (β)	$\{0.25, 0.35, 0.5, 0.75\}$
Dropout probability	$\{0.15, 0.2\}$
Usage penalty weight	$\{0.1, 0.15\}$

Table 1: Hyperparameter search space for VQ-VAE optimization.

The optimal configuration contains 2.1 million trainable parameters for a dataset of approximately 200,000 samples.

With regards to the results obtained, the selected model achieves an average total training loss of 0.1330 across the 28 different dataset splits, with values ranging from 0.1189 to 0.1437. The total validation loss, incorporating all loss components, reaches an average of 0.1402 with values ranging from 0.1269 to 0.1547, demonstrating consistent generalization with a modest train-validation gap.

Breaking down the validation performance, the reconstruction loss (normalized W_1 distance) averages 0.0254, with values ranging from 0.0201 to 0.0358, corresponding to approximately 20 to 36 standardized price levels of shift between the original and the reconstructed liquidity distributions. The model demonstrates partial codebook regularity with perplexity ranging from 10.94 to 24.66 (with an average of 14.67) and usage from 17.6% to 36.9% (with an average of 24.4%). Notably, the ratio between perplexity and actual usage reveals that approximately half of the learned embeddings are utilized with regularity: while an average of 31 codes are accessed at least once per batch, the average perplexity of 14.67 indicates that roughly 15 of these are used frequently and uniformly. Although the model does not exploit its full capacity, this configuration represents the optimal balance between diversity and utilization among evaluated configurations.

To assess the generalization capacity of the selected VQ-VAE model, a comprehensive experiment evaluates reconstruction quality on validation data across all 28 splits. The reconstruction metrics provide complementary perspectives on model performance, combining point-wise error measures with distributional and structural assessments.

The model achieves remarkably low point-wise reconstruction error, with an average MSE of 0.000010 (range: 0.000009 to 0.000011) across splits. To contextualize this error magnitude relative to the noise inherent in the data, it is instructive to compare the root mean squared error (RMSE) 0.0032 with the standard deviation of the original features (ranging from 0.00295 to 0.00332 across splits). This comparison reveals that reconstruction errors are of the same order as the natural variability in the data, indicating that the model has effectively learned the underlying process and that its residual errors may correspond primarily to irreducible noise in the observations.

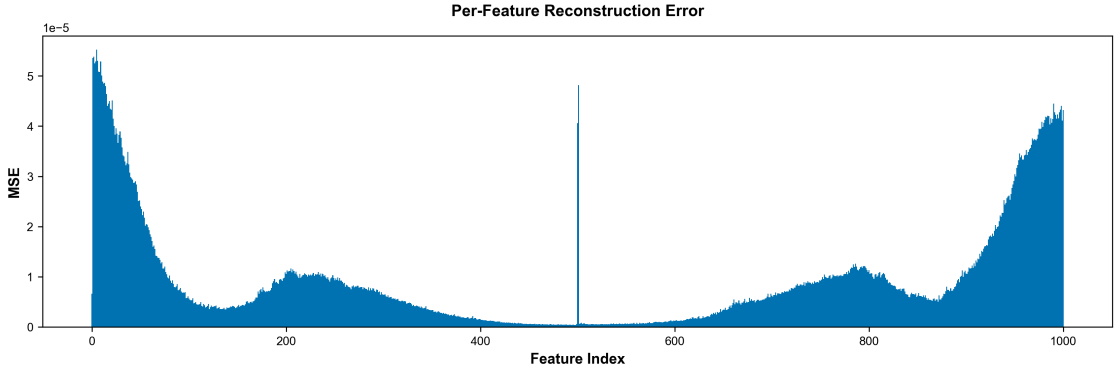


Figure 11: Per-feature reconstruction error for VQ-VAE on validation data (Split 0). MSE across the $B = 1001$ standardized price levels (bins, features).

The per-feature error analysis (Figure 11) reveals a characteristic U-shaped pattern where reconstruction errors are highest in the peripheral price levels and improve toward the central region, with a pronounced peak at the mid-price level (bin 500). This error concentration at the mid-price is expected, as this bin is artificially fixed in each snapshot by construction of the standardized representation, creating a discontinuity that the model must learn to replicate exactly.

Beyond this structural artifact, the logarithmic nature of the price quantization may create the effects that explain the observed error pattern. Near mid-price, bins represent fine-grained standardized price increments where the substantial dynamic liquidity concentrates: because each bin captures a narrow price range, individual bin values remain small despite high aggregate liquidity, resulting in correspondingly small absolute reconstruction errors. Conversely, in peripheral bins, the coarser quantization aggregates liquidity over wider standardized price ranges, producing larger bin values and thus larger absolute errors, though liquidity in these regions is sparse and less decision-relevant for HFT strategies. This error pattern reflects the model successfully adapting to the logarithmic-scale structure of the standardized representation, achieving strong reconstruction fidelity where liquidity dynamics are most active and informationally dense.

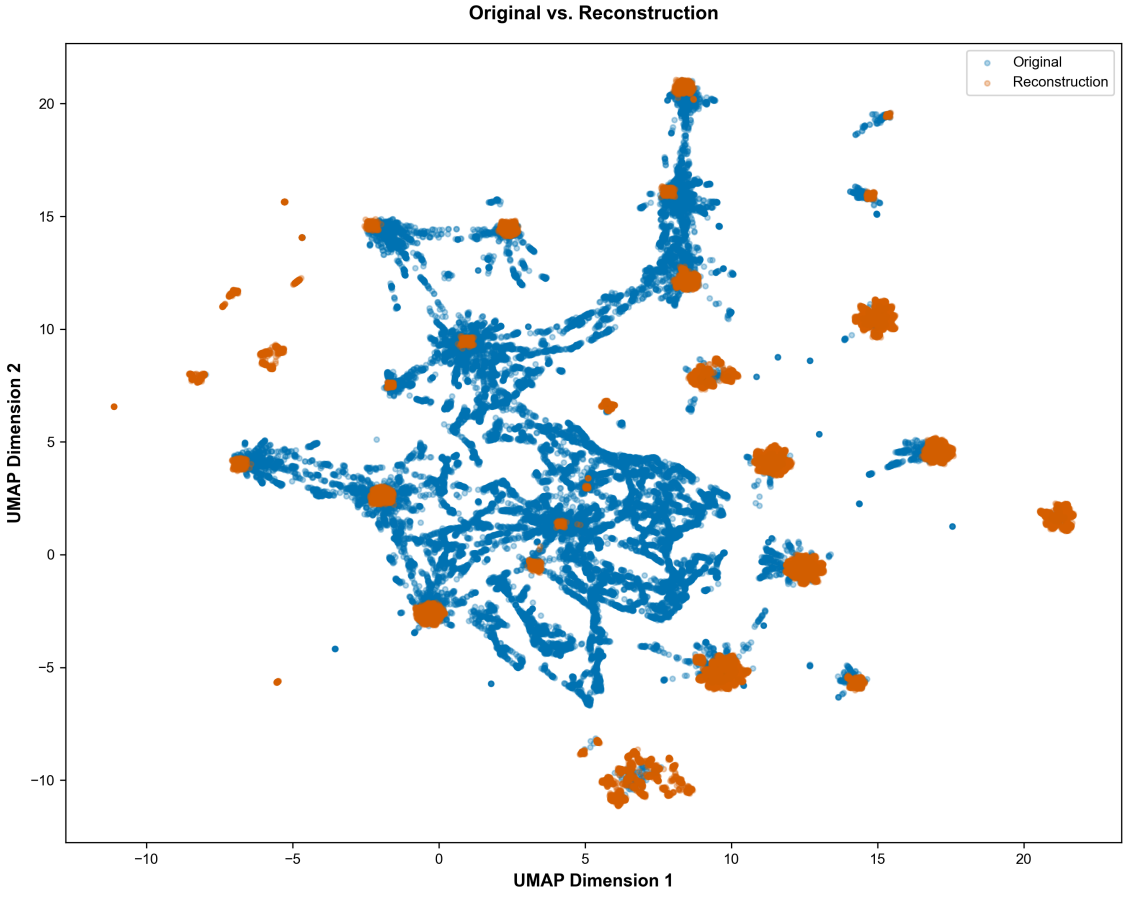


Figure 12: UMAP projection of original validation samples (blue) and VQ-VAE reconstructions (orange) for Split 0.

However, distributional metrics reveal the systematic effects of the discrete bottleneck. The Kolmogorov-Smirnov test uniformly rejects the null hypothesis that original and reconstructed distributions are identical (rejection rate 1.0 across all splits), indicating that the quantization process introduces consistent distributional shifts. This is expected given the discrete nature of the latent space: the VQ-VAE maps the continuous manifold of LOB snapshots onto a finite set of learned codebooks, transforming fine-grained distributional structure into discrete representations.

To properly assess correlation structure in normalized data such as LOB snapshots representation, the centered log-ratio (CLR) transformation is employed. This transformation removes normalized data constraints by mapping the data into a space where correlations meaningfully reflect proportional relationships. Following CLR transformation, the Frobenius correlation between original and reconstructed correlation matrices averages 0.358 (range: 0.321 to 0.397), quantifying the structural transformation from fine-grained to discrete representation. The mean cosine similarity between the original and reconstructed samples averages 0.342 (range: 0.289 to 0.360), indicating that, while the directional alignment is moderately preserved, the proportional structure of the individual LOB snapshots undergoes shape distortion through quantization to the nearest codebook.

Dimensionality reduction via UMAP visualization (Figure 12) confirms this behavior, revealing that reconstructed samples cluster into discrete groups corresponding to codebook centroids, yet these clusters are positioned throughout the fine-grained manifold of original samples. The learned discrete representations span the full range of observed market microstructure conditions, ensuring comprehensive state space coverage. This discretization provides a wide range of different market regime states that enable stable learning in downstream sequential modeling and reinforcement learning applications, while maintaining sufficient fidelity to underlying LOB dynamics for realistic synthetic data generation.

Temporal State Representation Learning for Limit Order Books

After deriving discrete representations of LOB snapshots through the VQ-VAE model, the strategic framework requires a component capable of modeling its temporal dynamics and translating market regime information into actionable trading decisions. While the previous developed component focuses on compressing liquidity distributions at a snapshot basis, this second component addresses the sequential decision-making problem inherent to HFT: determining optimal position sizing and directionality across time given the evolving state of the market. Here, the fundamental question differs from representation learning task: rather than compressing each LOB snapshot into a discrete market regime, the agent learns "what position should I take given the current market conditions and recent history?".

The implemented solution adopts a RL approach, where an agent learns to interact with the market environment, optimizing its behavior according to a reward function that balances profitability, risk, and transaction costs. The policy maps temporal sequences of market states to continuous trading actions, specifically position sizes ranging from full short exposure to full long exposure.

The architecture integrates transformer-based temporal modeling with the PPO algorithm, already introduced in the Theoretical Background chapter, leveraging the representational capacity of self-attention mechanisms to capture long-range dependencies in market microstructure dynamics while maintaining the causal constraint that future information cannot influence present actions.

Three experimental variants are implemented to empirically assess the informative value of different feature sets. The first experiment combines VQ-VAE learned codebooks with manually derived features, testing whether discrete learned representations complement hand-engineered market microstructure information. The second uses only manually derived features, establishing a baseline that relies exclusively on domain knowledge. The third uses only codebooks, evaluating whether compressed LOB representations contain sufficient information for profitable trading without explicit feature engineering. This experimental design enables direct measurement of the contribution that discrete representation learning brings to strategy performance.

The model processes temporal sequences through a modified transformer architecture adapted to handle heterogeneous inputs. A key architectural modification from standard transformer implementations is the fusion mechanism that enables processing of different feature combinations across the three experimental variants. For the dual-input variant, discrete codebooks (integers from 0 to 127 representing different market regimes) are embedded through a lookup table into continuous vectors, while the 18 manually derived features (volatility estimates, depth imbalances, liquidity concentration metrics, order flow features, and price dynamics) are projected through a linear transformation to matching dimensionality. These parallel representations are concatenated and passed through a fusion layer before temporal processing. For single-input variants, the architecture simplifies accordingly, maintaining dimensional consistency through appropriate embedding or projection layers.

Timestamp-sensitive sinusoidal positional encodings are incorporated to provide temporal context adapted to the regular sampling intervals characteristic of high-frequency market data. The temporal processing core consists of a causal transformer encoder, where causal masking ensures that the representation at time t can only attend to states at times $s \leq t$, preventing information leakage from future observations.

After the transformer layers process the temporal sequence, only the final token embedding is extracted as the compressed temporal context. This representation feeds into three specialized output heads that implement the actor-critic architecture required by PPO. The actor head consists of two parallel linear layers: one producing the action mean μ and another producing the logarithm of the standard deviation $\log \sigma$, both outputting scalar values. The log standard deviation is clamped to maintain numerical stability in the policy distribution. The critic head is a single linear layer producing a scalar state-value estimate $V(s)$, representing the expected cumulative discounted reward from the current state.

During training, actions are sampled from a Gaussian distribution parameterized by the actor head outputs using the reparameterization trick, then squashed through the $\tanh$ function to bound them within $[-1, 1]$, with the log-probability calculation including the Jacobian correction term:

$$\log \pi(a|s) = \log \mathcal{N}(a'|\mu, \sigma^2) - \log(1 - \tanh^2(a')).$$

In validation, deterministic mode takes the mean action $a = \tanh(\mu)$, eliminating the policy stochastic component.

The reward function at each timestep is defined as the change in agent portfolio value, incorporating both realized and unrealized profit and loss, adjusted for transaction costs according to the fee structure of the trading scenario:

$$r_t = \Delta \text{PnL}_t - \text{fees}_t,$$

where ΔPnL_t captures both position changes through executed trades and future rewards from the existing positions, and fees_t accounts for transaction costs based on the specific fee regime. This formulation ensures that the agent directly optimizes for net profitability while accounting for the microstructure reality of trading costs.

The PPO algorithm integrates with this architecture through a trajectory buffer and mini-batch update cycle. Each update computes advantages using GAE with discount factor γ and GAE parameter λ. Advantages and returns are normalized across the mini-batch to stabilize policy gradient estimation and prevent value function divergence. The policy loss employs the clipped surrogate objective with clipping parameter ϵ:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right],$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ represents the probability ratio between the new and old policies. This clipping mechanism prevents destructively large policy updates by restricting ratios to a bounded interval. The value loss is the mean squared error between predicted values and normalized returns.

Beyond the standard PPO loss components, three additional regularization terms shape the learning dynamics and encourage economically sensible trading behavior:

- **Entropy Regularization:** An entropy bonus term encourages exploration by penalizing overly deterministic policies. The entropy coefficient employs a linear annealing schedule that decays from a higher initial value to a lower terminal value across training epochs, promoting broad exploration during early training and convergence toward more deterministic policies in later stages.
- **Uncertainty Penalty:** A penalty term proportional to the mean policy standard deviation across the mini-batch prevents the pathological solution where the agent learns that remaining maximally uncertain constitutes a safe strategy. Without this term, the policy could collapse to high-variance distributions that avoid commitment to specific trading decisions.
- **Turnover Penalty:** A penalty term based on the mean absolute change in actions between consecutive timesteps encourages selective trading behavior over excessive position churning. This regularization reflects the economic reality that HFT strategies must balance signal responsiveness with transaction cost minimization.

The total loss function combines these components as:

$$\mathcal{L}_{\text{total}} = -\mathcal{L}^{\text{CLIP}} + \alpha_{\text{value}}\mathcal{L}^{\text{value}} - \alpha_{\text{ent}}\mathcal{H} + \alpha_{\text{unc}}\mathcal{L}^{\text{unc}} + \alpha_{\text{turn}}\mathcal{L}^{\text{turn}},$$

where $\mathcal{H}$ denotes the policy entropy, $\mathcal{L}^{\text{unc}}$ the uncertainty penalty, $\mathcal{L}^{\text{turn}}$ the turnover penalty, and the α coefficients control the relative weighting of each component.

Training employs gradient clipping to prevent training instabilities. The learning rate follows a ReduceLROnPlateau schedule that reduces the rate after a specified number of epochs without any improvement in validation Sharpe ratio, facilitating convergence to refined solutions.

The architectural versatility of the transformer-PPO framework enables systematic exploration of key design parameters to identify configurations that optimally balance model capacity with generalization capability. The hyperparameter search space encompasses the critical architectural and algorithmic parameters detailed in the following table.

Hyperparameter	Search Space
Embedding dimension	{68, 80}
Number of attention heads	{2, 4, 8}
Number of transformer layers	{1, 2}
Sequence length	{20, 50}
Batch size	{256, 512, 2048}
Value loss coefficient (α_{value})	{0.10, 0.15, 0.20}
Entropy coefficient (final)	{0.05, 0.07, 0.10}
Uncertainty penalty (α_{unc})	{0.01, 0.05, 0.10}
Turnover penalty (α_{turn})	{0.005, 0.015, 0.025}

Table 2: Hyperparameter search space for transformer-PPO architecture optimization.

Given the tendency of these architectures to be overparameterized relative to available samples, additional regularization mechanisms are incorporated to ensure generalization capacity, as in every model developed. The modeling workflow follows a two-phase approach: an initial hyperparameter search phase evaluates configurations across the dataset splits to identify optimal settings, followed by production training using the best configuration.

Unlike the VQ-VAE model, where training results and generalization analysis are presented immediately following the architectural description, the assessment of this component's performance is deferred to the Trading Policies section. This organizational decision reflects the integrated nature of the strategy evaluation: the RL agent's effectiveness cannot be meaningfully assessed through isolated loss metrics but must be evaluated through its economic performance in simulated trading scenarios. The Trading Policies section enables comprehensive comparison across all these feature configurations using the consistent evaluation criteria previously described in the Experimental Framework section, which account for profitability, risk-adjusted returns, and trading behavior metrics under multiple fee regimes. This approach ensures that all strategy feature sources, manually derived features, discrete learned representations, and synthetic discrete representations, are evaluated within the same backtesting framework.Therefore, the detailed training dynamics, convergence behavior, validation metrics, and comparative performance analysis are presented in the Trading Policies section.

Synthetic Data Augmentation for Limit Order Books

As reflected in the systemic approach of the HFT strategy, it consists of two complementary ML components, each designed to carry out different functions. The VQ-VAE learns latent representations of liquidity distribution in LOB snapshots, a task in which spatial properties and locality predominate over temporal dynamics. The second component captures the evolutionary dependencies between these representations and their relationship to price formation, motivating their joint development and integration with the RL algorithm.

At this point, a fundamental question arises: is it possible to learn the process of price formation from the information encoded in these discrete representations? More specifically, can a model sample the conditional distribution of latent code sequences and associated price movements to generate realistic synthetic trajectories of the LOB? This capability would expand the dataset for RL training and, in future work, allow the strategy to be evaluated under specific market regimes.

The proposed solution is to explicitly model the a priori distribution on latent representations using an autoregressive framework, with the aim of simultaneously capturing the temporal evolution of discrete states and their relationship with immediate price formation. The data is processed in the form of 120-length sequences, corresponding to one hour of data at 30-second intervals, a time window selected to capture the intra-hourly dynamics of the market while maintaining manageable computational complexity.

The architecture is based on dilated causal convolutions inspired by WaveNet. The dilation pattern generates an exponentially growing receptive field that fully covers the 120-timestep sequence length. Each sequence element consists of a codebook index and its associated immediate log-return. Discrete indices are embedded into 64-dimensional continuous representations matching the VQ-VAE latent space, then concatenated with the scalar log-return value.

The model has two specialized output heads: the codebook head produces logits for discrete class prediction, while the target head produces a scalar for log-return regression.

As a direct consequence of this multi-objective structure, training is defined by a compound loss function that combines two complementary components: a categorical cross-entropy for the prediction of the next codebook state and a Huber loss for log-return regression, with $\delta = 0.01$, suitable for the typical scale of returns (10^{-3}) and more robust to outliers than the MSE. The total loss is expressed as:

$$\mathcal{L}_{\text{total}} = \alpha \mathcal{L}_{\text{codebook}} + (1 - \alpha)\mathcal{L}_{\text{target}},$$

where the weighting parameter $\alpha = 0.5$ assigns equal importance to both components, a fixed choice that prioritizes codebook learning over target regression accuracy.

Regularization is provided through dropout, weight decay, and gradient clipping. Training employs cosine annealing learning rate scheduling and early stopping to prevent overfitting.

The modeling workflow follows a two-phase approach: hyperparameter search across the 28 dataset splits, followed by production training using the best configuration. The search space is detailed right below.

Hyperparameter	**Search Space**
Number of layers (L)	{10, 12, 16}
Number of channels (C)	{64, 80}
Dropout probability	{0.15, 0.2}

Table 3: Hyperparameter search space for Prior model optimization.

The optimal configuration contains 427,681 trainable parameters for a dataset of approximately 200,000 samples.

In terms of the results obtained, the selected configuration achieves an average training loss of 0.6843 across the 28 splits of the dataset, with values ranging from 0.1850 to 0.8692. The validation loss reaches an average of 0.7725 with values between 0.1630 and 1.0193, demonstrating a consistent generalization with a moderate gap between training and validation.

This corresponds to an average perplexity of 2.19, indicating that the model effectively reduces uncertainty by estimating the next codebook from 128 possible latent codes to approximately 2 comparable options. The model assigns an average probability of 46.78% to the next correct latent code, showing that the ground truth codebook is usually among these top candidates.

The model captures on average 84.1% of the temporal information (measured by mutual information), with individual splits ranging from 79.0% to 96.6%, showing a substantial reduction in the inherent uncertainty of the stochastic process in different market regimes.

Breaking down the validation performance mentioned above, the codebook component dominates the learning signal, while target regression achieves an average validation MSE of 1.91×10^{-7}, corresponding to an RMSE of 0.000437. This regression error is substantially less than the natural variability of immediate log-returns, almost half, suggesting that the sequential structure learned in latent space brings additional predictability to the stochastic process of short-term returns.

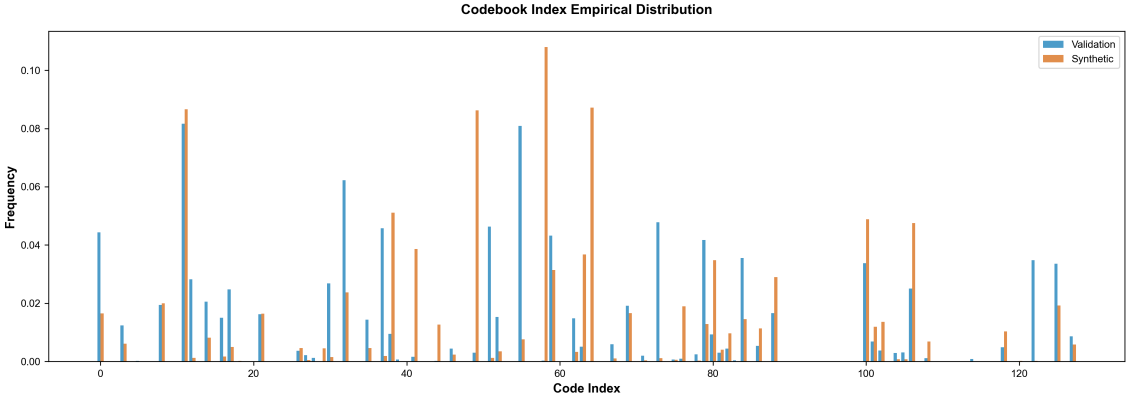


Figure 13: Empirical distribution of codebook indices in validation sequences (blue) versus synthetic sequences (orange) for Split 14.

To assess the generalizability of the selected Prior model, a comprehensive experiment evaluates the quality of the generated synthetic sequences compared to the validation data across all 28 splits. The following metrics provide complementary insights into the performance of the model, examining both the transition dynamics between discrete codebooks and statistically characterizing the trajectories of the LOB, culminating in the evaluation of the properties of the continuous target field.

The model demonstrates a reasonable ability to learn codebook transition patterns, achieving an average JS divergence of 0.202 (range: 0.006 to 0.385) between validation and synthetic distributions. Figure 13 reveals that synthetic sequences show reasonable coverage of discrete latent space, albeit with systematic biases. Some codes appear more frequently in synthetic data while others are underrepresented. This behavior indicates that while the model captures the overall structure of code usage, it tends to favor more frequently visited states over rare transitions, a feature that stems from the limited volume of training data.

Through the calculation of the Frobenius correlation between the transition matrices associated with the synthetic and validation data, it averages 0.428 (range: 0.239 to 0.694), indicating a moderate preservation of the first-order transition dynamics. Figure 14 shows that the validation matrix exhibits characteristic patterns of high self-persistence along the diagonal and structured out-of-diagonal transitions that reflect regime changes in the microstructure of the market.

Similarly, the synthetic matrix captures the dominant diagonal structure, indicating successful learning of codebook persistence, albeit with a less pronounced out-of-diagonal structure. The mean absolute difference in transition probabilities is consistently low, 0.0056 (range: 0.004 to 0.006), with the largest discrepancies in low-probability transitions, while high-probability

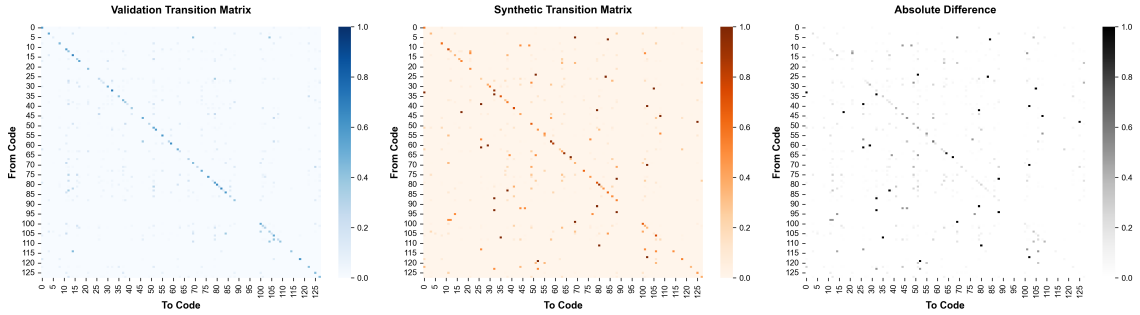


Figure 14: First-order transition matrices for Split 14. Left: validation data. Center: synthetic data. Right: absolute difference.

transitions reproduce accurately, a desirable property that prioritizes fidelity in the most relevant parts of the state space.

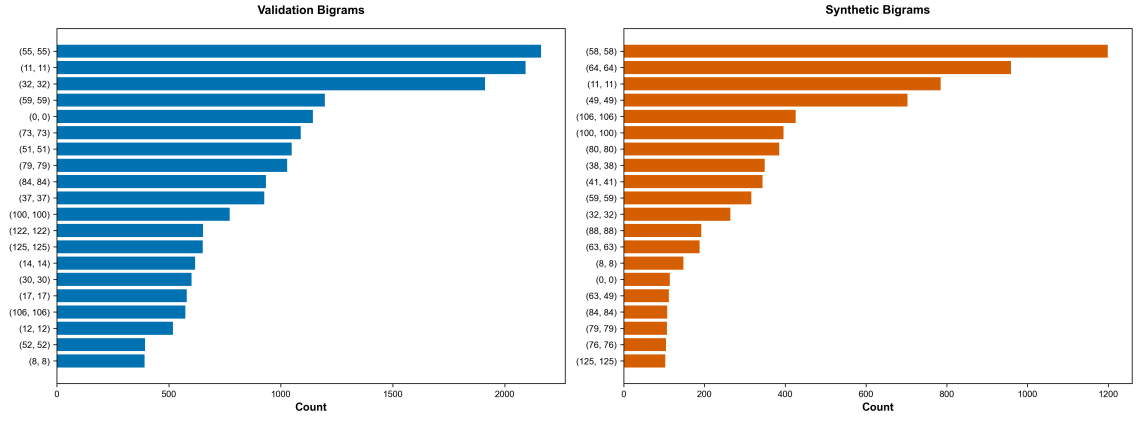


Figure 15: Most frequent bigrams (2-code sequences) in validation sequences (blue) and synthetic sequences (orange) for Split 14.

Analysis of higher-order sequential patterns using n-grams reveals how the performance of the model degrades with increasing time range. The bigram superposition, the proportion of 2-code sequences that appear in both validation and synthetic data, averages 37.68% (range: 10% to 80%), demonstrating that the model reproduces a substantial subset of immediate local transition patterns (Figure 15). The frequency correlation of shared bigrams averages 0.378 between partitions, indicating that, when common patterns are reproduced, their relative frequencies show moderate alignment.

When extending the analysis to trigrams (Figure 16), the degradation of performance becomes apparent. While bigrams capture immediate transitions, trigrams require the model to maintain consistency over three consecutive time steps, exposing limitations in long-range dependency modeling. Overlap ratios and frequency correlations for trigrams vary substantially between partitions, reflecting the fundamental challenge of balancing sequential fidelity with generative diversity.

The evidence presented so far regarding the model's ability to generalize in sequence generation focuses on evaluating the statistical characteristics at the sample level in the form of latent representations. However, it is essential to also evaluate the model's ability to statistically characterize sequence generation as a single process residing in the standardized LOB feature

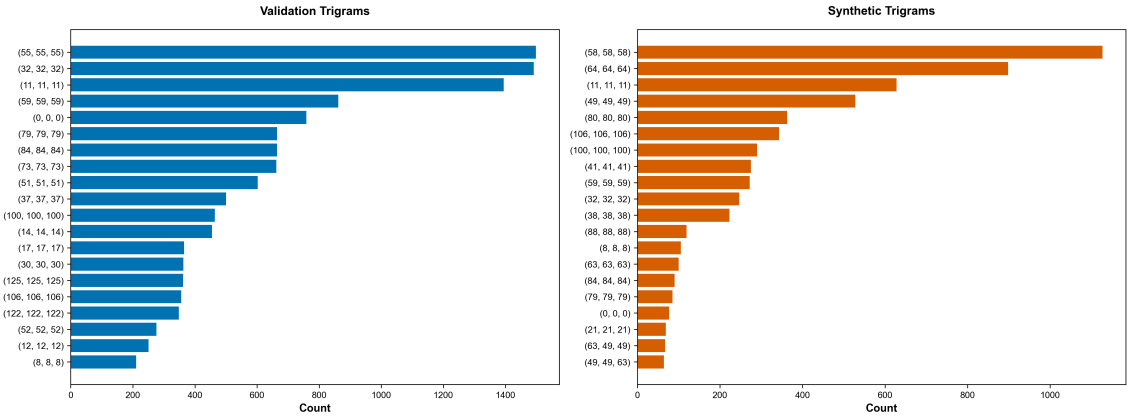


Figure 16: Most frequent trigrams (3-code sequences) in validation data (blue) and synthetic sequences (orange) for Split 14.

space. Thus, the generalization capacity of the Prior model lies in the quality of the complete synthetic LOB snapshots generated by the complete pipeline: sampling of code sequences from the Prior and subsequent decoding using the corresponding component of the VQ-VAE model.

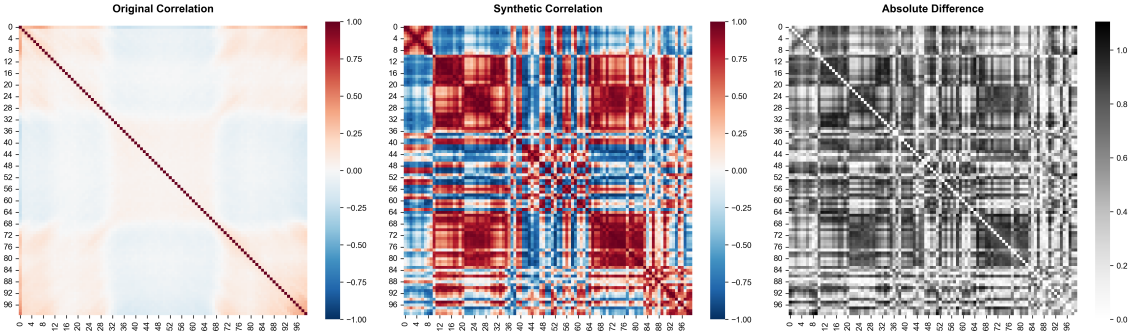


Figure 17: Correlation matrices after CLR transformation for Split 14. Left: validation data showing sparse structure. Center: synthetic data exhibiting elevated off-diagonal correlations. Right: absolute difference revealing widely distributed structural divergence.

Analysis of the correlation structure (Figure 17) reveals significant differences between synthetic and validation data in the standardized LOB feature space. After the CLR transformation to account for the compositional nature of liquidity distributions, the original correlation matrix presents a sparse off-diagonal structure with correlation values predominantly close to zero, reflecting the weak linear dependencies between liquidity at different standardized price levels. In contrast, the synthetic correlation matrix shows substantially high off-diagonal correlations, with prominent block structures indicating spurious dependencies introduced by the generation process. The Frobenius correlation between matrices averages 0.315 (range: 0.102 to 0.565) across splits, quantifying this structural divergence. The absolute differences panel reveals that these discrepancies are widely distributed throughout the correlation structure rather than concentrating on specific pairs of characteristics.

This correlation mismatch indicates that discrete quantization using VQ-VAE codes, despite being quite effective for local compression and reconstruction, introduces systematic dependencies when combined with autoregressive generation. The Prior model learns to sample individually plausible code sequences, but the correspondence of the discrete codebook de-

coder to continuous LOB snapshots creates artificial correlations not present in the validation data. This behavior is characteristic of quantization bottleneck: finite code cannot represent the full set of continuous manifolds of LOB states, and autoregressive sampling tends to favor smooth transitions between similar codes, resulting in locally realistic synthetic snapshots but with a simplified correlation structure compared to the complex and almost independent dynamics of real high-frequency liquidity distributions.

Dimensionality reduction by UMAP (Figure 18) provides a complementary perspective on the overall structure of synthetic versus validation data in the original high-dimensional feature space. The visualization reveals that synthetic samples occupy similar regions of the manifold as the validation data, with substantial spatial overlap between multiple clusters corresponding to different market regimes. However, synthetic samples exhibit narrower clustering patterns within each regimen, reflecting the model's tendency to generate more stereotyped representations of each market regime rather than the full continuous variation observed in the validation data. This clustering behavior is consistent with the high correlations observed in Figure 17: the process of autoregressive generation, restricted by discrete codebooks, produces samples representative of market regimes learned but lacking the fine stochastic variation present in the real dynamics of the LOB.

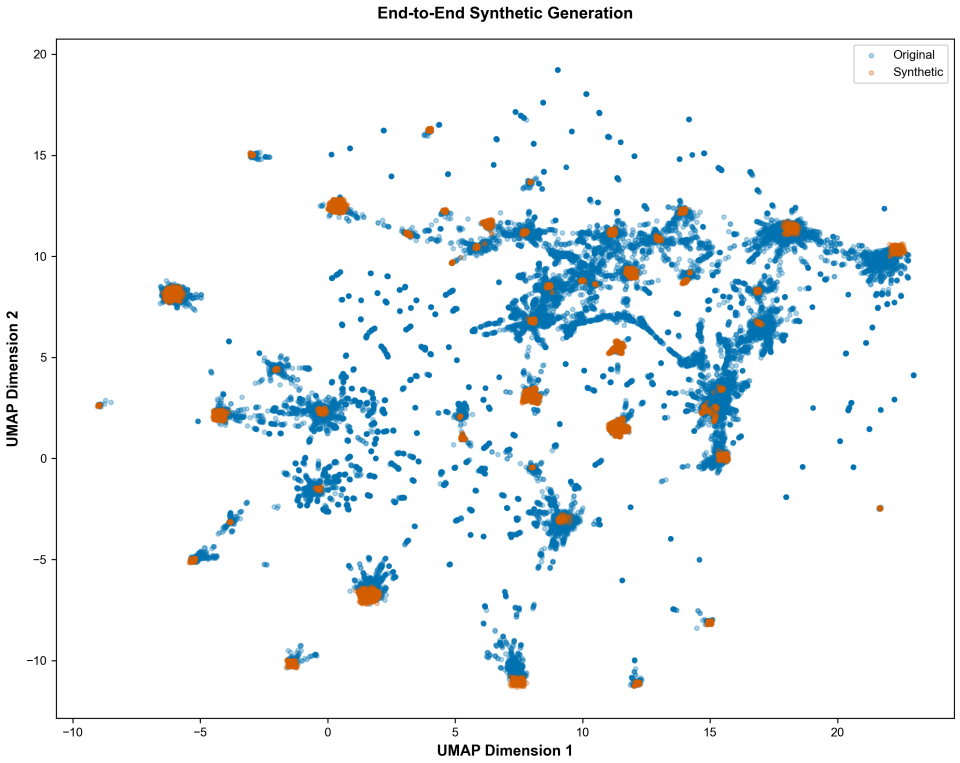


Figure 18: UMAP projection of complete LOB snapshots for validation data (blue) and end-to-end synthetic data (orange) in Split 14.

Returning to the main analysis, now with respect to the regression component of the target, it shows a mixed performance when reproducing the distributional characteristics of the immediate log-returns. The average difference between synthetic and validation targets is 0.000087 (range: 0.0000006 to 0.00076), indicating minimal systematic bias. Despite this fact, the standard deviation ratio averages 1.92 (range: 0.42 to 8.44), revealing that synthetic returns exhibit substantially higher volatility than validation data on most splits.

Despite the distributional mismatch, the model successfully captures stylized facts critical of financial returns. The autocorrelation analysis reveals that both synthetic and validation returns have almost zero autocorrelation in all delays beyond delay 1. The mean absolute error (MAE) of the ACF of 0.089 (range: 0.012 to 0.695) and the correlation of the ACF of 0.487 (range: -0.417 to 0.810) across splits indicate a variable performance in the correspondence of the temporal structure.

As we have seen above, a fundamental stylized fact in financial time series is volatility clustering, the tendency for periods of high volatility to persist. Analysis of the volatility clustering reveals that the validation data show strong persistence, with the autocorrelation of absolute returns remaining substantially above zero even at large delays. Synthetic data reproduce this essential characteristic, albeit with systematically lower magnitudes. The volatility cluster's average correlation of 0.877 (range: 0.495 to 0.986) across partitions indicates a strong preservation of this market microstructure property. The average MAE of 0.168 (range: 0.043 to 0.286) suggests that, although the model underestimates the magnitude of the volatility cluster, it successfully captures the qualitative behavior of volatility persistence.

Finally, the comprehensive evaluation that has been carried out reveals that the Prior model effectively learns time dynamics in discrete latent space and preserves essential stylized facts of price formation, in particular return volatility clustering, appart of the high volatility shortcoming. Additionally, bigram and trigram analysis demonstrates reasonable short-term sequential fidelity with expected degradation as the temporal range increases. However, end-to-end synthetic data have systematic limitations: simplified correlation structure and reduced variation within each regime compared to validation data. These limitations probably stem from the interaction between discrete quantization, fixed task weighting that favors codebook learning over target regression, and the inherent smoothing effect of autoregressive sampling from learned distributions.

Trading Policies

This section evaluates the trading strategies learned by the PPO algorithm across four feature configurations and four transaction cost scenarios. The analysis progresses through three subsections: feature source performance comparing discrete VQ-VAE representations, manually derived features, their combination, and synthetic data; cumulative returns and excess performance examining the temporal evolution of profitability; and return drivers decomposing strategy performance into alpha and beta components to separate skill-based returns from systematic market exposure.

Feature Source Performance. The four evaluated experiments differ in the market state representation provided to the PPO algorithm. Experiment 1 combines discrete representations learned by the VQ-VAE with manually derived features, creating a hybrid continuous-discrete state space. Experiment 2 uses exclusively manually derived features in a purely continuous state space. Experiment 3 employs only VQ-VAE discrete representations, encoding market state through an average of 15 learned regime embeddings, resulting in a purely discrete state space particularly suited for RL algorithms. Experiment 4 uses VQ-VAE synthetic discrete representations generated by the Prior model, providing a controlled environment to assess strategy viability under artificial market dynamics.

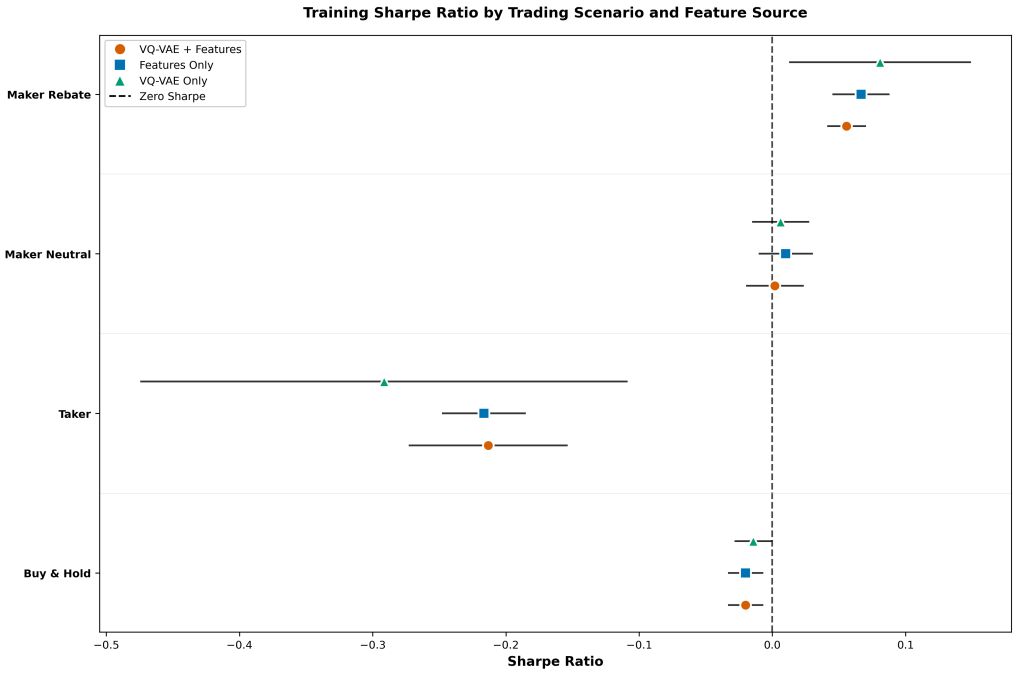


Figure 19: Training Sharpe ratios by trading scenario and feature source across 28 splits for real data experiments. Error bars represent 95% confidence intervals. Experiment 4 (synthetic data) excluded due to scale differences that would compress visualization of real data performance.

Figure 19 presents training Sharpe ratios across the first three experiments and four trading scenarios. The algorithm was optimized exclusively on the taker scenario, which deliberately imposes extreme transaction costs (10 bps + spread) to force discovery of robust trading edges. Training performance shows universally negative taker Sharpe ratios across real data experiments (Exp1: -0.214, Exp2: -0.227, Exp3: -0.292), confirming that even during in-sample optimization, the learned edge cannot overcome the prohibitive fee structure. This result validates the stress-testing design: transaction costs are sufficiently high to prevent

exploitation of spurious patterns.

In contrast, Experiment 4 achieves substantially higher Sharpe ratios (taker: 0.70, maker rebate: 1.02), approximately ten times higher than real data experiments, indicating that synthetic data contains artificial predictability absent in actual market dynamics.

Despite optimizing exclusively on the taker scenario, maker rebate achieves the highest training Sharpe ratios across real data feature sources (Exp1: 0.055, Exp2: 0.070, Exp3: 0.081). The algorithm was never exposed to maker rebate performance during training, yet the learned policy naturally gravitates toward liquidity provision. Optimizing execution timing under extreme transaction costs indirectly teaches the agent when to place limit orders favorably; the same temporal patterns that minimize taker losses maximize maker rebate gains. Maker neutral performance remains close to zero (Exp1: 0.002, Exp2: 0.011, Exp3: 0.006), indicating that without rebate subsidies, the trading edge barely compensates for execution costs.

Figure 20: Validation Sharpe ratios by trading scenario and feature source across 28 splits for real data experiments. Error bars represent 95% confidence intervals. Experiment 4 (synthetic data) excluded from visualization due to scale differences.

Validation results (Figure 20) reveal substantial performance divergence between feature sources and dramatically wider confidence intervals. Taker performance remains universally negative across real data (Exp1: -0.061, Exp2: -0.110, Exp3: -0.057), confirming that the learned edge cannot overcome the 10 bps transaction cost barrier even out-of-sample. This result validates the training design: the extreme fee structure successfully prevented exploitation of spurious patterns while forcing the algorithm toward execution strategies that minimize sudden position rebalancing and maximize temporal precision. Even significantly reducing transaction costs to more realistic exchange levels, the magnitude of the learned edge suggests that aggressive market execution would remain marginally profitable at best, making limit order usage nearly imperative for economic viability. This underscores the need for extensive modeling of fill procedures in backtesting scenarios that approximate realistic market conditions.

Experiment 4 maintains elevated performance in validation (taker: 0.89, maker rebate: 1.07)

with remarkably tight confidence intervals, demonstrating that synthetic data lacks the market efficiency, noise characteristics, and adversarial dynamics present in real LOB data, providing an overly optimistic assessment of strategy viability that would not translate to live trading conditions.

Maker rebate maintains positive validation performance only in VQ-VAE-enhanced experiments using real data. Experiment 1 (VQ-VAE + Features) achieves a Sharpe ratio of 0.017 with substantial uncertainty (-0.034 to $+0.079$), while Experiment 3 (VQ-VAE only) reaches 0.064 (0.039 to 0.080), demonstrating robust out-of-sample profitability. Critically, Experiment 2 (Features only) collapses to negative performance (-0.011 Sharpe, -0.045 to $+0.039$), revealing that continuous manually derived features alone fail to generalize. VQ-VAE discrete representations capture market regime structure that transfers across temporal periods, while manually derived features overfit to training period microstructure.

Validation maker neutral performance diverges notably: Experiment 3 achieves significantly positive Sharpe (0.040, CI: 0.025 to 0.062), Experiment 1 remains close to zero (0.001), and Experiment 2 is negative (-0.031). This separation confirms that successful out-of-sample liquidity provision requires learning when LOB configurations favor passive orders, a pattern that VQ-VAE discrete representations capture more effectively than continuous features.

The progression from training to validation reveals that the optimization framework design functions as intended. The market taker scenario underperforms in both training and validation with consistently negative Sharpe ratios for real data experiments, confirming that extreme transaction costs successfully force learning of robust strategies rather than representing algorithm failure. The learned policy mitigates losses by reducing trading frequency, evidenced by training turnover remaining at moderate values (Exp1: $6.06\% \pm 1.18\%$, Exp2: $6.79\% \pm 0.50\%$, Exp3: $8.31\% \pm 4.92\%$). This turnover level indicates the algorithm learned to be selective in execution, avoiding excessive trading that would amplify the impact of 10 bps transaction costs.

Training activity rates reveal high market engagement across all experiments (Exp1: $94.8\% \pm 1.1\%$, Exp2: $94.4\% \pm 0.9\%$, Exp3: $93.3\% \pm 3.0\%$), indicating the agent maintains continuous market presence through limit orders rather than remaining passive. Policy uncertainty metrics during training (Exp1: 0.628 ± 0.028, Exp2: 0.634 ± 0.035, Exp3: 0.666 ± 0.110) reflect moderate confidence in action selection. Experiment 3 shows greater uncertainty because the discrete representation-based state space means small variations in LOB state can traverse boundaries between the average 15 market regimes, requiring more context-sensitive agent decisions.

The same temporal patterns learned to minimize taker losses manifest as positive performance in the maker rebate scenario. The strategy discovers that liquidity provision, waiting for the market liquidity demands rather than aggressively chasing it, transforms marginal opportunities into profitable trades, enhancing them far when combined with exchange rebate structures. The superiority of Experiment 3 (VQ-VAE only) in validation maker rebate (Sharpe 0.064) compared to Experiment 2 (Features only, Sharpe -0.011) demonstrates that the discrete state space identifies liquidity provision opportunities more effectively than manually derived features, which fail to capture generalizable market microstructure patterns necessary for optimal market timing.

The stark performance gap between real data experiments and Experiment 4's synthetic environment demonstrates that realistic market simulation requires more careful modeling to match the low signal-to-noise ratio and adaptive behavior of actual price formation under .

Cumulative Returns and Excess Performance. This analysis focuses on Experiment 3, as the discrete state space has demonstrated the best generalization in the risk-return analysis. Figures 21 and 22 show training evolution across 1,114 hourly episodes, while Figures 23 and 24 present out-of-sample performance over 367 episodes.

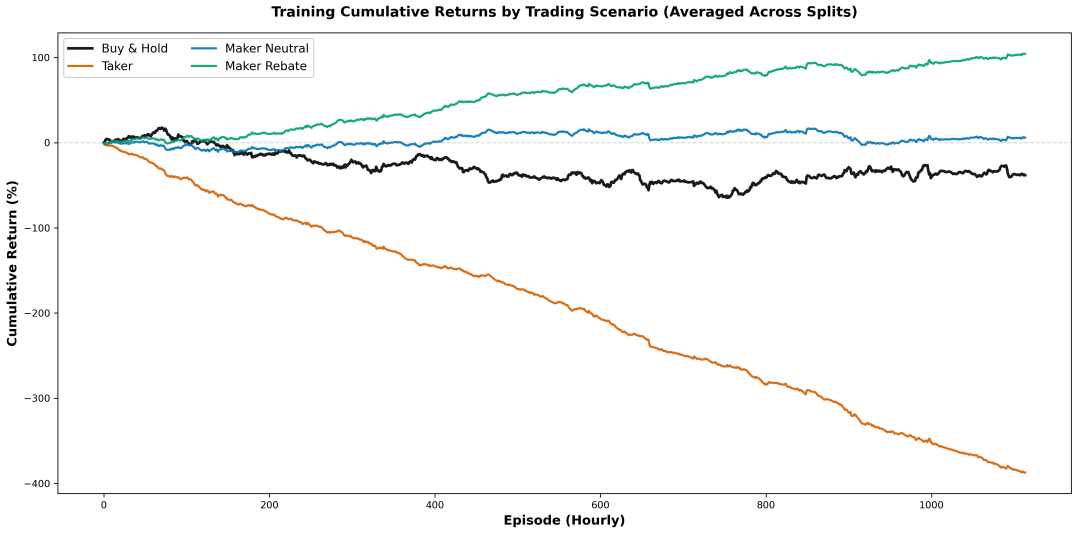


Figure 21: Training cumulative returns by trading scenario for Experiment 3 (VQ-VAE only) averaged across splits.

During training, maker rebate accumulates $+102\%$ return while buy-and-hold experiences -31% loss, resulting in 133 percentage points of outperformance. The taker scenario shows catastrophic collapse with -408% cumulative return, confirming that aggressive execution under extreme transaction costs systematically destroys capital. Maker neutral remains near zero ($+3.6\%$), evidencing that liquidity provision without rebates operates at breakeven.

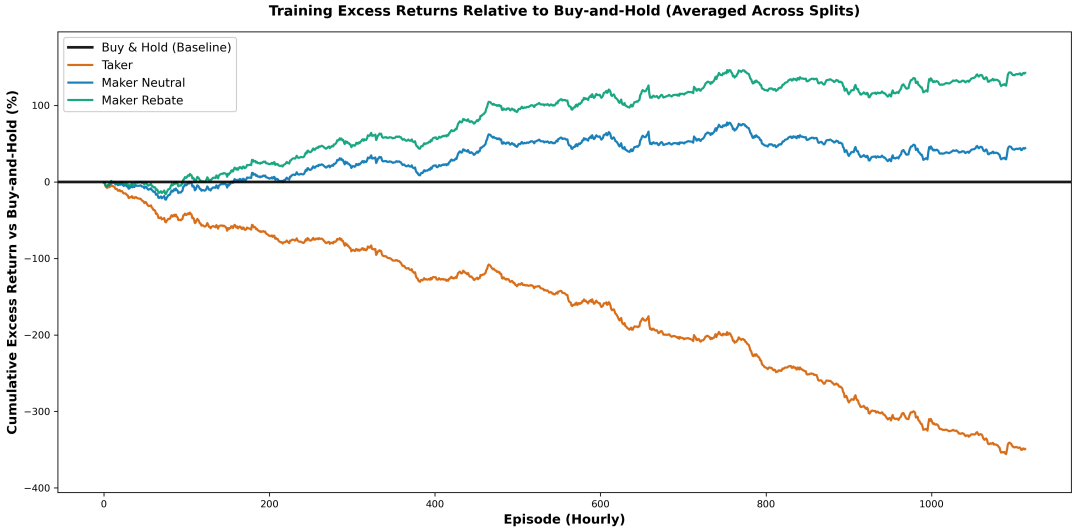


Figure 22: Training excess returns relative to buy-and-hold baseline for Experiment 3. The zero line represents buy-and-hold performance.

The excess returns chart (Figure 22) shows clear separation between maker scenarios and taker, indicating the algorithm rapidly discovers the advantage of liquidity provision.

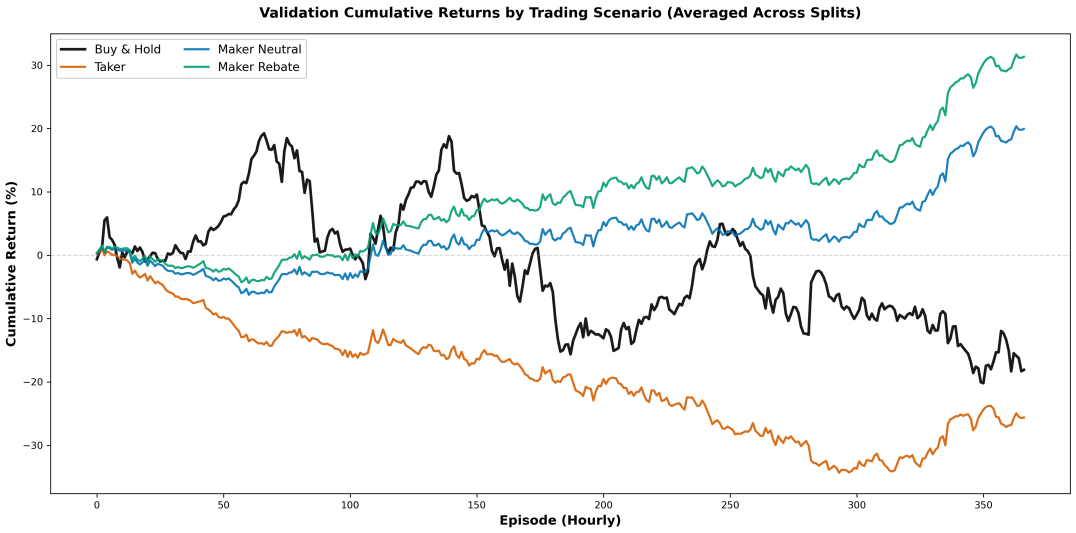


Figure 23: Validation cumulative returns by trading scenario for Experiment 3 averaged across 28 splits over 367 hourly episodes.

In validation (Figure 23), return magnitudes compress but relative relationships persist. Maker rebate achieves $+31.3\%$ while buy-and-hold loses -18.1%, generating 49.4 percentage points of excess return. Maker neutral produces $+19.9\%$ (38.0 percentage points of outperformance), demonstrating that the policy generalizes liquidity provision timing independently of rebates. Taker underperforms with -25.6%, falling below the declining market by 7.5 percentage points.

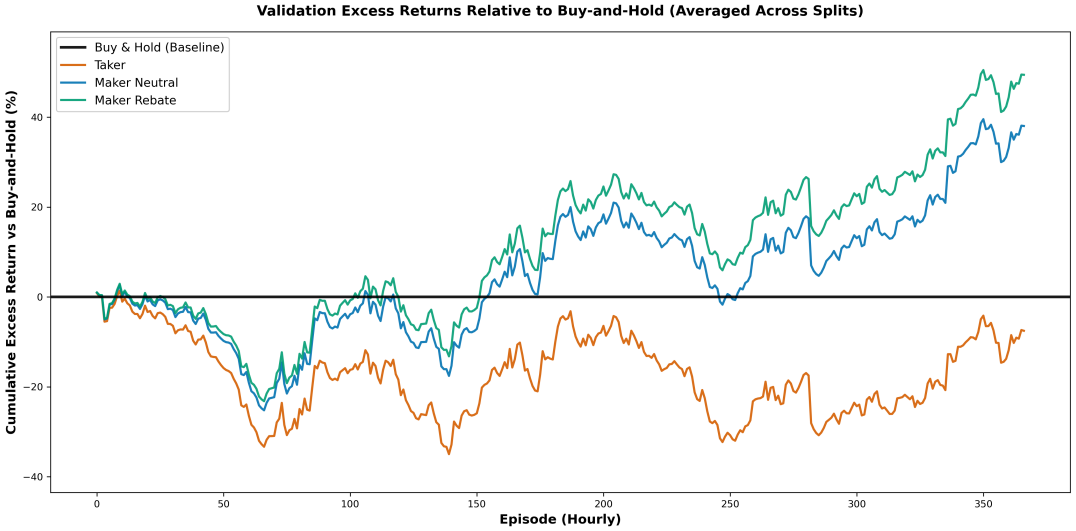


Figure 24: Validation excess returns relative to buy-and-hold baseline for Experiment 3.

Sharpe ratios computed using buy-and-hold as the baseline provide evidence of active strategy value-add. In training, maker rebate achieves 0.0548, maker neutral 0.0129, and taker -0.1552, confirming that aggressive execution destroys value even during in-sample optimization. Validation results maintain the performance hierarchy with compression: maker rebate achieves 0.0685, maker neutral 0.0250, while taker improves to -0.0221 but remains negative, demonstrating that aggressive execution fails to add value after accounting for both market exposure and transaction costs. This positive risk-adjusted performance validates that

the learned policy successfully exploits microstructure patterns rather than riding directional trends.

The substantially higher volatility in validation, reflected in sharper oscillations of cumulative returns, reveals sensitivity to market conditions unseen during training. Episode periods like 50-100 in validation show temporary drawdowns across all scenarios, suggesting that changes in volatility or liquidity regimes challenge the learned policy. Nevertheless, maker strategies recover and continue accumulating excess returns, while taker never recovers from initial losses, confirming that the liquidity provision advantage is more robust to microstructure variations than directional predictions exploited via aggressive execution.

Return Drivers. The previous performance analysis established that maker strategies outperform buy-and-hold substantially in risk-adjusted terms. However, this comparison does not reveal whether profitability originates from genuine predictive skill (alpha generation) or systematic directional market exposure (beta). To decompose return sources, strategy returns are modeled as a linear function of market returns:

$$R_{\text{strategy}} = \alpha + \beta R_{\text{BH}} + \epsilon.$$

This framework separates excess returns into alpha (the intercept, representing returns independent of market direction) and beta (the slope, measuring systematic market exposure). The analysis uses episode returns from Experiment 3, averaged across splits.

In training, maker rebate produces significant positive alpha of $+8.66$ bps per episode, demonstrating substantial value creation during in-sample learning. Maker neutral shows near-zero alpha (-0.16 bps, not statistically significant), confirming that liquidity provision without rebates operates at breakeven. Taker exhibits significant negative alpha of -35.44 bps per episode, indicating that aggressive execution systematically destroys value even during training optimization. The average beta across splits is approximately -0.14, with $R^2 > 0.98$ indicating that the regression successfully explains strategy returns through the combination of constant alpha and market exposure.

In validation, alpha magnitudes compress substantially but maintain statistical significance. Maker rebate achieves $+3.59$ bps per episode ($p < 0.05$), confirming genuine value creation that generalizes to unseen data, explained by rebate arbitrage. Maker neutral shows $+0.49$ bps (not statistically significant), while taker demonstrates significant negative alpha of -11.92 bps per episode, confirming that aggressive execution destroys value through poor timing and excessive transaction costs, arbitrarily imposed. The improvement from -35.44 bps (training) to -11.92 bps (validation) suggests the training environment may have been particularly unfavorable for aggressive execution, though taker remains economically unviable in both cases.

The regression reveals regime-dependent directional positioning rather than consistent market-neutral behavior. The average validation beta is -0.128 with $R^2 > 0.99$, indicating the strategy tactically times both liquidity provision opportunities and directional positioning. Training shows similar beta variability (-0.14 ± 0.51), confirming this adaptive behavior is learned during optimization rather than emerging only in validation.

Final Evaluation

The final evaluation assesses learned trading policies on a held-out test set that remained completely isolated throughout the HFT strategy development process. Unlike cross-validation results averaged across 28 splits, the test set provides a single definitive measurement on entirely unseen market conditions, serving as the ultimate validation of whether discovered patterns reflect genuine microstructural phenomena or artifacts of the training/validation temporal distribution. This evaluation determines the extent to which strategies trained on historical data generalize to future market regimes and whether the performance hierarchy observed during validation persists under completely novel conditions.

Trading Policies

Feature Source Performance. Figure 25 presents test set Sharpe ratios across the three feature configurations and four trading scenarios. The single test period evaluation reveals substantial divergence from validation performance patterns, with feature source rankings reversing between evaluation phases.

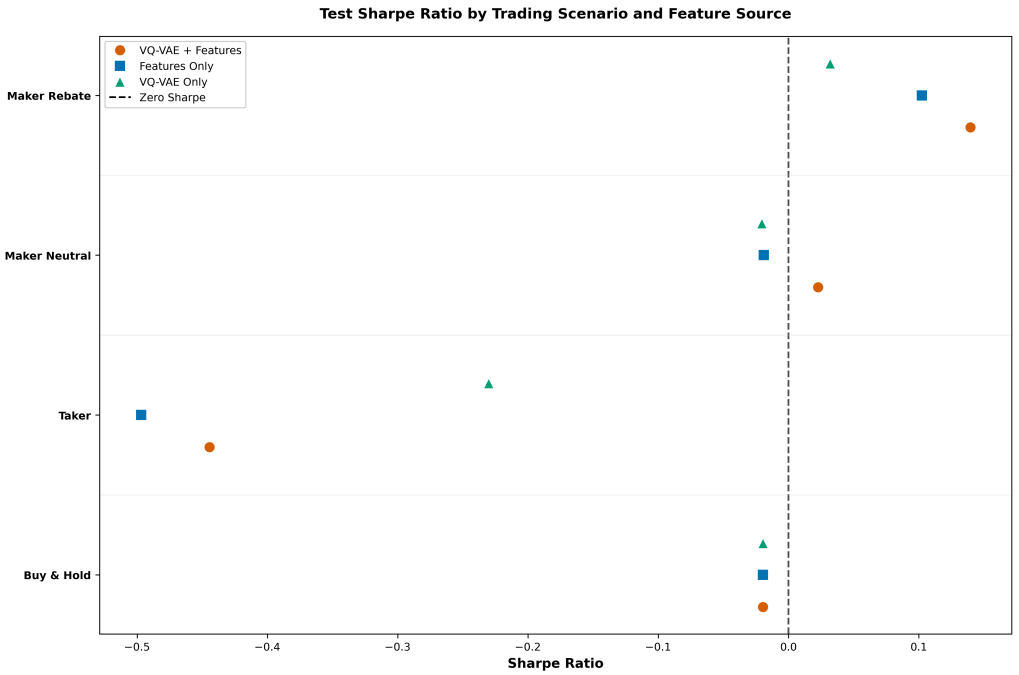


Figure 25: Test set Sharpe ratios by trading scenario and feature source. Points represent single definitive measurements on held-out data never exposed during training or validation.

Experiment 1 (VQ-VAE + manually derived features) achieves the strongest test performance with maker rebate Sharpe of 0.140 and mean return of 14.87 basis points per episode. This substantially exceeds its validation performance (Sharpe 0.017, mean 2.70 bps), indicating the hybrid representation generalizes effectively to this out-of-sample period. Maker neutral achieves slightly positive 2.41 bps, while taker remains deeply negative at -41.21 bps with Sharpe -0.423.

Experiment 2 (manually derived features only) achieves maker rebate Sharpe of 0.102 with mean return of 10.85 bps on the test set, reversing its negative validation performance (Sharpe -0.011, mean -1.41 bps). This dramatic improvement suggests the continuous feature rep-

resentation encounters more favorable microstructure conditions in the test period, though absolute performance remains below the hybrid approach. Maker neutral shows small negative return of -2.02 bps, while taker collapses to -53.45 bps (Sharpe -0.497).

Experiment 3 (VQ-VAE only) demonstrates the weakest test performance with maker rebate Sharpe of 0.032 and mean return of 4.28 bps, marking significant degradation from its superior validation results (Sharpe 0.064, mean 8.45 bps). The discrete state space representation that achieved best cross-validation generalization proves less effective for this test period's specific microstructure characteristics. Maker neutral shows -1.85 bps return, while taker generates -30.81 bps (Sharpe -0.230).

The taker scenario remains universally unprofitable across all experiments with Sharpe ratios ranging from -0.230 to -0.497 and mean returns between -30.81 and -53.45 bps. This consistent failure to overcome the 10 basis point transaction cost across training, validation, and test sets confirms that learned directional edges are insufficient for profitable aggressive execution regardless of feature representation or temporal period. The extreme costs successfully prevented spurious pattern exploitation throughout the entire evaluation framework.

The performance ranking reversal between validation and test sets, where Experiment 3 achieved best validation but worst test performance while Experiment 1 shows the opposite pattern, highlights the fundamental challenge of assessing algorithmic trading strategies with limited out-of-sample data. The single test period cannot provide statistical confidence about which feature representation genuinely generalizes best, as performance differences may reflect the particular microstructure dynamics of this specific temporal window rather than systematic superiority of one approach over another.

Buy-and-hold performance on the test set shows small negative returns across all experiments (approximately -1 to -2 bps), indicating the test period captured slight downward market drift. The maker strategies' positive performance thus represents returns derived from execution timing combined with systematic market exposure, consistent with the beta decomposition analysis presented below.

The divergence between cross-validation and test set results underscores the limited reliability of single-period evaluations for strategy assessment. While Experiment 1's strong test performance is encouraging, the lack of statistical replication across multiple test periods prevents definitive conclusions about relative feature representation effectiveness. The validation framework's 28-split methodology provided robust uncertainty quantification through repeated temporal evaluation, whereas the single test observation offers a point estimate without confidence bounds. This limitation emphasizes that true strategy viability assessment requires either multiple held-out test periods or extended live trading evaluation under operational conditions with realistic fill modeling.

Return Drivers. The test set regression analysis for Experiment 3 reveals fundamental shifts in market exposure compared to validation results. While validation showed regime-dependent beta alternating between -0.555 and $+0.591$ across temporal splits, the test period exhibits uniformly positive beta of approximately 0.56 during training episodes and 0.60 during test episodes. This indicates the learned policy maintains consistent long market exposure throughout the held-out period.

Maker rebate generates statistically significant positive alpha of $+6.89$ bps per episode during training on test data and $+2.06$ bps during test episodes, both smaller than the validation average of $+3.59$ bps but maintaining economic significance when combined with the $+2.5$ basis point rebate subsidy. Maker neutral shows essentially zero alpha at -0.13 and -0.00 bps

respectively, confirming that liquidity provision without rebates operates at breakeven. Taker produces significant negative alpha of -28.20 and -8.24 bps, demonstrating that aggressive execution remains economically unviable despite the test period's more favorable conditions compared to validation.

The regression achieves extraordinarily high R^2 exceeding 0.989 for training and approaching 1.000 for test episodes, surpassing validation's 0.994, indicating the test period exhibits more consistent directional trends with less regime variation. The absence of beta switching suggests either the period's LOB characteristics consistently activate the same VQ-VAE codes inducing long positioning, or the market conditions lack the regime diversity that would trigger the adaptive behavior learned during training. This uniformly positive directional exposure explains the performance divergence across experiments, as different feature representations may differentially capture the specific trend characteristics present in this single test period.

Conclusions

This work addresses the fundamental question of whether microstructural patterns in high-frequency LOB data contain exploitable predictive information for algorithmic trading. The research hypothesis posits that market inefficiencies or risk factor exposures manifest as learnable patterns in LOB dynamics, enabling profitable trading strategies when properly extracted and exploited through ML modeling.

The proposed approach combines a DL task such as representation learning by using a VQ-VAE, with a RL task of learning trading policies optimized for execution under different transaction costs scenarios through a Transformer-based PPO algorithm approach. The methodology emphasizes empirical validation through a Combinatorial Purged Cross-Validation with Embargo setup across 28 splits plus a held-out test set, balancing computational efficiency, ensuring statistical robustness and honest performance assessment in the inherently non-stationary financial environment.

The strategy development pipeline progresses through four stages: dataset construction with temporal partitioning, preprocessing through feature engineering and transformation, model training encompassing representation learning and policy optimization with synthetic data generation as an additional modeling approach, and strategy evaluation across transaction cost scenarios with return source decomposition. Each design decision is guided by observed results rather than theoretical assumptions, with intermediate experiments validating choices before propagating through subsequent pipeline stages.

The following sections synthesize the main findings, examining whether the developed framework successfully identifies exploitable patterns, assessing the economic viability of learned strategies, and evaluating which methodological components contribute meaningfully to final performance versus those that, despite intermediate success metrics, fail to translate into reliable trading behavior.

Main Findings

The empirical analysis confirms the existence of exploitable microstructural patterns in high-frequency LOB data, though profitability depends critically on execution mode and transaction cost structure. Initial experiments revealed that no realistic transaction cost configuration enabled taker strategies to outperform buy-and-hold, indicating marginal predictive edges cannot overcome typical execution costs through aggressive trading. To push learning toward robust patterns, the framework deliberately optimizes under extreme costs (10 basis points plus spread), forcing the algorithm to identify temporal structures justifying execution despite prohibitive fees.

This stress-testing design yields the critical discovery: while the optimized taker scenario systematically destroys value across all configurations even during training, the same policies achieve substantial outperformance in maker scenarios never exposed during optimization. Maker neutral generates 38.0 percentage points excess return over buy-and-hold, while maker rebate reaches 49.4 percentage points, demonstrating that the algorithm learns when to provide liquidity rather than take directional positions, with exchange rebates further amplifying this execution timing edge.

Return decomposition reveals profitability stems from tactical directional positioning combined with execution timing rather than pure market-neutral strategies. Validation beta of -0.128

indicates the strategy adaptively times liquidity provision opportunities and directional exposure rather than maintaining systematic neutrality. Maker rebate's alpha of $+3.59$ basis points per episode primarily reflects arbitraging the 2.5 basis point exchange rebate structure rather than genuine predictive edge.

The most significant architectural finding is the systematic performance gap between feature sources during cross-validation. Experiment 3 (VQ-VAE only) achieves validation Sharpe of 0.064 for maker rebate, while Experiment 2 (manual features only) collapses to -0.011, revealing that discrete regime representations generalize across temporal periods whereas continuous features overfit to training microstructure. The VQ-VAE's approximately 15 learned codes compress LOB snapshots into regime indicators preserving decision-relevant information while discarding noise, enabling transferable pattern identification that continuous feature engineering fails to capture. Hybrid Experiment 1 achieves intermediate performance (0.017), confirming discrete representations drive cross-validation generalization.

However, the held-out test set reveals substantial performance divergence from validation patterns. Experiment 1 achieves strongest test performance (maker rebate Sharpe 0.140), while Experiment 3 demonstrates weakest results (Sharpe 0.032), reversing the cross-validation ranking. This ranking reversal highlights the challenge of assessing feature representation effectiveness across varying temporal regimes.

The cross-validation methodology proves essential for honest performance assessment. Wide validation confidence intervals reflect genuine sensitivity to market regime shifts, with temporary drawdowns during changed volatility or liquidity conditions. Critically, maker strategies recover from drawdowns and continue accumulating excess returns, while taker scenarios never recover, confirming liquidity provision advantages are more resilient to microstructure variations than directional predictions exploited through aggressive execution. The 28-split framework successfully quantifies temporal uncertainty, preventing overconfident conclusions from single-split results.

Component Impact Assessment

The preprocessing pipeline's empirical approach to feature engineering proves essential for enabling stable neural network optimization in manually derived feature experiments. Initial attempts to train the PPO algorithm on raw features with extreme asymmetries and unconstrained scales resulted in catastrophic training instability, with gradient explosions and divergent policy updates preventing convergence.

The stylized facts analysis justified reducing the initial 37 features to 18 through identification of redundancies and temporal characteristics that introduced numerical pathologies incompatible with gradient-based optimization. Excluding historical return horizons beyond h$=$20 snapshots and eliminating intermediate LOB depth levels (K_{15}, K_{all}) due to high cross-correlation with retained levels (K_5, K_{50}) removes noise sources while reducing input dimensionality that exacerbates optimization difficulties.

The dual-stage transformation and standardization procedure addresses distributional challenges that would otherwise prevent PPO convergence. Features with extreme pathologies (kurtosis $>$ 100, skewness $>$ 7) achieve normalized reductions exceeding 0.90, bringing Peterson statistics from over 2000 to below 120. EWMA standardization introduces controlled autocorrelation for regime-adaptive scaling, with final preprocessed features exhibiting mean kurtosis near zero (returns: -0.35, non-returns: 0.09) versus raw data means of 13.8 and 85.5. This preprocessing enables stable PPO training for continuous feature experiments.

The VQ-VAE representation learning component achieves its design objective of compressing LOB snapshots into discrete codes while preserving decision-relevant information. With reconstruction error (RMSE 0.0032) comparable to inherent data noise and approximately 15 codes used regularly from 24.4% codebook utilization, the architecture balances compression against information preservation. The ultimate validation comes from Experiment 3's superior trading performance, demonstrating that learned discrete representations capture microstructural patterns that transfer across temporal regimes and enable profitable policy learning.

In contrast, the Prior model and synthetic data generation approach, despite reasonable intermediate metrics (volatility clustering correlation 0.877, perplexity improvement from 5.03 to 2.19), proves completely unsuitable for strategy training augmentation. Experiment 4 achieves Sharpe ratios approximately 10 times higher than real data experiments, with even taker scenarios showing positive performance (training 0.70, validation 0.89) that never materializes with actual market data. This reveals that synthetic sequences lack the market efficiency, noise characteristics, and adversarial dynamics of real LOB data. The simplified correlation structure (Frobenius 0.315) and stereotyped regime representations produce artificial predictability exploitable by algorithms but bearing no relationship to live trading viability. This establishes that matching stylized facts alone is insufficient, the adversarial nature of price formation and low signal-to-noise ratio in actual markets cannot be captured through autoregressive generation from learned discrete codes with the current approach.

Economic Viability and Practical Implications

The economic viability assessment reveals that limit order execution is nearly imperative for profitable HFT under realistic conditions. Maker scenarios demonstrate robust profitability (validation Sharpe 0.064 for rebate, 0.040 for neutral) with cumulative returns of $+31.3\%$ and $+19.9\%$ respectively against buy-and-hold's -18.1% loss, while taker scenarios consistently underperform (-25.6% return, Sharpe -0.0221), confirming that aggressive execution fails to overcome transaction costs. The learned behavioral characteristics, turnover rates of 6-8% combined with activity rates of 93-95%, indicate selective execution through continuous limit order presence rather than high-frequency churning.

The framework demonstrates that ML-based automatic feature extraction, particularly through discrete representations, achieves superior cross-validation generalization compared to manual feature engineering in RL trading applications. However, the test set reveals this superiority does not guarantee performance on all out-of-sample periods, as Experiment 1's hybrid approach achieves strongest test results (Sharpe 0.140) despite weaker validation performance (0.017). This indicates relative feature representation effectiveness depends critically on specific market regime characteristics of the evaluation period.

However, critical simplifying assumptions limit immediate practical applicability. The guaranteed fill assumption abstracts from adverse selection, queue position dynamics, and partial fills that dominate real LOB mechanics, likely overstating maker neutral's $+38.0\%$ excess return. The single-instrument focus on BTC/USD pair on BitMEX limits generalizability, as learned patterns may exploit cryptocurrency market idiosyncrasies or exchange-specific microstructure rather than universal mechanisms.

Methodological Contributions and Limitations

The experimental framework's primary contribution lies in demonstrating that rigorous empirical validation can guide design decisions through complex ML pipelines for financial appli-

cations. The adopted validation methodology with combinatorial splits, purge and embargo zones, and systematic validation propagation prevents optimizing intermediate metrics that fail to improve end-task performance. Each design choice undergoes validation across multiple temporal regimes, revealing which decisions generalize versus those exploiting training artifacts. The component-wise experimental approach enables performance attribution to specific architectural choices during cross-validation.

The dual evaluation framework combining cross-validation for design decisions with a held-out test set establishes appropriate boundaries between model development and performance claims. Cross-validation results represent honest generalization estimates across multiple temporal regimes, while the test set provides definitive assessment on completely unseen conditions. The substantial performance divergence between validation and test, particularly the ranking reversal where Experiment 3 dominated validation but Experiment 1 dominated test, underscores that single-period evaluations cannot resolve which approach genuinely generalizes best, as different temporal regimes activate different aspects of learned policies.

However, methodological limitations constrain conclusions. The data-driven approach requires substantial computational resources (28 splits $\times$ multiple hyperparameter configurations $\times$ iterative stages $=$ hundreds of runs), potentially prohibiting practitioners from conducting exhaustive validation. Single-instrument focus and 30-second sampling frequency limit validation across market microstructures and higher-frequency dynamics. The VQ-VAE achieves partial codebook utilization (24.4%, approximately 15 codes used regularly), suggesting either that LOB dynamics compress into few distinct regimes or that the architecture fails to discover the full microstructural pattern range. The Prior model's difficulty with long-range dependencies indicates temporal dynamics in discrete latent space may be more complex than captured by the autoregressive architecture, though this does not impair VQ-VAE utility for direct policy learning. Most critically, the single test period cannot provide statistical confidence about relative feature representation effectiveness, as observed performance differences may reflect this specific temporal window's characteristics rather than systematic superiority.

Hypothesis Evaluation

The research hypothesis stated that BTC-USD limit order book data contains microstructural patterns exploitable by deep learning and reinforcement learning models to develop systematic and profitable trading strategies. Based on empirical evidence, the null hypothesis is rejected: both cross-validation and held-out test evaluation demonstrate consistent maker strategy outperformance over passive baselines.

However, this rejection requires critical qualifications. Profitability materializes exclusively through maker execution rather than aggressive taker scenarios. Returns combine tactical directional positioning with execution timing rather than pure anomaly exploitation, with maker rebate's alpha primarily reflecting exchange rebate arbitrage. The substantial validation-test performance divergence demonstrates that pattern exploitability varies significantly across temporal regimes. Additionally, the idealized evaluation environment (guaranteed fills, absence of latency modeling) leaves practical viability under operational constraints unvalidated.

In summary, the hypothesis is conditionally accepted: microstructural patterns exist and enable strategies outperforming passive baselines, but exploitation requires maker execution, combines timing with tactical positioning, varies across temporal regimes, and faces uncertain practical viability.

Future Research

The most critical extension involves incorporating realistic limit order fill modeling to assess economic viability under operational conditions. Future work must model queue position dynamics, partial fills, adverse selection from informed traders, and the stochastic nature of limit order execution. This extension would likely reduce maker scenario profitability substantially, providing honest assessment of whether the learned liquidity provision timing truly generates positive returns after accounting for the times when limit orders fail to execute at favorable prices or execute primarily during adverse price movements.

The Prior model's synthetic data generation, though unsuitable for training augmentation, could enable regime-specific stress testing once improved to match real market characteristics more faithfully. Rather than augmenting training data, synthetic sequences could generate adversarial scenarios or rare market conditions for strategy evaluation, assessing robustness to regime transitions or stress events underrepresented in historical data. This application requires resolving the correlation structure simplification and artificial predictability issues identified in Experiment 4.

Higher-frequency evaluation at sub-second granularity with realistic latency modeling would test whether learned patterns persist at the timescales where actual HFT operates. The current 30-second snapshots may aggregate away microstructural dynamics that dominate profitability at millisecond scales, or conversely, the patterns discovered at 30-second resolution may derive from slower regime transitions that remain exploitable at higher frequencies. Incorporating realistic network latency, exchange matching engine delays, and order placement timing constraints would reveal whether execution timing advantages persist under operational conditions.

A. Machine Learning Background

A.1 Deep Learning

This appendix provides foundational deep learning concepts and mathematical derivations that support the architectures employed in the main work.

A.1.1 Feedforward Neural Networks

Feedforward Neural Networks (FFNs), also known as Multilayer Perceptrons (MLPs) or Fully Connected Neural Networks (FCNNs), among other nomenclatures, are the most fundamental components of DL architectures. In such artifacts, information flows in a single direction from the input layer, through hidden ones to the output layer, with no recurrent connections or cycles.

Formally, given a dataset of pairs $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, each containing the features that will be mapped to the associated ground truth label, a FNN with L layers processes these features $\mathbf{x}$, of arbitrary dimensionality, through a composition of affine transformations followed by nonlinear activations. A single one of these operations is what is widely known as a neuron in ML terms. Considering the backpropagation algorithm, that is, the method used to tune the layer weights, for each layer $l = 1, \dots, L$, the forward propagation step, is recursively defined as:

$$\mathbf{h}^{(l)} = f(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}),$$

where $\mathbf{h}^{(l)} \in \mathbb{R}^{d_l}$ represents the l-th layer activations, $\mathbf{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ and $\mathbf{b}^{(l)} \in \mathbb{R}^{d_l}$ are the weights and biases of the l-th layer, f is a nonlinear activation function such as ReLU, tanh, or sigmoid, and $\mathbf{h}^{(0)} = \mathbf{x}$ is the input data. The final output of the network is $\hat{\mathbf{y}} = \mathbf{h}^{(L)}$.

Training a FNN consists of finding the optimal parameters $\boldsymbol{\theta} = \{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^{L}$ that minimize a loss function on the training dataset. This is done by applying the backward propagation step of the aforementioned backpropagation algorithm, computing loss function gradients and updating the parameters. For regression tasks, the loss function is typically the mean squared error (MSE):

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}\Big[\, \|\mathbf{y}_i - f_{\boldsymbol{\theta}}(\mathbf{x}_i)\|_2^2 \,\Big] = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|_2^2 \,,$$

where N is the number of training samples, $\mathbf{y}_i$ are the ground truth values, and $\hat{\mathbf{y}}_i$ are the network outputs. For classification tasks, the cross-entropy loss function is typically used:

$$\mathcal{L}(\boldsymbol{\theta}) = -\mathbb{E}\Big[\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c})\Big] = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c}),$$

where $y_{i,c}$ indicates whether class c is correct for sample i, and $\hat{y}_{i,c}$ is the predicted probability obtained through a softmax activation at the output layer.

Parameter optimization is performed by stochastic gradient descent or its variants (Adam, RMSprop), applying the backpropagation algorithm to efficiently calculate the gradients $\nabla_{\boldsymbol{\theta}}\mathcal{L}$.

Although FNNs are universal approximators of functions and can learn complex correspondences with sufficient data and adequate depth, their fully connected nature implies that they do not exploit structural regularities in the input data, requiring a large number of parameters and substantial training data. This limitation motivates more specialized architectures focused on modeling data with spatial or temporal structure.

A.1.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are specific neural network architectures designed to process grid-like data, such as images or time series, by exploiting the spatial or temporal structure respectively found in them. Unlike FNNs, CNNs are fundamentally built on three key architectural principles: local receptive fields, shared weights across spatial locations, and hierarchical feature extraction through progressive pooling operations.

Formally, given an input tensor $\mathbf{x} \in \mathbb{R}^{H \times W \times C_{in}}$ with height H, width W, and C_{in} input channels, a convolutional layer applies a set of learnable filters (or kernels) $\mathbf{W}^{(l)} \in \mathbb{R}^{k \times k \times C_{in} \times C_{out}}$ of spatial extent $k \times k$ to produce a set of feature maps $\mathbf{h}^{(l)} \in \mathbb{R}^{H' \times W' \times C_{out}}$. The convolution operation at spatial position (i, j) and output channel c is defined as:

$$\mathbf{h}^{(l)}_{i,j,c} = f\left(\sum_{m=0}^{k-1}\sum_{n=0}^{k-1}\sum_{c'=0}^{C_{in}-1} \mathbf{W}^{(l)}_{m,n,c',c} \cdot \mathbf{h}^{(l-1)}_{i+m,j+n,c'} + \mathbf{b}^{(l)}_{c}\right),$$

where f is a nonlinear activation function analogous to those used in FNN architectures, $\mathbf{b}^{(l)}_{c}$ is the bias for output channel c, and the output spatial dimensions H' and W' depend on the padding and stride configuration, which control how the filter is applied to the input.

CNN architectures typically alternate convolutional layers with pooling operations that progressively reduce spatial dimensions while increasing the number of feature channels, building hierarchical representations from low-level features (such as edges and textures) to high-level semantic concepts. Common pooling operations include max pooling, which selects the maximum activation within a local spatial window, and average pooling, which computes the mean. The final feature maps are typically flattened and fed to FNN layers for classification or regression tasks.

Training CNNs follows the same backpropagation framework as FNNs, with gradients computed efficiently through the convolution operations. The convolutional structure introduces inductive biases that are particularly well-suited for data where nearby elements are more strongly correlated than distant ones, making CNNs the dominant architecture for computer vision tasks and increasingly common for sequential data processing. However, the inherent locality of convolutions requires information to propagate through multiple layers to capture long-range dependencies, motivating architectures that enable direct interaction between distant sequence positions.

A.1.3 Autoencoders

The Autoencoder (AE) is DL model that aims to learn a latent (lower-dimensional) representation of data $\mathbf{x} \in \mathbb{R}^N$, with the goal of reconstructing it as accurately as possible from this simplified representation $\mathbf{z} \in \mathbb{R}^M$ (with $M \ll N$), yielding a reconstruction $\hat{\mathbf{x}}$.

The architecture that implements this type of model consists mainly of two parts:

- **Encoder:** The encoder $E_\phi : \mathbb{R}^N \to \mathbb{R}^M$ is optimized to extract informative features, producing the latent representation $\mathbf{z} = E_\phi(\mathbf{x})$, which captures the most relevant structure of the data necessary for reconstruction.
- **Decoder:** The decoder $D_\theta : \mathbb{R}^M \to \mathbb{R}^N$ is optimized to be able to reconstruct the original data as faithfully as possible from its latent representation, yielding $\hat{\mathbf{x}} = D_\theta(\mathbf{z})$.

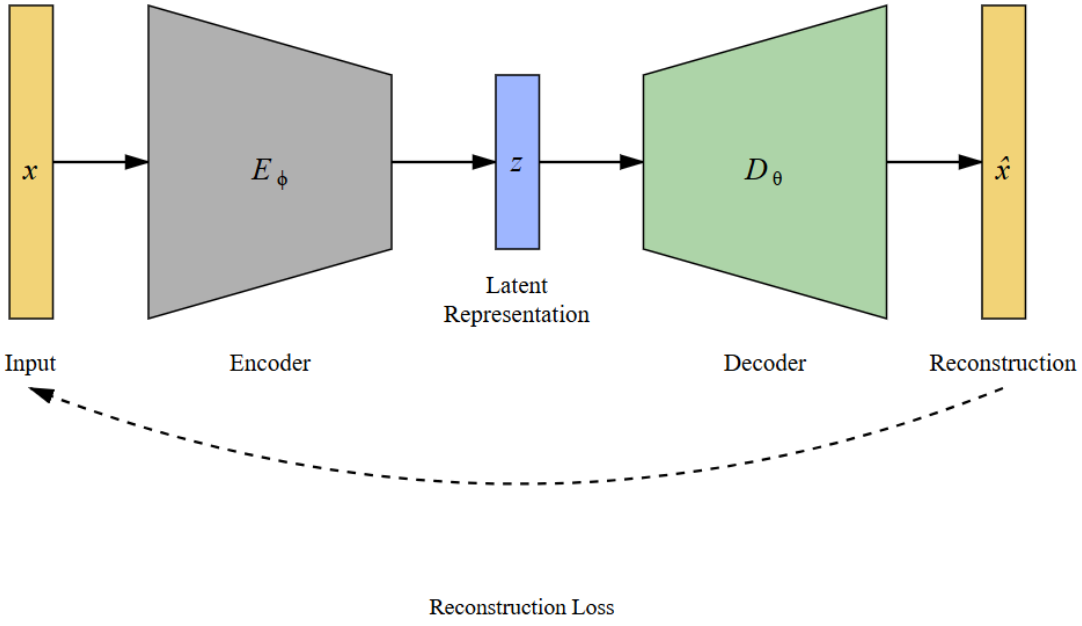


Figure 26: Autoencoder architecture showing the encoder-decoder structure with latent representation z.

The autoencoder learns its parameters by minimizing the reconstruction loss between the input data $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$ over the entire dataset:

$$\mathcal{L}(\mathbf{x}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 = \|\mathbf{x} - D_{\boldsymbol{\theta}}(E_{\boldsymbol{\phi}}(\mathbf{x}))\|_2^2 .$$

Thus, through this model, it is possible to extract informative characteristics that disentangle redundancy from the essential information required to reconstruct the input with minimal loss.

A.1.4 Variational Autoencoders

Building upon the deterministic representation-learning framework introduced by Bengio, Courville, and Vincent (2013) [4], the Variational Autoencoder (VAE) extends the AE architecture by incorporating a probabilistic perspective.

If one conceptually starts from the previous model with the intention of modeling the underlying data generation process, it must be noted that the AE, by itself, is merely a model for representation and reconstruction, possibly involving some loss of information.

To learn the underlying generative process of the data and to enable generation of new one, the model must be endowed with a probabilistic interpretation. This objective is achieved by applying the principles of variational inference (VI) to the AE framework, an approach rooted in variational Bayesian (VB) methods first formalized by Jordan, Ghahramani, Jaakkola, and Saul [26], and later extended to DL models by Kingma and Welling [28].

To this end, from now on it is considered that the data to be modeled, or observed data $\mathbf{x}$, is the result of a generation mechanism governed by a set of unknown or unobserved variables, which are precisely latent variables $\mathbf{z}$. Thus, according to Bayes' theorem, it is possible to express mathematically the probabilistic relationship between the observed data and the latent variables as:

$$P(\mathbf{z}|\mathbf{x}) = \frac{P(\mathbf{x}|\mathbf{z})P(\mathbf{z})}{P(\mathbf{x})},$$

where $P(\mathbf{z}|\mathbf{x})$ is the posterior distribution, describing how well the observed data explain the latent variables; $P(\mathbf{x}|\mathbf{z})$ represents the likelihood, indicating how the latent variables explain the observed data; $P(\mathbf{x})$ is the marginal likelihood, that is, the probability of observing the data; and $P(\mathbf{z})$ is the prior distribution, defining how latent variables are expected to behave.

Through the marginal likelihood $P(\mathbf{x})$, it is possible to measure how well the entire model, comprising the prior distribution $P(\mathbf{z})$ and the likelihood $P(\mathbf{x}|\mathbf{z})$, adequately explain the observed data. In the field of probabilistic modeling, the maximum likelihood framework seeks to find the parameters that maximize this marginal likelihood $P(\mathbf{x})$, ensuring that the model fits data appropriately.

Formally, the marginal likelihood is defined as the probability of observing the data integrated over all possible latent variables:

$$P(\mathbf{x}) = \int P(\mathbf{x}|\mathbf{z})P(\mathbf{z})\, d\mathbf{z}.$$

However, due to the components involved in the estimation of the marginal likelihood, this integral is typically computationally intractable, and the likelihood to be parameterized may also be highly complex.

For these reasons, a simpler parameterized distribution $q_\theta(\mathbf{z}|\mathbf{x})$ is introduced to approximate the true posterior distribution, together with a deterministic function $f_\theta(\mathbf{z})$ that relates latent variables to observed data by parameterizing the likelihood.

$$P(\mathbf{x}) = \int q_\phi(\mathbf{z} \mid \mathbf{x}) \frac{P(\mathbf{x} \mid \mathbf{z})\, P(\mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})}\, d\mathbf{z} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\frac{P(\mathbf{x} \mid \mathbf{z})\, P(\mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})} \right].$$

Finally, assuming that the latent variables follow a normal prior $P(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ (thereby imposing a known a priori distribution), it is possible to express the distributions of each component of the model architecture and the metric to be optimized.

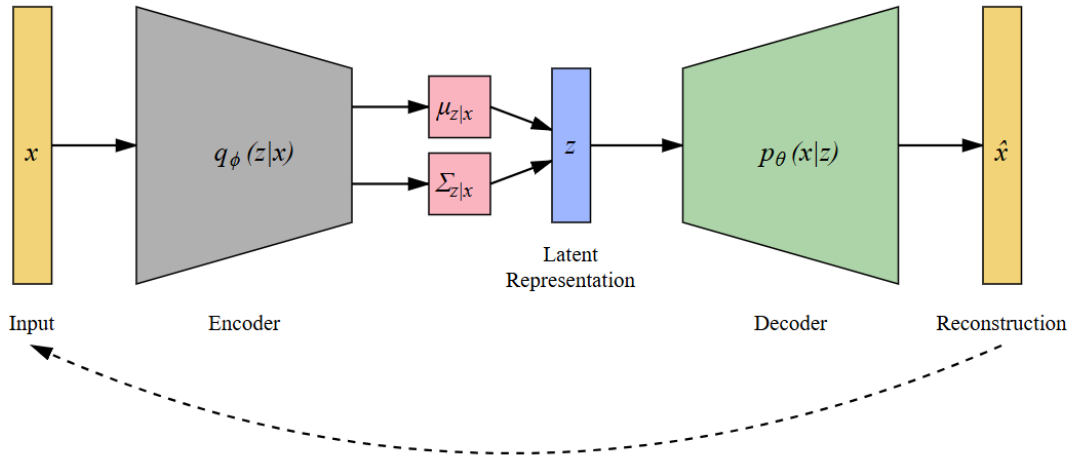
- **Encoder:** The encoder $E_\phi(\mathbf{x})$, conditioned on the observed data, produces the parameters that define the latent-space distribution, typically modeled as

$$E_\phi(\mathbf{x}) = q_\phi(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\mathbf{z};\, \boldsymbol{\mu}_\phi(\mathbf{x}), \mathrm{diag}(\boldsymbol{\sigma}^2_\phi(\mathbf{x}))).$$

  This allows assigning to each input $\mathbf{x}$ a latent representation that is likely to have generated the observation.

- **Decoder:** The decoder $D_\theta(\mathbf{z})$, in turn, defines a likelihood whose mean is given by a deterministic function $f_\theta(\mathbf{z})$; a common choice is

$$D_\theta(\mathbf{z}) = p_\theta(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mathbf{x};\, f_\theta(\mathbf{z}), \sigma^2 \mathbf{I}).$$



ELBO and KL Divergence

Figure 27: Variational Autoencoder architecture. The encoder $q_\phi(z|x)$ outputs distribution parameters $\mu_{z|x}$ and $\Sigma_{z|x}$, from which latent variable z is sampled. The decoder $p_\theta(x|z)$ reconstructs the input. The model is optimized using the ELBO objective, which balances reconstruction accuracy with KL divergence regularization.

In contrast to the deterministic loss used in the AE model, the VAE learns its parameters by maximizing the marginal likelihood of the observed data. This loss, known as the Evidence Lower Bound (ELBO), balances two competing terms: the reconstruction loss and the regularization loss imposed on the latent distribution. Formally, the VAE maximizes the following loss over the entire dataset:

$$\mathcal{L}(\mathbf{x}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x} \mid \mathbf{z})\big] - D_{\mathrm{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z}))\,.$$

where the first term represents the expected log-likelihood (measuring how well the model reconstructs the data), and the second term is the Kullback–Leibler divergence, which regularizes the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ to remain close to the prior $p(\mathbf{z})$.

To implement the objective and make the model trainable, while also enabling the generation of unseen data, it is necessary that sampling from the latent distribution be differentiable to

allow the application of the backpropagation algorithm. This is achieved through the reparameterization trick, which simply consists of sampling a noise term from a normal distribution and using it to generate the latent sample in a differentiable way:

$$\mathbf{z} = \boldsymbol{\mu}_{\phi}(\mathbf{x}) + \boldsymbol{\sigma}_{\phi}(\mathbf{x}) \odot \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

Consequently, once the model has learned the underlying distribution of the data-generating process, it can be used as a generative model: by sampling ε from the prior and feeding the decoder, new data can be generated that share the same statistical characteristics as the observed one.

A.2 Reinforcement Learning

This appendix provides foundational reinforcement learning concepts and mathematical formulations that support the policy learning algorithms employed in the main work.

A.2.1 Finite Markov Decision Process

The finite Markov decision process provides the mathematical framework for the reinforcement learning problem. It formalizes sequential decision-making as a discrete-time interaction between an agent and an environment, where the agent's choices influence not only immediate outcomes but also the trajectory of future states and rewards.

At each discrete time step $t \in \{0, 1, 2, ...\}$, the agent observes the current state $S_t \in \mathcal{S}$, where $\mathcal{S}$ denotes the finite set of all possible states. Based on this observation, the agent selects an action A_t from the set of available actions $\mathcal{A}(S_t) \subseteq \mathcal{A}$, where $\mathcal{A}$ represents the complete action space. The environment responds by transitioning to a new state S_{t+1} and providing a reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$, where $\mathcal{R}$ is the finite set of possible rewards. The evolution of these components over time generates trajectories of the form $(S_0, A_0, R_1, S_1, A_1, R_2, S_2, ...)$, where each realization is determined stochastically according to underlying probability functions that govern the system dynamics. For instance, the transition from state S_t to S_{t+1} and the corresponding reward R_{t+1} both depend probabilistically on the current state S_t and the chosen action A_t.

This stochastic nature means that repeated interactions with the same initial conditions can produce different trajectories, each representing a possible realization of the system. The goal is to understand and optimize the agent's behavior within this probabilistic framework, which we construct incrementally by progressively enriching the model with additional structure.

Markov Process. The starting point for an incremental resolution of the problem in question is to understand it first as a Markov process that describes the stochastic transitions of a series of states. Specifically, each of these states contains relevant information regarding the interaction of the agent with the environment. Thus, it is nothing further away than a first-order Markov chain, where the probabilities of transition between states are captured by a transition probability function $p : \mathcal{S} \times \mathcal{S} \to [0, 1]$ and can be denoted as follows, thanks to the fact that the Markov property holds:

$$P(S_{t+1} = s' | S_t = s, S_{t-1}, ..., S_0) = P(S_{t+1} = s' | S_t = s) = p(s' | s).$$

This first model, formed by $\mathcal{S}$, heads the most basic stage of the description: the dynamics of states are known, but there is no notion of an objective or cumulative return.

Markov Reward Process. To make progress in solving the problem, the model must be extended to capture not only the evolution of states, but also the impact they have in terms of reward. By adding a reward $R_t \in \mathcal{R} \subset \mathbb{R}$ and an expected reward function $r : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$, defined as $r(s, s') = \mathbb{E}[R_t | S_{t-1} = s, S_t = s']$, along with a discount factor $\gamma \in [0, 1]$, the Markov process becomes a Markov reward process. The transition probability function naturally extends to $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \to [0, 1]$ to account for the reward dimension.

In this context, the total cumulative return from time t is defined as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

and the state value, understood as the expected future return from being in state s, is defined by:

$$v(s) = \mathbb{E}[G_t | S_t = s].$$

This value satisfies Bellman's equation:

$$v(s) = \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s) \left[r + \gamma v(s') \right].$$

This second stage of construction introduces the notion of trajectory evaluation, but it does not yet allow us to influence the dynamics of the process, since there are no actions.

Markov Decision Process. The final extension needed to complete the model and obtain a true decision problem is the introduction of actions and a mechanism for selecting them. At each state $s \in \mathcal{S}$, the agent can choose from a set of available actions $\mathcal{A}(s) \subseteq \mathcal{A}$, where the selection is governed by a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ that maps state-action pairs to probabilities.

This extension converts the Markov Reward Process into a Markov Decision Process, where the dynamics now depend on both the current state and the chosen action. The transition probability function extends to $p : \mathcal{S} \times \mathcal{A} \times \mathcal{R} \times \mathcal{S} \to [0, 1]$, capturing the joint probability of transitioning to state s' and receiving reward r given that action a was taken in state s:

$$p(s', r|s, a).$$

This change allows us to define two fundamental objects and its corresponding recurrent forms:

- **State-value function under policy π:** $v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$,

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a)[r + \gamma v_\pi(s')],$$

- **Action-value function under policy π:** $q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$,

$$q_\pi(s, a) = \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a)[r + \gamma \sum_{a' \in \mathcal{A}(s')} \pi(a'|s') q_\pi(s', a')].$$

With this final extension, the model evolves from a simple stochastic process to a complete sequential decision-making problem, in which the objective is to determine a policy that maximizes the expected return.

Optimality and Approximation. Once the finite Markov decision process is fully defined, the central question becomes determining which policy provides the maximum expected return over time. A policy π is considered better than or equal to another policy π' if and only if $v_\pi(s) \geq v_{\pi'}(s)$ for all states $s \in \mathcal{S}$. This partial ordering over policies allows us to establish the existence of at least one optimal policy π_* that is better than or equal to all other policies.

From the definitions of the state-value and action-value functions, the Bellman optimality equations can be established, which characterize the optimal policy and the associated optimal values. Let $v_*(s)$ be the maximum value achievable from state s. This value satisfies:

$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a)[r + \gamma v_*(s')].$$

This equation establishes that the optimal value of a state corresponds to the maximum expected return over all available actions, considering both the immediate reward and the discounted future value of the successor states.

Equivalently, the optimal action-value function $q_*(s, a)$ satisfies:

$$q_*(s, a) = \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a)[r + \gamma \max_{a' \in \mathcal{A}(s')} q_*(s', a')].$$

These two forms, state-value and action-value, are coherent and define the same concept: finding the policy that maximizes the expected return at each decision point.

While the Bellman optimality equations provide the theoretical foundation for optimal policies, solving them exactly is often computationally intractable for real-world problems with large or continuous state spaces. Therefore, practical reinforcement learning algorithms such as dynamic programming, Monte Carlo methods, and temporal-difference techniques are employed to approximate these optimal value functions and derive near-optimal policies through iterative learning and interaction with the environment.

A.2.2 Policy Gradient Methods

The algorithms described above, such as dynamic programming, Monte Carlo methods, or temporal-difference techniques, constitute different ways to solve a Markov decision process by explicitly estimating value functions and approximating (or directly exploiting) Bellman's optimality equations. However, all these approaches share a common limitation: the explicit dependence on a model of the environment or on the dynamics that govern state transitions, which can be difficult to model accurately in complex real-world scenarios.

This limitation motivates an alternative approach based on directly parameterizing the agent's policy and optimizing it according to the expected return. This shift in perspective leads to policy gradient methods, where the policy $\pi_{\boldsymbol{\theta}}(a|s) = P(A_t = a|S_t = s, \boldsymbol{\theta}_\mathbf{t} = \boldsymbol{\theta})$ is governed by a set of parameters $\boldsymbol{\theta} \in \mathbb{R}^d$, with d representing the dimensionality of the parameter space. When a state-value function is used to reduce variance in gradient estimation, it is similarly parameterized as $v_\mathbf{w}(s)$ with parameters $\mathbf{w} \in \mathbb{R}^{d'}$.

In this framework, the fundamental quantity to be optimized is the objective function $J(\boldsymbol{\theta})$, defined as the expectation of the return that the agent obtains when following the parameterized policy. Formally:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[G_t].$$

The main interest lies in estimating the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$, which allows the parameters to be updated iteratively by:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)},$$

where α is the learning rate and $\widehat{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)}$ denotes the estimated gradient at iteration t.

Policy Gradient Theorem. A direct differentiation of $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ is not straightforward because the policy influences not only the immediate action selection but also the future state distribution encountered by the agent. This dependency creates a complex interplay between policy parameters and the states visited during execution.

The policy gradient theorem resolves this difficulty by providing an analytical expression for the gradient that can be estimated from experience. Specifically, the theorem establishes that:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \propto \sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}(s)} q_{\pi_{\boldsymbol{\theta}}}(s, a) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s),$$

where $\mu(s)$ represents the on-policy state distribution, defined as the stationary distribution of states encountered when following policy $\pi_{\boldsymbol{\theta}}$ in an episode. This formulation is particularly valuable because it expresses the gradient in terms of quantities that can be estimated through interaction with the environment, without requiring knowledge of the system dynamics or explicit differentiation through the state distribution.

REINFORCE: Monte Carlo Policy Gradient. From the policy gradient theorem, the REINFORCE algorithm emerges as a direct Monte Carlo approach to estimate the policy gradient. The key step is to rewrite the gradient expression by treating states and actions as random variables sampled from the trajectory distribution. Starting from the policy gradient theorem:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \propto \mathbb{E}_{\pi} \left[\sum_{a \in \mathcal{A}(S_t)} \pi_{\boldsymbol{\theta}}(a|S_t) q_{\pi_{\boldsymbol{\theta}}}(S_t, a) \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|S_t)}{\pi_{\boldsymbol{\theta}}(a|S_t)} \right],$$

where S_t is now a random variable distributed according to the on-policy distribution $\mu(s)$. By recognizing that the sum over actions is itself an expectation with respect to actions sampled from $\pi_{\boldsymbol{\theta}}(\cdot|S_t)$, this becomes:

$$= \mathbb{E}_{\pi} \left[q_{\pi_{\boldsymbol{\theta}}}(S_t, A_t) \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(A_t|S_t)}{\pi_{\boldsymbol{\theta}}(A_t|S_t)} \right], \quad \text{(replacing } a \text{ by the sample } A_t \sim \pi_{\boldsymbol{\theta}}\text{)}.$$

Finally, replacing the action-value function with the sample return G_t, since $\mathbb{E}_{\pi}[G_t|S_t, A_t] = q_{\pi_{\boldsymbol{\theta}}}(S_t, A_t)$:

$$= \mathbb{E}_{\pi} \left[G_t \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(A_t|S_t)}{\pi_{\boldsymbol{\theta}}(A_t|S_t)} \right].$$

This leads to the REINFORCE update rule[4]:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha G_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t).$$

This algorithm updates the policy parameters in the direction that increases the log-probability of actions that led to high returns, effectively reinforcing successful behavior through direct gradient ascent on the objective function.

[4]The gradient fraction can be equivalently expressed using the logarithm: $\frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(A_t|S_t)}{\pi_{\boldsymbol{\theta}}(A_t|S_t)} = \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t)$.

REINFORCE with Baseline. A well-known limitation of the REINFORCE algorithm is the high variance of its gradient estimates, which can lead to slow and unstable learning. This variance arises because the full return G_t can vary significantly across episodes, even for the same state-action pair. To address this issue, a baseline function $b(S_t)$ can be subtracted from the return without introducing bias to the gradient estimate, provided the baseline does not depend on the action:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi} \left[(G_t - b(S_t)) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) \right].$$

A natural and effective choice for the baseline is the state-value function $v_{\pi_{\boldsymbol{\theta}}}(S_t)$, which represents the expected return from state S_t. By subtracting this baseline, the algorithm updates parameters based on the advantage of taking action A_t relative to the average performance in that state. In practice, the true state-value function is unknown and must be estimated using the parametrized approximation $\hat{v}_{\mathbf{w}}(s)$ introduced earlier. This gives rise to the REINFORCE with baseline update:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha (G_t - \hat{v}_{\mathbf{w}}(S_t)) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t).$$

Actor-Critic Methods. This formulation represents an actor-critic architecture, where the parametrized policy $\pi_{\boldsymbol{\theta}}$ acts as the actor (selecting actions) and the value function approximation $\hat{v}_{\mathbf{w}}$ serves as the critic (evaluating states). The critic parameters $\mathbf{w}$ are learned simultaneously by minimizing the mean squared error between the predicted value and the observed return:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \beta (G_t - \hat{v}_{\mathbf{w}}(S_t)) \nabla_{\mathbf{w}} \hat{v}_{\mathbf{w}}(S_t),$$

where β is the learning rate for the critic. The quantity $(G_t - \hat{v}_{\mathbf{w}}(S_t))$ is known as the advantage estimate, quantifying how much better the actual return was compared to the expected value of the state.

While the Monte Carlo approach uses the full return G_t, actor-critic methods can leverage temporal-difference learning to reduce variance and enable online learning. Instead of waiting for complete episodes, the advantage can be estimated using the temporal-difference error:

$$\delta_t = R_{t+1} + \gamma \hat{v}_{\mathbf{w}}(S_{t+1}) - \hat{v}_{\mathbf{w}}(S_t),$$

which approximates the advantage using only the immediate reward and bootstrapped value estimates. This leads to the one-step actor-critic updates:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \delta_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t),$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \beta \delta_t \nabla_{\mathbf{w}} \hat{v}_{\mathbf{w}}(S_t).$$

This temporal-difference-based formulation allows the algorithm to update parameters at every time step rather than waiting for episode completion, making it more suitable for continuous or long-horizon tasks.

Policy Parametrization for Continuous Actions. In many practical applications, the action space is continuous rather than discrete, requiring a different approach to policy parametrization. Instead of learning a probability distribution over a finite set of actions, the policy must output a continuous distribution from which actions can be sampled. A common and effective choice is to model the policy as a Gaussian distribution, where the mean and standard deviation are themselves parametrized functions of the state.

Specifically, for a continuous action space $\mathcal{A} \subseteq \mathbb{R}^{d_a}$, where d_a is the dimensionality of the action space, the policy is defined as:

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{1}{\sigma(s, \boldsymbol{\theta})\sqrt{2\pi}} \exp\left(-\frac{(a - \mu(s, \boldsymbol{\theta}))^2}{2\sigma(s, \boldsymbol{\theta})^2}\right),$$

where $\mu : \mathcal{S} \times \mathbb{R}^d \to \mathbb{R}$ and $\sigma : \mathcal{S} \times \mathbb{R}^d \to \mathbb{R}^+$ are two parametrized function approximators, typically implemented as neural networks. Here, $\mu(s, \boldsymbol{\theta})$ represents the mean of the Gaussian distribution conditioned on state s, while $\sigma(s, \boldsymbol{\theta})$ represents the standard deviation, controlling the exploration-exploitation trade-off.

In this formulation, the parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$ contains all learnable weights for both the mean and standard deviation networks. During training, actions are sampled from this distribution: $A_t \sim \mathcal{N}(\mu(S_t, \boldsymbol{\theta}), \sigma(S_t, \boldsymbol{\theta})^2)$, and the policy gradient methods described previously can be applied by computing $\nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t, \boldsymbol{\theta})$. This parametrization allows the agent to learn both the preferred action (through μ) and the degree of stochasticity (through σ) in a state-dependent manner, adapting its behavior to different regions of the state space.

References

[1] Walter Bagehot. "The only game in town". In: *Financial Analysts Journal* 27.2 (1971), pp. 12–14.

[2] Rolf W Banz. "The relationship between return and market value of common stocks". In: *Journal of Financial Economics* 9.1 (1981), pp. 3–18.

[3] Sanjoy Basu. "Investment performance of common stocks in relation to their price-earnings ratios: A test of the efficient market hypothesis". In: *The Journal of Finance* 32.3 (1977), pp. 663–682.

[4] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828. DOI: 10.1109/TPAMI.2013.50.

[5] BitMEX. *BitMEX WebSocket API Documentation*. 2025.

[6] Jean-Philippe Bouchaud, J Doyne Farmer, and Fabrizio Lillo. "How markets slowly digest changes in supply and demand". In: *Handbook of Financial Markets: Dynamics and Evolution* (2008), pp. 57–160.

[7] Mark M Carhart. "On persistence in mutual fund performance". In: *The Journal of Finance* 52.1 (1997), pp. 57–82.

[8] Roger Clarke, Harindra De Silva, and Steven Thorley. "The fundamental law of active management: Redux". In: *The Journal of Portfolio Management* 28.2 (2002), pp. 10–20.

[9] Rama Cont, Sasha Stoikov, and Rishi Talreja. "A stochastic model for order book dynamics". In: *Operations Research* 58.3 (2010), pp. 549–563.

[10] Rama Cont, Sasha Stoikov, and Rishi Talreja. "The price dynamics of financial assets: A survey of microstructure models". In: *Quantitative Finance* 11.7 (2011), pp. 991–1010.

[11] David A Dickey and Wayne A Fuller. "Distribution of the estimators for autoregressive time series with a unit root". In: *Journal of the American Statistical Association* 74.366 (1979), pp. 427–431.

[12] Robert F Engle. "Autoregressive conditional heteroscedasticity with estimates of the variance of UK inflation". In: *Econometrica* 50.4 (1982), pp. 987–1007.

[13] Eugene F Fama. "Efficient capital markets: II". In: *The Journal of Finance* 46.5 (1991), pp. 1575–1617.

[14] Eugene F Fama and Kenneth R French. "A five-factor asset pricing model". In: *Journal of Financial Economics* 116.1 (2015), pp. 1–22.

[15] Eugene F Fama and Kenneth R French. "Common risk factors in the returns on stocks and bonds". In: *Journal of Financial Economics* 33.1 (1993), pp. 3–56.

[16] Eugene F. Fama. "Efficient Capital Markets: A Review of Theory and Empirical Work". In: *The Journal of Finance* 25.2 (1970), pp. 383–417.

[17] Mark B Garman. "Market microstructure". In: *Journal of Financial Economics* 3.3 (1976), pp. 257–275.

[18] Lawrence R Glosten and Paul R Milgrom. "Bid, ask and transaction prices in a specialist market with heterogeneously informed traders". In: *Journal of Financial Economics* 14.1 (1985), pp. 71–100.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[20] Martin D Gould et al. "Limit order books". In: *Quantitative Finance* 13.11 (2013), pp. 1709–1742.

[21] Richard C Grinold and Ronald N Kahn. *Active portfolio management*. 2nd. New York: McGraw-Hill, 1999.

[22] Larry Harris. *Trading and exchanges: Market microstructure for practitioners*. New York: Oxford University Press, 2003.

[23] Stefan Jansen. *Machine Learning for Algorithmic Trading*. 2nd ed. Birmingham, UK: Packt Publishing, 2020.
[24] Carlos M Jarque and Anil K Bera. "A test for normality of observations and regression residuals". In: *International Statistical Review* 55.2 (1987), pp. 163–172.
[25] Narasimhan Jegadeesh and Sheridan Titman. "Returns to buying winners and selling losers: Implications for stock market efficiency". In: *The Journal of Finance* 48.1 (1993), pp. 65–91.
[26] Michael I Jordan et al. "An Introduction to Variational Methods for Graphical Models". In: *Machine Learning* 37.2 (1999), pp. 183–233. DOI: 10.1023/A:1007665907178.
[27] Alec N Kercheval and Yuan Zhang. "Modelling high-frequency limit order book dynamics with support vector machines". In: *Quantitative Finance* 15.8 (2015), pp. 1457–1467.
[28] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *arXiv preprint arXiv:1312.6114* (2014).
[29] Denis Kwiatkowski et al. "Testing the null hypothesis of stationarity against the alternative of a unit root". In: *Journal of Econometrics* 54.1–3 (1992), pp. 159–178.
[30] John Lintner. "The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets". In: *The Review of Economics and Statistics* 47.1 (1965), pp. 13–37.
[31] Greta M Ljung and George EP Box. "On a measure of lack of fit in time series models". In: *Biometrika* 65.2 (1978), pp. 297–303.
[32] Andrew W Lo. "The adaptive markets hypothesis: Market efficiency from an evolutionary perspective". In: *Journal of Portfolio Management* 30.5 (2004), pp. 15–29.
[33] Andrew W Lo. "The statistics of Sharpe ratios". In: *Financial Analysts Journal* 58.4 (2002), pp. 36–52.
[34] Marcos López de Prado. *Advances in Financial Machine Learning*. Wiley, 2018.
[35] Harry Markowitz. "Portfolio selection". In: *The Journal of Finance* 7.1 (1952), pp. 77–91.
[36] Jan Mossin. "Equilibrium in a capital asset market". In: *Econometrica* 34.4 (1966), pp. 768–783.
[37] Maureen O'Hara. *Market microstructure theory*. Cambridge, MA: Blackwell Publishers, 1995.
[38] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. "Neural Discrete Representation Learning". In: *arXiv preprint arXiv:1711.00937* (2017).
[39] Aäron van den Oord et al. "WaveNet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* (2016).
[40] Ryan A. Peterson. "Finding Optimal Normalizing Transformations via bestNormalize". In: *The R Journal* 13.1 (2021), pp. 310–329.
[41] Marc R Reinganum. "Misspecification of capital asset pricing: Empirical anomalies based on earnings' yields and market values". In: *Journal of Financial Economics* 9.1 (1981), pp. 19–46.
[42] John Schulman et al. "High-dimensional continuous control using generalized advantage estimation". In: *International Conference on Learning Representations*. 2016.
[43] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).
[44] John Schulman et al. "Trust region policy optimization". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1889–1897.
[45] William F Sharpe. "Capital asset prices: A theory of market equilibrium under conditions of risk". In: *The Journal of Finance* 19.3 (1964), pp. 425–442. DOI: 10.2307/2977928.

[46] William F Sharpe. "Mutual fund performance". In: *The Journal of Business* 39.1 (1966), pp. 119–138.
[47] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2nd. MIT press, 2018.
[48] Ruey S Tsay. "Nonlinearity tests for time series". In: *Biometrika* 73.2 (1986), pp. 461–466.
[49] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).