POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Latent Variable-based Reinforcement Learning for FX Trading

Laurea Magistrale in Computer Science Engineering - Ingegneria Informatica

Author: Adriano Mundo

Advisor: Prof. Marcello Restelli

Co-advisor: Riccardo Poiani

Academic year: 2021-2022

1. Introduction

The Foreign Exchange, also known as Forex or FX, dominates as the world's largest financial market. It is a decentralized global marketplace where currencies are bought and sold simultaneously 24 hours daily from Sunday to Friday. According to the Triennial Central Bank Survey of Foreign Exchange by the Bank of International Settlements (BIS), the FX market has a volume of traded instruments up to $7.5 trillion per day, surpassing the stock market in size. The motivations behind why FX trading is commonly pursued are the facilitation of international trade and commerce, hedging against risks and speculation. The technology supporting FX has completely evolved in the last two decades. The advancement in computational power, coupled with the availability of high-frequency data, has facilitated the development of new advanced strategies. Indeed, classical algorithmic trading and, more recently, Machine Learning (ML) techniques have been extensively studied in financial markets with application to FX, equities and commodities aiming to construct systems able to generate profits by outperforming human traders. The trading problem, independently from the asset class, can be framed mathematically as a Sequential Decision Problem (SDP) where an agent must continuously decide what action to perform to maximize returns. It is possible to model such a problem in a discrete-time setting as a Markov Decision Process (MDP), where the agent collects information from the environment and selects the position to hold at each time step. If the underlying model is known, the MDP can be solved through Dynamic Programming (DP), but since the dynamics of the FX market are uncertain, Reinforcement Learning (RL) must be used [12].

The *scope* of the thesis was to investigate the effectiveness of RL techniques enhanced by latent variable models in a non-stationary setting. The intuition was that the latent models could extract relevant information for learning from the latent variables. Those are not directly observable but are inferred from other observable variables and used to represent underlying factors that cannot be measured. Then, the model was applied to FX trading. Therefore, the work sought to answer the two **research questions**: *what is the impact of incorporating latent variables on the performance of a batch-RL algorithm in financial markets? Can an RL agent be trained to learn trading strategies that outperform human traders and algorithmic strategies?*

The study experimented with two of the most liquid currency pairs: EUR/USD and USD/JPY, to trade small sizes intraday without any market impact. The focus was on the application of a batch-RL algorithm, Fitted-Q-Iteration (FQI) [5], which is efficient in an offline setting with historical data, and its extensions with action persistence (PFQI) [8]. It is a model-free and off-policy algorithm that learns the optimal policy without interacting with the environment. Indeed, directly employing model-based planning approaches can lead to significant errors when the underlying model is imperfect. In the thesis, FQI has been extended with latent vari-

ables, namely Latent Variable (Persistent) Fitted-Q-Iteration (LV-FQI, LV-PFQI), where the latent representation was extracted through Variational Auto-Encoders (VAE) models. VAE help obtains a set of latent features representing the time series where only the most influential and significant aspects are kept. The resulting model demonstrates the ability to leverage the structure of future temporal sequences effectively. This implies the algorithm can capture and utilise the inherent patterns and dynamics of the non-stationary time series without relying on explicit model-based assumptions.
The *contribution* of the study is two-fold and mainly empirical. On one side, it provides a performance comparison of LV-FQI, and its persistent version, augmented with latent variables, in a realistic trading scenario based on FX historical data and transaction fees, against existing RL algorithms and trading baselines. On the other side, it presented a two-step approach for combining the training of deep VAE for learning compact feature spaces with state-of-the-art batch-RL algorithms to learn trading strategies (or policies) that can capture the structure of future temporal sequences of exchange rates.

2. Background

This section provides the necessary theoretical background in order to understand the proposed approach.

2.1. Reinforcement Learning

Reinforcement Learning (RL) is one of the main paradigms of ML alongside Supervised and Unsupervised Learning. It focuses on developing artificial agents able to learn from sequential interaction with the environment. A sequential decision-making problem, if the state is completely observable and Markovian, is often mathematically modelled as a *Markov Decision Process* (MDP) [9]. A discrete-time MDP is defined as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu \rangle$ where $\mathcal{S}$ is a measurable state-space containing all the states, $\mathcal{A}$ is a measurable action-space containing all the actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition model, or kernel, that assigns to each state-action pair (s, a) a probability measure $\mathcal{P}(\cdot|s, a)$ of the next state, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, which assigns to every (s, a) a probability measure $\mathcal{R}(\cdot|s, a)$, $\gamma \in [0, 1)$ is the discount factor to weight future reward and $\mu : \mathcal{S} \to \Delta(\mathcal{S})$ is the initial state distribution from which the starting state is sampled. Here $\Delta(\mathcal{S})$ denotes the set of probability measures over $\mathcal{S}$.
To take actions, the agent follows a *policy* π that associates to each state a probability over the actions; hence it is defined as $\pi : S \to \pi(\cdot|s) \in \mathcal{P}(A)$. Instead, to evaluate the utility of a state is used the *state-value function* of a given policy π, formally defined as $V^\pi = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \pi]$ where r_t is the immediate observed reward at time step t. The expectation is over all the possible trajectories starting in state s and by following the policy π. A more helpful quantity used in practice is the *action-value function*, or $Q-$function, which allows stating the first action explicitly, and it is defined as $Q^\pi = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a, \pi]$.

2.2. Batch Reinforcement Learning

RL often requires having a simulator for the agent to interact with. This is not always available because it is either nonexistent or hard to build due to the complex unknown systems dynamics. *Batch-RL*, also known as *offline-RL*, aims at filling this gap by providing algorithms that can learn near-optimal control policy from a fixed dataset without additional interaction with the environment [7]. There exist two different settings in batch-RL: *pure batch* where the agent learns a policy from the pre-collected dataset without interacting with the environment during training; *growing batch* where the agent can occasionally interact with the environment with the latest policy to build the batch incrementally.
This thesis focuses on the finite horizon or *episodic* MDP, where the agent learns to optimize its behaviour within the context of each episode. They can also be used in batch-RL, where a fixed dataset of episodes is collected and used. Therefore, in practice, it is convenient to define a problem with a distinct beginning and end via *episodic* MDP, where the agent interacts with the environment over a series of discrete episodes, and each episode consists of a finite number of time steps, or horizon T. The goal is to learn an optimal policy π^* is the one that maximizes the expected sum of rewards over all possible episodes:

$$\pi^* = \arg\max_\pi J(\pi) = \arg\max_\pi \mathbb{E}\left[\sum_{t=1}^{T} r_t \mid \pi\right] \quad (1)$$

where $J(\pi)$ is the sum of rewards from time step t to the end of the episode.

2.3. Fitted Q-Iteration

Fitted-Q-Iteration (FQI) [5] is a model-free, off-policy, and offline algorithm that learns an approximation of the optimal action-value function Q^* starting from a set of experience samples collected in the dataset $\mathcal{D} = \{(s_t^k, a_t^k, s_{t+1}^k, r_{t+1}^k) | k = 1, 2, ..., |\mathcal{D}|\}$ where s_{t+1} is the state that the agent reaches after applying action a_t in state s_t while collecting reward r_{t+1} for this transition. At the N-th iteration, given Q-function approximated at the previous iteration $Q_{N-1}(s, a) \forall (s, a), Q_N$ is trained on the following training set:

$$TS_{FQI} = \{(i^k, o^k) | k = 1, 2, ..., |\mathcal{D}|\} \quad (2)$$

where the input is the state-action pair $i^k = (s_t^k, a_t^k)$ and the output is:

$$o^k = r_{t+1}^k + \gamma \max_{a \in \mathcal{A}} Q_{N-1}(s_{t+1}^k, a) \tag{3}$$

FQI can be understood intuitively as expanding the optimization horizon at each iteration performing regression. However, an increasing number of iterations may lead to a larger planning horizon and propagation of errors, so the Q-function may not converge to Q^*. *Persistent Fitted-Q-Iteration* (PFQI) [8] extends the original algorithm taking into account the possibility of persisting actions, which means repeating each action for a certain number of consecutive steps. A higher persistence gives the agent more control but decreases the signal-to-noise ratio and negatively impacts sample complexity. Lastly, it influences the optimization horizon since it requires fewer iterations to reach the same horizon. Formally, *persistence* can be seen as an environmental parameter k that can be configured to generate a family of related decision processes $\mathcal{M}_k = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}_k, \mathcal{R}_k, \gamma^k, \mu \rangle$ that whenever an action is issued, the resulting transition lasts for k steps, with all the one-step rewards collected with discount in the new distribution $\mathcal{R}_k$.

2.4. Variational Auto-encoders

Variational Auto-encoders (VAE) [6] are latent variable models trained to ensure that their latent space has general properties to generate new data. VAE combines elements from different research areas. The term variational stems from the relationship between the regularization and the variational inference in statistics, while auto-encoders from the *encoder-decoder* architecture used to produce new features in a low-dimensional space, also called *latent space*, and the reverse process of decompressing the space into the original input while retaining as much information as possible, as described in Fig. 1.
In a formal setting, we have a dataset X and a vector of *latent variables* z in a high-dimensional space $\mathcal{Z}$ that we can sample according to some probability density function $\mathcal{P}(z)$ over $\mathcal{Z}$. The wish is to optimize θ such that we can sample z from $\mathcal{P}(z)$ and, with high probability $\mathcal{P}(X|z;\theta)$ will be as close as possible to X.

$$\mathcal{P}(X) = \int \mathcal{P}(X|z;\theta)\mathcal{P}(z)dz \tag{4}$$

Theoretically, by sampling a large number of z values $\{z_1, z_2, \ldots, z_n\}$, it is possible to approximately compute the probability $\mathcal{P}(X) \approx \frac{1}{n}\sum_{i=1}^{n} \mathcal{P}(X|z_i)$. Instead, in practice, for most z, the value of $\mathcal{P}(X|z)$ will be nearly zero, hence contributing almost nothing to the estimate of $\mathcal{P}(X)$. The key idea behind the VAE is to attempt sampling values of z that are likely to have produced X and compute $\mathcal{P}(X)$ from those.

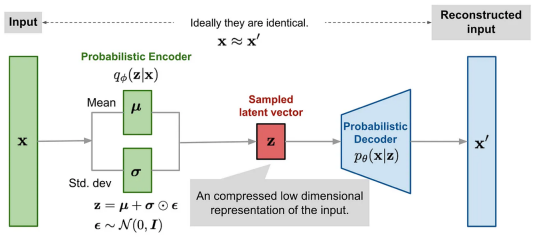


Figure 1: Architecture of a VAE, where **X** and **X'** represents respectively the input and the reconstructed input; q_ϕ and p_ϕ the encoder, decoder probabilities, z the reparameterization trick and ϵ the Gaussian noise.

Thus, it is needed to define a new function $\mathcal{Q}(z|X)$ which can take a value of X and give us a distribution over z values that are likely to produce X, the *latent distribution*. In this way, $\mathcal{Q}$ is "encoding" X into z, and $\mathcal{P}$ is "decoding" z to reconstruct X. Therefore, the objective is two-fold: maximizing $\log \mathcal{P}(X)$ while simultaneously minimizing $\mathcal{KL}[\mathcal{Q}(z|X)\|\mathcal{P}(z|X)]$, the $\mathcal{KL}-$*divergence*.
To compute the optimization can be sampled a single value of X and a single value of z from the distribution $\mathcal{Q}(z|X)$, and consequently computed the gradient of:

$$\log \mathcal{P}(X|z) - \mathcal{KL}[\mathcal{Q}(z|X)\|\mathcal{P}(z)] \tag{5}$$

Then, the result will converge by averaging the gradient of the function over arbitrarily many samples of X and z.

3. Related Works

This section briefly overviews RL techniques applied to FX trading, focusing on those relevant to our work. The core idea behind *Evolutionary RL* (ERL) is that *genetic algorithms* (GA) can improve the performance of RL trading systems by finding a suitable state representation; hence these works try to combine value-based methods with such algorithms. The *Recurrent RL* (RRL) approach represents the first attempt to overcome the limitation of supervised learning methods by creating a system that combines recurrent neural networks and policy-based mechanisms to outperform value-based methods and maximize profit. Instead, with the availability of computational power, there has been the emergence of *Deep Reinforcement Learning* (DRL), which extends the RRL approach by using deep neural networks to handle the diversity of market dynamics and directly optimizing trading strategies.
Studies that share more similarities with our work fall into the category of the *Batch-RL* (BRL). The first attempt at using FQI for finding a trading strategy has been proposed by framing the problem to include risk aversion as a *multi-objective* MDP [1]. To build the Pareto frontier of different financial objectives, they implemented *Multi-Objective* FQI con-

sidering as risk measure the reward volatility, which evaluates the uncertainty about the step-by-step rewards obtained from the environment. Thus, profit maximization alongside risk minimization creates a multi-objective optimization problem. Experimental results demonstrated that the agent can identify profitable temporal patterns exploited to maximize returns. An extension of the previous work introduced a simultaneous multi-currency trading framework via FQI with extra trees, focusing on the two-currencies and three-currencies models [10]. Their contribution is two-fold. On the one hand, they analyzed the performance on a multi-asset scenario; on the other hand, they evaluated performance with different trading frequencies. They studied the importance of tuning the control frequency by proposing the idea of action persistence, which is crucial to obtain effective policies and exploiting the best opportunities. The most recent approach tried to overcome the financial market regime switches issue by modelling the trading task as a non-stationary RL problem [11]. Their two-layer approach allows choosing algorithmically the best strategy among a set of RL models with an online-learning method. They employed FQI with XGBoost as a regressor for the RL layer and *Optimistic Adapt ML-Prod* for online learning. Hence, the model selection procedure becomes dynamic and choosing a strategy based on its performance in past regimes is unnecessary.

4. Research Methods

In this thesis, we approach the FX trading problem from a *machine learning* perspective, using RL value-based techniques and following the pipeline in Fig 2. It consists of five steps that must be followed to ensure proper execution and accurate performance assessment.

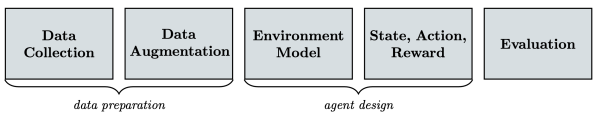


Figure 2: RL pipeline workflow for single asset trading problem.

4.1. Problem Formulation

This section aims to define the *trading* task for a single FX currency pair and explain how it can be mapped into a mathematical framework.

Definition 4.1. *(Trading). Given an asset to trade, trading can be defined as a sequential decision process in which at each (discrete) round* $t \in \{1, \dots, T\}$ *over a trading horizon* $T \in \mathbb{N}$*, a trader decides whether to go long, short or stay flat with respect to the asset to maximize his wealth. The trader's position is represented by the action* $a_t \in \{-1, 0, 1\}$.

The asset is a *currency pair* in our problem. It refers to the exchange rate between two currencies. For example, the EUR/USD currency pair represents the exchange rate between the Euro and the US Dollar. In this case, the EUR is the *domestic, or base* currency, and the USD is the *quote, or foreign* currency. An exchange rate indicates how much foreign currency is needed to buy one unit of the base currency. Our *problem formulation* aims to maximize the profits obtained in the domestic currency. Therefore, we trade a variable amount of foreign currency for some fixed amount of the base one, assuming USD as the base currency. Returns must be expressed in the same currency to evaluate performance adequately; hence the collected rewards are converted into the base currency at the end of a trading day. To make a realistic assumption, we are considering transaction costs but no market impact or slippage; we operate on highly liquid instruments of small size; therefore, no decision around the allocation size has to be made.

4.2. Data Preparation

4.2.1 Data Collection

This work experiments on one-minute bar FX data of EUR/USD and USD/JPY currency pairs from 2019 to 2022. The dataset covers the period from Monday to Friday since the markets are closed on Saturday and Sunday. It includes date and time information that allows for temporal analysis. At the same time, the minute bars provide price information on the opening, closing, highest, and lowest quotes for each minute (OHLC), which is crucial for developing and testing trading strategies.

4.2.2 Data Augmentation with Latent Variables

The proposed framework involves evaluating VAE [6] architectures, namely *TimeVAE* [4] and *LSTM-VAE* [3], to extract the latent representation from the encoder output. The *latent variables* are used as features for FQI to improve the learning of trading strategies in a non-stationary setting and demonstrate they can capture the underlying patterns to better exploit the temporal structure during regression. VAEs are trained using a specific loss function that consists of two parts: the *reconstruction* loss and the $\mathcal{KL}$*-divergence* loss, formally $L_{total} = L_{recon} + L_{\mathcal{KL}}$. The *reconstruction* loss measures how well it can reconstruct the input data from the variables generated by the encoder as mean squared error, while the $\mathcal{KL}$*-divergence* loss measures the difference between the distribution of the latent variables and a predefined prior distribution.

TimeVAE. It is a model for multivariate time series generation with several properties: interpretability,

ability to encode domain knowledge, and reduced training times. The architecture uses a combination of traditional deep learning layers and custom layers to model time-series specific components such as multi-polynomial trends and seasonal patterns [4].

LSTM-VAE. It is a deep learning model that extracts features from time series data by combining tLSTM and VAE [3]. In this architecture, LSTM is a type of recurrent neural network responsible for learning and representing long-term dependencies in sequential data, while VAE learns a compressed representation of the data.

4.3. Agent Design

In this section on *agent design*, we focus on the process of designing an RL agent as an *episodic* (MDP). To do that, we must consider how the environment is described as an MDP and the actions, states, and rewards.

4.3.1 Environment

The single currency pair trading problem can be modelled as an *episodic* MDP, where each episode corresponds to a trading day composed of 1230 time steps. There are two reasons for the choice of the fixed-length episode: it allows for the undiscounted setting ($\gamma = 1$); closing all positions before the end of the day is more practical from a financial standpoint.

4.3.2 State

The agent *state* design depends on the specific problem. A critical aspect is the *Markov property*, so the state should contain all relevant information related to past market observations that may affect the transition to the next state. Therefore, we included: the last 60 exchange rate variations between consecutive minutes computed as the differences between the price at a certain time-step and the previous one, normalized by the value of the former one: $d_{k,t} = \frac{p_{t-k+1} - p_{t-k}}{p_{t-k}}$ where t is the time, p the reference price and $\{k = 1, 2, \ldots, 60\}$; the portfolio position x_t of the previous time-step; the current time until the end of the trading day; the day of the week as a number between 1 (Monday) and 5 (Friday); the vector $\lambda_z = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ containing n estimated latent variables from the underlying process.

4.3.3 Action

The *action* is the allocation the agent wants to keep for the next minute or the next ρ minutes if action persistence $\rho > 1$. The action space is discrete, and at time t, a_t is the portfolio position. Thus, the set of available actions can be defined as {-1, 0, 1} whose elements correspond to *buy, sell*, and *hold*.

4.3.4 Transition Probability

The *transition probability* refers to the probability $\mathcal{P}(s'|s, a)$. In our case, the action affects only the portfolio feature, so we have a deterministic transition $x_{t+1} = a_t$. In contrast, all the other features are exogenous, and the action does not affect their value.

4.3.5 Reward

Given the current portfolio allocation x_t, the current exchange rate p_t, the action taken a_t, the next exchange rate p_{t+1}, and the fee rate ϕ, the *reward* received by the agent at persistence ρ is defined as: $r_{t+1} = t(p_{t+\rho} - p_t) - \phi|a_t - x_t|$. The first term consists of the gain (or loss) associated with the exchange rate variation, whereas the second corresponds to the transaction costs that must be paid ($\phi = 2 \cdot 10^{-5}$).

4.3.6 Latent Variable FQI

FQI is a model-free, off-policy, and offline algorithm that allows an RL agent to learn the optimal policy without interacting with the environment but by regression starting from the training set $\mathcal{D}$, refer to Sec. 2.3. Therefore, a novelty of this work was to include *latent variables* extracted via VAE into the FQI features and consequently as part of the agent state. Hence, it is needed to enlarge the set $\mathcal{D}$ to include a vector λ_z and define $\mathcal{D}_\lambda$ as: $\mathcal{D}_\lambda = \left\{(s_t^k, a_t^k, \lambda_{z,t}^k, s_{t+1}^k, r_{t+1}^k) \mid k = 1, 2, ..., |\mathcal{D}_\lambda|\right\}$ where $\lambda_{z,t}$ correspond to a latent features vector $\lambda_{z,t} = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ at time t with n number of extracted features.

4.4. Evaluation

The evaluation process of RL algorithms is critical to validate their effectiveness and robustness. It consisted of two parts: an idea validation phase with synthetic data generated and a following experimental phase with FX data.

4.4.1 Stochastic Models

The financial models used are respectively the *Vasicek* and the *Geometric Brownian Motion* (GBM).
The *Vasicek* model is a stochastic process used in finance to model the evolution of interest rates over time that assumes the short-term interest rate follows a mean-reverting process given by the following equation: $dS_t = a(\mu - S_t)dt + \sigma dW_t$, where S_t is the price of the underlying asset at time t, a the speed of mean reversion, μ the long term mean level, σ the volatility and W_t is the Wiener process.
The *GBM* is a stochastic process used to model the dynamics of stock price following a Brownian motion process with a drift, given by the equation $dS_t = \mu S_t dt + \sigma S_t dW_t$, where S_t is the price of the

underlying asset at time t, W_t is Brownian motion, μ the drift and σ the volatility.

4.4.2 Metrics

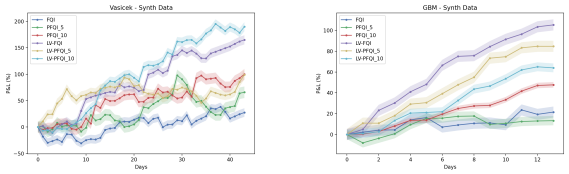
This section presents the evaluation *metrics* used to assess the performance of the RL algorithm in both synthetic and FX scenarios. The most important is the *cumulative P&L* that when considering a fixed action size and without reinvesting gains/losses, the cumulative P&L becomes a sum $W_T = \sum_{t=1}^{T} r_t$, where r_t is the reward. Other important financial metrics are: the *Sharpe Ratio* (SR) calculated as $\mathcal{SR} = \frac{W_A - r_f}{vol_A}$ where W_A is the annualized cumulative return and r_f is the risk-free rate; and the *Maximum Drawdown* ($\mathcal{MDD}$) which is the maximum observed loss from peak performance to the trough before a new peak is attained, formally $\mathcal{MDD} = \frac{TroughValue - PeakValue}{PeakValue}$.

5. Experimental Results

This chapter presents the empirical results obtained from our Latent-Variable extension of FQI, trained on synthetic and FX market data. It discusses the findings that emerged from our experiments.

5.1. Synthetic Data

The synthetic experiments generated data from predefined values where we hard-coded the latent variables, allowing us to evaluate the impact of using latent variables on the performance in a controlled setting. We assumed a prices environment where the agents have access to the lagged shift price rate variations normalized at the first price of the day. Figure 3 shows the outperforming performance of LV-PFQI with persistence ten and LV-FQI compared to the best model without the latent variables. Furthermore, even the worst model with latent variables is way above the best without.
The results confirmed our assumption that using latent variables effectively enhances the model performance providing a solid foundation for the subsequent estimation of latent variables from the FX data.



(a) Vasicek model. (b) GBM model.

Figure 3: Performance comparison of cumulative P&L for Latent Variable models against standard FQI with synthetic data.

Currency Pair	Model	Total Loss	Total Loss (Testing)
EUR/USD	TimeVAE	0.008	0.024
	LSTM-VAE	0.093	0.104
	stacked-LSTM-VAE	0.256	0.555
	attention-LSTM-VAE	0.354	0.653
USD/JPY	TimeVAE	0.015	0.033
	LSTM-VAE	0.101	0.147
	stacked-LSTM-VAE	0.351	0.550
	attention-LSTM-VAE	0.649	0.948

Table 1: Performance of VAE models for EUR/USD and USD/JPY on train and testing data. TimeVAE has the lowest total loss, while stacked-LSTM-VAE and attention-LSTM-VAE have higher total losses compared to TimeVAE and LSTM-VAE.

5.2. VAE & Latent Variables

Existing methods fail to account for non-stationary environments where time series are time-varying. Therefore, we experimented with VAE [6] to derive a compressed time-series representation that contained the essential information for the modelling phase. The objective was to evaluate the performance in capturing the underlying time-series structure with an informative latent representation. We selected the best-performing as the *oracle* model for extracting latent variables from the encoder output provided, denoted as z. Specifically, we tested TimeVAE [4] and LSTM-VAE [3].
We performed extensive hyperparameter tuning for all architectures, but the dimensionality of the latent features was the most critical to tune. We found that a dimension of 4 was the most effective. Thus, the latent representation generated by the encoder had four dimensions, providing a good representation retaining the essential features required for offline RL. TimeVAE was the most performing model for both currency pairs, as shown in Table 1. For EUR/USD, the best model had a batch size of 32 and a latent space dimension of 4. The model was trained for 30 epochs using an Adam optimizer with a learning rate of 0.001, and early stopping was used to prevent overfitting. For USD/JPY, the best model had the same characteristics but a batch size of 64.

(a) EUR/USD Training. (b) EUR/USD Testing.

(c) USD/JPY Training. (d) USD/JPY Testing.

Figure 4: Best TimeVAE for EUR/USD and USD/JPY time-series reconstruction in train: 2020-21 and testing: 2022.

5.3. FX Data

After outlining the synthetic experiments and latent variable estimation, this section will go into FX.

Hyperparameters				Performance		
Algorithm	Persistence	MinChildWeight	Iterations	P&L	SR	MDD
FQI	1	40.000	5	6.5 ± 2.26	1.75	4.23
		60.000	**3**	**7.9 ± 2.5**	**1.8**	**5.19**
		80.000	4	0.9 ± 3.28	0.22	2.0
PFQI	5	**10.000**	**2**	**11.8 ± 0.6**	**2.88**	**0.6**
		20.000	3	6.6± 1.26	1.75	4.2
		40.000	2	8.5 ± 2.21	2.7	1.22
PFQI	10	500	1	2.2 ± 2.3	0.35	1.25
		5000	**2**	**15.4 ± 1.8**	**4.05**	**1.26**
		60.000	2	3.3 ± 2.4	0.65	2.8
LV-FQI	1	10.000	2	8.9 ± 2.43	1.42	0.95
		40.000	5	11.1± 1.3	2.5	44.9
		80.000	**2**	**26.8 ± 3.4**	**2.35**	**0.95**
LV-PFQI	5	10.000	3	16.6 ± 1.36	3.17	0.47
		20.000	3	14.1 ± 1.46	2.12	2.5
		40.000	**3**	**31.8 ± 2.2**	**3.0**	**1.08**
LV-PFQI	10	500	2	15.4 ± 3.25	4.1	30.6
		20.000	2	8.6 ± 2.24	2.49	3.28
		40.000	**2**	**15.8 ± 2.6**	1.38	1.2

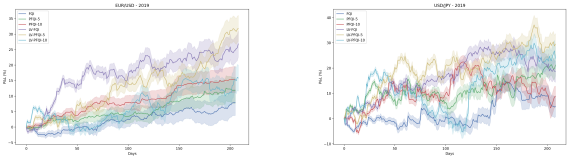
Table 2: Performance for EUR/USD on validation year: 2019 of FQI variations. **SR** stands for Sharpe Ratio, **MDD** for Maximum Drawdown, **P&L** as mean ± standard deviation. The table shows the best three runs after the tuning phase.

5.3.1 Model Selection

We trained our models in different conditions to comprehensively evaluate the LV-FQI algorithm. The experiments were conducted on data from 2019 to 2022. To increase robustness, we used a training set of two consecutive years, 2020 and 2021, to let the agents experience enough market conditions. Then, to select the models that better generalize, we considered the years before and after the training set: the policies were validated in the former, 2019, and tested in the latter, 2022. We trained different FQI models to select the best trading agents, each characterized by an action *persistence* (1, 5 and 10) and a set of hyperparameters with a 1-minute sampling frequency. Due to its outstanding performances in terms of accuracy and high scalability, we chose *XGBoost* [2] as the regression mechanism for the *Q*-function. It is sufficient to tune the *min child weight* to regulate the model complexity. Generally, the higher the threshold is, the simpler the trained model becomes. Also, it is necessary to tune the number of FQI *iterations*. As it grows, the optimized horizon increases, allowing the model to learn longer-term patterns. As mentioned previously, with LV-FQI, we introduced an efficient method for integrating the *latent space* representation of VAE into the batch algorithm. Therefore, we extracted the latent variables from the two years of training and included them as features set for the regressor to improve the learning of the optimal policy. To summarize, after extracting the latent variables, we tuned the XGBoost complexity and FQI iterations for each value of persistence and performed three different runs. The best trading agents were selected on the validation set.

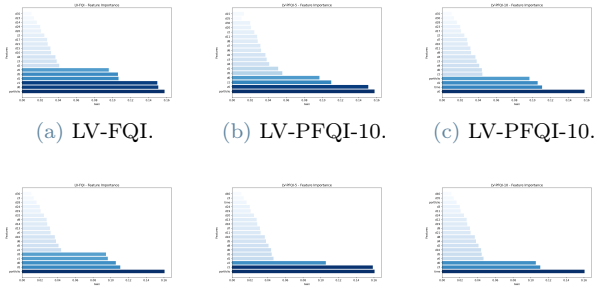
5.3.2 Algorithm Performance

This section focuses on the performance comparison between the Latent-Variable FQI and standard FQI. All the experiments consider a fixed $100K allocation and transaction cost of $\phi = 2 \cdot 10^{-5}$. The results are reported in Table 2 for EUR/USD and Table 3 for USD/JPY. They show the performance in the validation year (2019) for the best three models after the selection phase. Examining the performance metrics, it becomes clear that the *latent variable* variation of FQI is the top performer not only in terms of P&L but also in terms of SR and MDD. Indeed, the proposed method shows, on average, a higher SR than the other algorithms, indicating that it can generate higher returns with lower risk and a lower MDD; hence it can mitigate losses during market downturns. Furthermore, in Fig. 5, the charts show a comparative performance of the best validation run for all the FQI variations and manifest that our proposed approach nearly doubles the P&L so generating high profits. Another insight is that those with persistence equal to 5 and 10 outperform the ones trained with persistence equal to 1. The worse signal-to-noise ratio can explain the poor performances, which affects learning using high frequencies. The significant impact of the latent variables *latent variable* can be noticed in prediction by analyzing the feature importance. They allow to capture hidden patterns and trends that may not be observable through other means. Overall, using latent variables in the LV-FQI algorithm is essential because it enables the algorithm to capture more complex relationships between the input features and the output variable, leading to more performant trading strategies. For example, in financial markets, aspects such as market sentiment or the influence of large institutional investors may not be directly observable but can significantly impact market trends.



(a) EUR/USD. (b) USD/JPY.

Figure 5: Comparison performance of cumulative P&L for FQI variations in Validation Year: 2019.



(a) LV-FQI. (b) LV-PFQI-10. (c) LV-PFQI-10.

(d) LV-FQI. (e) LV-PFQI-10. (f) LV-PFQI-10.

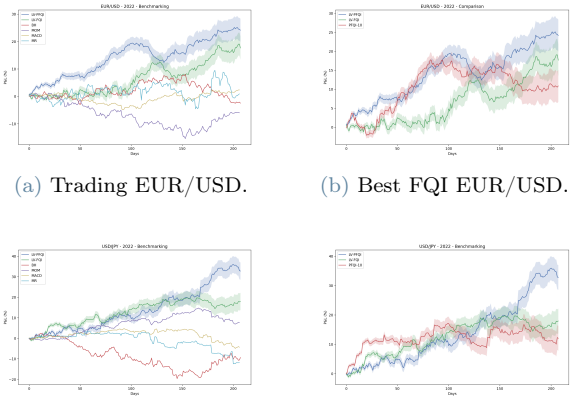
Figure 6: Feature importance EUR/USD (a,b,c) and USD/JPY (d,e,f) in Validation: 2019

Hyperparameters				Performance		
Algorithm	Persistence	MinChildWeight	Iterations	P&L	SR	MDD
FQI	1	20.000	3	-5.0 ± 2.4	-1.74	15.5
		60.000	**4**	**4.9 ± 3.0**	**0.3**	**6.1**
		80.000	1	3.65 ± 2.35	4.6	4.2
PFQI	5	**5000**	**3**	**8.6 ± 1.2**	**0.41**	**10.2**
		20.000	4	7.9 ± 1.2	1.1	2.4
		40.000	2	6.85 ± 3.6	0.60	8.52
PFQI	10	20.000	2	17.5 ± 2.1	1.8	1.4
		40.000	**1**	**21.2 ± 3.5**	**1.17**	**0.76**
		80.000	1	9.4 ± 3.2	0.7	6.2
LV-FQI	1	40.000	3	12.09 ± 2.74	1.15	7.90
		60.000	2	13.5 ± 1.8	1.91	3.1
		80.000	**3**	**18.6 ± 2.4**	**1.14**	**4.7**
LV-PFQI	5	**20.000**	**2**	**29.49 ± 1.8**	**1.62**	**1.67**
		40.000	1	22.2 ± 3.2	2.4	9.2
		80.000	2	20.1 ± 2.9	2.02	3.96
LV-PFQI	10	5000	4	19.95 ± 2.05	1.93	2.66
		40.000	**3**	**23.88 ± 3.1**	**1.014**	**1.27**
		80.000	1	8.2 ± 0.9	0.84	5.2

Table 3: Performance for USD/JPY on validation year: 2019 of FQI variations. **SR** stands for Sharpe Ratio, **MDD** for Maximum Drawdown, **P&L** as mean ± standard deviation. The table shows the best three runs after the tuning phase.

5.3.3 Trading Strategies

This section presents the performance of our Latent-Variable FQI on EUR/USD and USD/JPY against the best standard FQI and different active and passive trading benchmarks in the test year (2022) to assess the effectiveness of our proposed approach. The passive strategy is called *Buy & Hold*. It consists in keeping a constant long portfolio position, instead *active strategies* based on technical indicators. They comprise: *Momentum*: it involves buying securities already showing an upward price trend and selling securities in a downward trend. The idea is to follow the market's momentum and take advantage of the trend; *MACD* is a popular momentum indicator used to identify potential trend reversals. This strategy involves buying when the MACD line crosses above the signal line and selling when it crosses below the signal line, calculated by subtracting the 26-period exponential moving average (EMA) from the 12-period EMA; *Mean-Reverting*: it involves buying securities trading below their historical average price and selling securities trading above their historical average price. The idea is that prices will eventually revert to their mean.



(a) Trading EUR/USD. (b) Best FQI EUR/USD.

(c) Trading USD/JPY. (d) Best FQI USD/JPY.

Figure 7: Comparison performance of cumulative P&L for FQI variations against trading strategies in Test Year: 2022.

The results obtained in this study indicate that the proposed algorithm, which utilizes latent variables with offline RL, is a promising approach for algorithmic trading in the FX market. Specifically, the LV-PFQI model with the persistence of 5 and the LV-FQI model outperformed existing trading strategies, as demonstrated by the significantly higher cumulative P&L achieved in both trading scenarios, as shown in Fig. 7. Moreover, these models consistently performed in the test year, indicating they can generalize well by beating the best PFQI model. These findings suggest that utilizing latent variables can enhance the performance of offline RL algorithms in algorithmic trading, highlighting the potential of this approach for practical applications.

6. Conclusions

The research of effective medium and high-frequency trading strategies is a significant challenge for AI in the FX market, as exchange rates are a prime example of non-stationary and noisy time-series data. It is challenging to learn from them due to their unpredictability, and any attempt to extract patterns using models designed for stationary time series is susceptible to errors. To address this issue, we proposed a novel two-stage RL framework to separate representation learning and task learning by relying on VAE to acquire a latent representation. Then, we train the RL agent by leveraging the features from the latent space with a model-free and offline algorithm, FQI. It is a batch algorithm where the agent can evaluate the effects of its possible portfolio allocations on a historical dataset while observing the rates of the last hour. This raised interesting experimentation around the optimal trading frequency to find trading opportunities by trying different action persistence values. Furthermore, by utilizing latent variables for the underlying structure of the input data, we could improve the trading policy learned in the RL phase. Our proposed algorithm, Latent Variable (Persistent) FQI, obtained robust results on synthetic data generated from classical financial models, where we validated the direct impact of the latent features on the performance. In addition, it obtained consistent results with an attractive performance profile compared to existing baseline RL agents and active and passive trading benchmark strategies in terms of various metrics such as profit and loss, Sharpe ratio, and maximum drawdown for the EUR/USD and USD/JPY currency pairs, among the most liquid ones. This empirically proves two key points: firstly, the advantage of using latent variables in a batch-RL algorithm to have a compact representation of the market non-linearity, hence improving the results of previous work and overcoming the limitations of model-based approaches; secondly, that RL agents can outperform human and algorithmic tradings strategies. To our knowledge, this is the

first attempt to investigate the framework's effectiveness in the FX trading domain. Although the research objectives have been achieved, improvements can still be made to our approach from both practical and theoretical perspectives. Future studies could develop a theoretical framework around the latent approach in RL and test the generality of the two-step method in other domains with high non-stationarity, including mechanisms that can detect regime shifts or handle price forecasting with non-stationary time series. Instead, from an application perspective, the system can be made more realistic by incorporating the bid-ask spread and the order book dynamics to create more complex strategies, including limit and stop-loss orders and partial allocation of the portfolio amount. Additionally, the method can be applied to other asset classes, such as equities and commodities. Finally, other promising approaches in asset trading include hierarchical, multi-objective and robust RL.

References

[1] Lorenzo Bisi, Pierre Liotet, Luca Sabbioni, Gianmarco Reho, Nico Montali, Marcello Restelli, and Cristiana Corno. Foreign exchange trading: A risk-averse batch reinforcement learning approach. In *Proceedings of the First ACM International Conference on AI in Finance (ICAIF '20)*, pages 26:1–26:8, New York, New York, USA, 2020. Association for Computing Machinery, ACM.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[3] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data, 2016.

[4] Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation, 2021.

[5] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

[6] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*, 2014.

[7] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement Learning*, pages 45–73. Springer, 2012.

[8] Alberto Maria Metelli, Flavio Mazzolini, Lorenzo Bisi, Luca Sabbioni, and Marcello Restelli. Control frequency adaptation via action persistence in batch reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[9] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. Wiley & Sons, 2014.

[10] Antonio Riva, Lorenzo Bisi, Pierre Liotet, Luca Sabbioni, Edoardo Vittori, Marco Pinciroli, Michele Trapletti, and Marcello Restelli. Learning fx trading strategies with fqi and persistent actions. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–9, 2021.

[11] Antonio Riva, Lorenzo Bisi, Pierre Liotet, Luca Sabbioni, Edoardo Vittori, Marco Pinciroli, Michele Trapletti, and Marcello Restelli. Addressing non-stationarity in fx trading with online model selection of offline rl experts. In *Proceedings of the Third ACM International Conference on AI in Finance*, ICAIF '22, page 394–402, New York, NY, USA, 2022. Association for Computing Machinery.

[12] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.