Distinguish the indistinguishable: a Deep Reinforcement Learning approach for volatility targeting models

Eric Benhamou
AI Square Connect
eric.benhamou@aisquareconnect.com

David Saltiel
AI Square Connect
david.saltiel@aisquareconnect.com

Serge Tabachnik
Lombard Odier
s.tabachnik@lombardodier.com

Sui Kai Wong
Lombard Odier
sk.wong@lombardodier.com

François Chareyron
Lombard Odier
f.chareyron@lombardodier.com

ABSTRACT

Can an agent efficiently learn to distinguish extremely similar financial models in an environment dominated by noise and regime changes? Standard statistical methods based on averaging or ranking models fail precisely because of regime changes and noisy environments. Additional contextual information in Deep Reinforcement Learning (DRL), helps training an agent distinguish different financial models whose time series are very similar. Our contributions are four-fold: (i) we combine model-based and model-free Reinforcement Learning (RL). The last model-free RL allows us selecting the different models, (ii) we present a concept, called "walk-forward analysis", which is defined by successive training and testing based on expanding periods, to assert the robustness of the resulting agent, (iii) we present a method based on the importance of features that looks like the one in gradient boosting methods and is based on features sensitivities, (iv) last but not least, we introduce the concept of statistical difference significance based on a two-tailed T-test, to highlight the ways in which our models differ from more traditional ones. Our experimental results show that our approach outperforms the benchmarks in almost all evaluation metrics commonly used in financial mathematics, namely net performance, Sharpe ratio, Sortino, maximum drawdown, maximum drawdown over volatility.

KEYWORDS

Deep Reinforcement learning, Model-based, Model-free, Portfolio allocation, Walk forward, Features sensitivity

ACM Reference Format:
Eric Benhamou, David Saltiel, Serge Tabachnik, Sui Kai Wong, and François Chareyron. 2021. Distinguish the indistinguishable: a Deep Reinforcement Learning approach for volatility targeting models. In *MILES Working paper, Dauphine University, LAMSADE*, IFAAMAS, 10 pages.

1 INTRODUCTION

Reinforcement Learning (RL) aims at the automatic acquisition of skills or some other form of intelligence, to behave appropriately and wisely in situations potentially never seen before. When it comes to real world situations, there are two challenges: having a data-efficient learning method and being able to handle complex and unknown dynamical systems that can be difficult to model. Because the dynamic nature of the environment may be challenging to learn, a first stream of RL methods has consisted in modeling the environment with a model called model-based RL. Model-based methods tend to excel in learning complex environments. Examples include, robotics applications, where it is highly desirable to learn using the lowest possible number of real-world trails [34] or in finance where there are a lot of regime changes [26, 30, 47]. A first generation of model-based RL, relying on Gaussian processes and time-varying linear dynamical systems, provides excellent performance in low-data regimes [18–20, 37, 38]. A second generation, leveraging neural network predictive models [**?**] and more recently, deep networks [21, 27, 46], has emerged to leverage the fact that neural networks offer high-capacity function approximators even in domains with high-dimensional observations [24, 35, 49] while retaining some sample efficiency of a model-based approach. Recently, [17] have proposed some solutions for model-based RL to allowing them to achieve asymptotic performance of model-free models. For a full survey of model-based RL model, we refer to [45].

In contrast, the model-free approach aims to learn the optimal actions blindly without a representation of the environment dynamics. Works like [29, 41, 44] have come with the promise that such models learn from raw inputs (and raw pixels) regardless of the game and provide some exciting capacities to handle new situations and environments, though at the cost of data efficiency as they require millions of training runs.

Hence, it is not surprising that the research community has focused on a new generation of models combining model-free and model-based RL approaches. [16] aims at combining model-based and model-free updates for Trajectory-Centric RL. [51] uses temporal difference models to have a model-free deep RL approach for model-based control. [54] answers the question of when to use parametric models in reinforcement learning. Likewise, [32] gives some hints when to trust model-based policy optimization versus model-free. [25] shows how to use model-based value estimation for efficient model-free RL.

All these works, mostly applied to robotics and virtual environments, have not hitherto been widely used for financial time series. Our aim is to be able to distinguish various financial models that can be read or interpreted as model-based RL methods. These models aim at predicting volatility in financial markets in the context of portfolio allocation according to volatility target methods.

These models are quite diverse and encompass statistical models based on historical data such as simple and naive moving average models, multivariate generalized auto-regressive conditional heteroskedasticity (GARCH) models, high-frequency based volatility models (HEAVY) [48] and forward-looking models such as implied volatility or PCA decomposition of implied volatility indices. To be able to decide on an allocation between these various models, we rely on deep model-free RL. However, using just the last prices does not work in our cases as the various volatility models have very similar behaviors. Following [14] and [13], we also add contextual information like macro signals and risk appetite indices to include additional information in our DRL agent hereby allowing us to choose the pre-trained models that are best suited for a given environment.

1.1 Related works

The literature on portfolio allocation in finance using either supervised or reinforcement learning has been attracting more attention recently. Initially, [26, 30, 47] use deep networks to forecast next period prices and to use this prediction to infer portfolio allocations. The challenge of this approach is the weakness of predictions: financial markets are well known to be non-stationary and to present regime changes (see [22, 52, 59]).

More recently, [11, 15, 33, 39, 40, 42, 55–58, 60] have started using deep reinforcement learning to do portfolio allocation. Transaction costs can be easily included in the rules. However, these studies rely on very distinct time series, which is a very different setup from our specific problem. Secondly, they only do one training and test period, and never test for model stability, which is a great concern when the environment presents regime changes. Third, they do not provide any tools that allow the interpretation of, or at least explain the role and the importance of the model's features. Last but not least, they never test the statistical difference between the benchmark and the resulting model.

1.2 Contribution

Our contributions are precisely motivated by the shortcomings presented in the aforementioned remarks. They are four-fold:

- **The use of model-free RL to select various models that can be interpreted as model-based RL**. In a noisy and regime-changing environment like financial time series, the practitioners' approach is to use a model to represent the dynamics of financial markets. We use a model-free approach to learn from states to actions and hence distinguish between these initial models and choose which model-based RL to favor. In order to augment states, we use additional contextual information.
- **The walk-forward procedure.** Because of the non-stationary nature of time-dependent data, and especially financial data, it is crucial to test DRL model stability. We present a new methodology in DRL model evaluation, referred to as walk-forward analysis that iteratively trains and tests models on extending data sets. This can be seen as the analogy of cross-validation for time series. This allows us to validate that the selected hyper-parameters work well over time and that the resulting models are stable over time.
- **Features sensitivity procedure.** Inspired by the concept of feature importance in gradient boosting methods, we have created a feature importance of our deep RL model based on its sensitivity to features inputs. This allows us to rank each feature at each date to provide some explanations why our DRL agent chooses a particular action.
- **A statistical approach to test model stability.** Most RL papers do not address the statistical difference between the obtained actions and predefined baselines or benchmarks. We introduce the concept of statistical difference as we want to validate that the resulting model is statistically different from the baseline results.

2 PROBLEM FORMULATION

Asset allocation is a major question for the asset management industry. It aims at finding the best investment strategy to balance risk versus reward by adjusting the percentage invested in each portfolio asset according to risk tolerance, investment goals and horizons.

Among these strategies, volatility targeting is very common. Volatility targeting forecasts the amount to invest in various assets based on their level of risk to target a constant and specific level of volatility over time. Volatility acts as a proxy for risk. Volatility targeting relies on the empirical evidence that a constant level of volatility delivers some added value in terms of higher returns and lower risk materialized by higher Sharpe ratios and lower drawdowns, compared to a buy and hold strategy [23, 31, 50]. Indeed it can be shown that Sharpe ratio makes a lot of sense for manager to measure their performance. The distribution of Sharpe ratio can be computed explicitly [3]. Sharpe ratio is not an accident and is a good indicator of manager performance [10]. It can also be related to other performance measures like Omega ratio [9] and other performance ratios [8]. It also relies on the fact that past volatility largely predicts future near-term volatility, while past returns do not predict future returns. Hence, volatility is persistent, meaning that high and low volatility regimes tend to be followed by similar high and low volatility regimes. This evidence can be found not only in stocks, but also in bonds, commodities and currencies. Hence, a common model-based RL approach for solving the asset allocation question is to model the dynamics of the future volatility.

To articulate the problem, volatility is defined as the standard deviation of the returns of an asset. Altough it is hard to predict volatility as financial times series are particularly noisy (see [2]), predicting volatility can be done in multiple ways:

- Moving average: this model predicts volatility based on moving averages.
- Level shift: this model is based on a two-step approach that allows the creation of abrupt jumps, another stylized fact of volatility.
- GARCH: a generalized auto-regressive conditional heteroskedasticity model assumes that the return r_t can be modeled by a time series $r_t = \mu + \epsilon_t$ where μ is the expected return and ϵ_t is a zero-mean white noise, and $\epsilon_t = \sigma_t z_t$, where $\sigma_t^2 = \omega + \alpha\epsilon_{t-1}^2 + \beta\sigma_{t-1}^2$. The parameters $(\mu, \omega, \alpha, \beta)$ are estimated simultaneously by maximizing the log-likelihood.

- GJR-GARCH: the Glosten-Jagannathan-Runkle GARCH (GJR-GARCH) model is a variation of the GARCH model (see [28]) with the difference that σ_t, the variance of the white noise ϵ_t, is modelled as: $\sigma_t^2 = \omega + (\alpha + \gamma_{t-1})\epsilon_{t-1}^2 + \beta\sigma_{t-1}^2$ where $I_{t-1} = 1$ if $r_{t-1} < \mu$ and 0 otherwise. The parameters $(\mu, \omega, \alpha, \gamma, \beta)$ are estimated simultaneously by maximizing the log-likelihood.
- HEAVY: the HEAVY model utilizes high-frequency data for the objective of multi-step volatility forecasting [48].
- HAR: this model is an heterogeneous auto-regressive (HAR) model that aims at replicating how information actually flows in financial markets from long-term to short-term investors.
- Adjusted TYVIX: this model uses the TYVIX index to forecast volatility in the bond future market.
- Adjusted Principal Component: this model uses Principal Component Analysis to decompose a set of implied volatility indices into its main eigenvectors and renormalizes the resulting volatility proxy to match a realized volatilty metric.
- RM2006: RM2006 uses a volatility forecast derived from an exponentially weighted moving average (EWMA) metric.

In addition, volatility is the subject of intense modeling in option pricing as presented in [4], [6], [5] or [7], and can even modeled with Levy process see [1].

2.1 Mathematical formulation

We have $n = 9$ models. Each model predicts a volatility for the rolled U.S. 10-year note future contract that we shall call "bond future" in the remainder of this paper. The bond future's daily returns are denoted by r_t^{bond}. These forecasts are then used to compute the allocation to the bond future's models. Mathematically, if the target volatility of the strategy is denoted by σ_{target} and if the model i predicts a bond future's volatility $\sigma_t^{i,pred}$, the allocation in the bond future's model i at time t is given by the ratio between the target volatility and the predicted volatility: $b_t^i = \frac{\sigma_{target}}{\sigma_{t-1}^{i,pred}}$.

Hence, we can compute the daily amounts invested in each of the bond future volatility models and create a corresponding time series of returns $r_t^i = b_t^i \times r_t^{bond}$, consisting of investing in the bond future according to the allocation computed by the volatility targeting model i. This provides n time series of compounded returns whose values are given by $P_t^i = \prod_{u=t_1 \ldots t} \left(1 + r_u^i\right)$. Our RL problem then boils down to selecting the optimal portfolio allocation (with respect to the cumulative reward) in each model-based RL strategies a_t^i such that the portfolio weights sum up to one and are non-negative $\sum_{i=1..n} a_t^i = 1$ and $a_t^i \geq 0$ for any $i = 1..n$. These allocations are precisely the continuous actions of the DRL model. This is not an easy problem as the different volatility forecasts are quite similar. Hence, the n time series of compounded returns look almost the same, making this RL problem non-trivial. Our aim is, in a sense, to distinguish between the indistinguishable strategies that are presented in Figure 1.

Compared to standard portfolio allocation problems, these strategies' returns are highly correlated and similar as presented by the correlation matrix 2, with a lowest correlation of 97%.

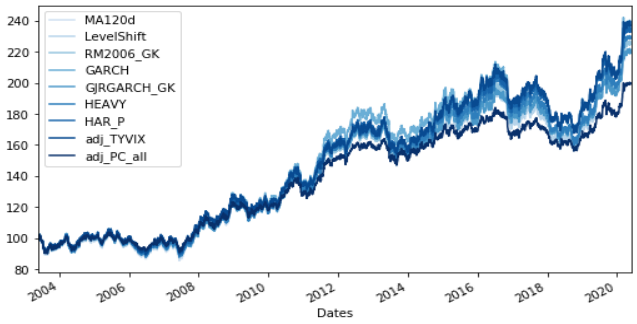

Figure 1: Volatility targeting model price evolution

Following [53], we formulate this RL problem as a Markov Decision Process (MDP) problem. We define our MDP with a 6-tuple $\mathcal{M} = (T, \gamma, \mathcal{S}, \mathcal{A}, P, r)$ where:

- T is the (possibly infinite) decision horizon,
- $\gamma \in]0, 1]$ is the discount factor,
- $\mathcal{S}$ is the state space,
- $\mathcal{A}$ is the action space,
- $p(s_{t+1}|s_t, a_t)$ is the transition probability from the state s_t to s_{t+1} given that the agent has chosen the action a_t,
- $r(s_t, a_t)$ is the reward for a state s_t and an action a_t

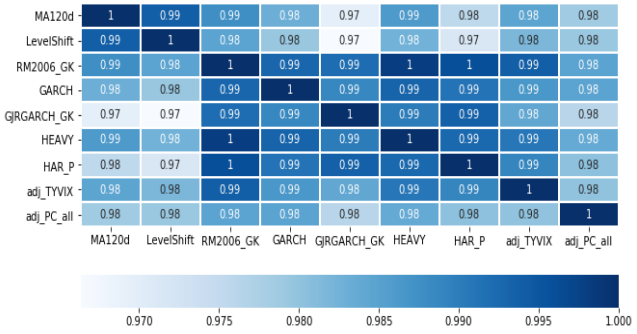

Figure 2: Correlation between the different volatility targeting models' returns

The agent's objective is to maximize its expected cumulative returns, given the start of the distribution. If we denote by π the policy mapping specifying the action to choose in a particular state, $\pi : \mathcal{S} \to \mathcal{A}$, the agent wants to maximize the expected cumulative returns. This is written as: $J^\pi = \mathbb{E}_{s_t \sim P, a_t \sim \pi}\left[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t)\right]$.

MDP assumes that we know all the states of the environment and have all the information to make the optimal decision in every state.

From a practical standpoint, there are a few limitations to accommodate. First of all, the Markov property implies that knowing the current state is sufficient. Hence, we modify the RL setting by taking a pseudo state formed with a set of past observations $(o_{t-n}, o_{t-n-1}, \ldots, o_{t-1}, o_t)$. The trade-off is to take enough past observations to be close to a Markovian status without taking too many observations which would result in noisy states.

In our settings, the actions are continuous and consist in finding at time t the portfolio allocations a_t^i in each volatility targeting model. We denote by $a_t = \left(a_t^1, ..., a_t^n\right)^T$ the portfolio weights vector. Likewise, we denote by $p_t = \left(p_t^1, ..., p_t^n\right)^T$ the closing price vector, and by $u_t = p_t \oslash p_{t-1} = \left(p_t^1/p_{t-1}^1, ..., p_t^n/p_{t-1}^n\right)^T$ the price relative difference vector, where $\oslash$ denotes the element-wise division, and by $r_t = \left(p_t^1/p_{t-1}^1 - 1, ..., p_t^n/p_{t-1}^n - 1\right)^T$ the returns vector which is also the percentage change of each closing prices $p_t^1, ..., p_t^n$. Due to price change in the market, at the end of the same period, the weights evolve according to $w_{t-1} = (u_{t-1} \odot a_{t-1})/(u_{t-1}.a_{t-1})$ where $\odot$ is the element-wise multiplication, and . the scalar product, as shown by figure 3.

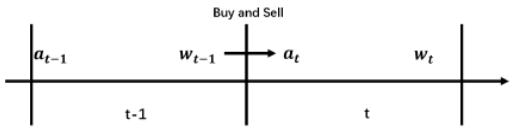

Figure 3: Weights evolution due to cost

The goal of the agent at time t is hence to reallocate the portfolio vector from w^{t-1} to a_t by buying and selling the relevant assets, taking into account the transaction costs that are given by $\alpha|a_t - w_{t-1}|_1$ where α is the percentage cost for a transaction (which is quite low for future markets and given by 1 basis point) and $|.|_1$ is the L_1 norm operator. Hence at the end of time t, the agent receives a portfolio return given by $a_t.u_t - \alpha|a_t - w_{t-1}|_1$. The cumulative reward corresponds to the sum of the logarithmic returns of the portfolio strategy given by $\mathbb{E}\left[\prod_{t=1}^{T} \log\left(a_t.u_t - \alpha|a_t - w_{t-1}|_1\right)\right]$, which is easier to process in a tensor flow graph as a log sum expression and is naturally given by $\mathbb{E}\left[\log\left(\sum_{t=1}^{T} a_t.u_t - \alpha|a_t - w_{t-1}|_1\right)\right]$.

Actions are modeled by a multi-input, multi-layer convolution network whose details are given by Figure 6. As noted in [12], [11] and [15], convolution networks are better for selecting features in DRL for portfolio allocation problem. The goal of the model-free RL method is to find the network parameters. This is done by an adversarial policy gradient method summarized by the algorithm 1 using traditional Adam optimization so that we have the benefit of adaptive gradient descent with root mean square propagation [36] with a learning rate of 5% and a number of iteration steps of 100,000 with an early stop criterion if the cumulative reward does not improve after 15 iteration steps.

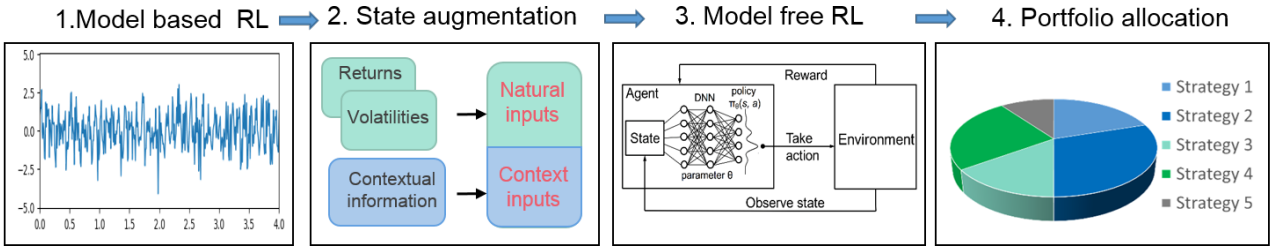

Figure 4: Overall architecture

2.2 Benchmarks

2.2.1 Markowitz. In order to benchmark our DRL approach, we need to compare to traditional financial methods. Markowitz allocation as presented in [43] is a widely-used benchmark in portfolio allocation as it is a straightforward and intuitive mix between performance and risk. In this approach, risk is represented by the variance of the portfolio. Hence, the Markowitz portfolio minimizes variance for a given expected return, which is solved by standard quadratic programming optimization. If we denote by $\mu = (\mu_1, ..., \mu_n)^T$ the expected returns for our n model strategies and by Σ the covariance matrix of these strategies' returns, and by r_{min} the targeted minimum return, the Markowitz optimization problem reads

$$\begin{aligned} &\text{Minimize} && w^T \Sigma w \\ &\text{subject to} && \mu^T w \geq r_{min}, \sum_{i=1...l} w_i = 1, w \geq 0 \end{aligned}$$

2.2.2 Average. Another classical benchmark model for indistinguishable strategies, is the arithmetic average of all the volatility targeting models. This seemingly naive benchmark is indeed performing quite well for indistinguishable models as it mixes diversification and the mean reversion effects of these strategies.

2.2.3 Follow the winner. Another common strategy is to select the best performer of the past year, and use it the subsequent year. If there is some persistence of the models' performance over time, this simple methodology works well. It replicates the standard investor's behavior that tends to select strategies that have performed well in the past. This strategy is referred to as "follow the winner" or simply "the winner" strategy.

2.3 Procedure and walk forward analysis

The whole procedure is summarized by Figure 4. We have n models that represent the dynamics of the market volatility. We then add the volatility and the contextual information to the states, thereby yielding augmented states. The latter procedure is presented as

the second step of the process. We then use a model-free RL approach to find the portfolio allocation among the various volatility targeting models, corresponding to steps 3 and 4. In order to test the robustness of our resulting DRL model, we introduce a new methodology called walk forward analysis.

2.3.1 Walk forward analysis. In machine learning, the standard approach is to do k-fold cross-validation. This approach breaks the chronology of data and potentially uses past data in the test set. Rather, we can take sliding test set and take past data as training data. To ensure some stability, we favor to add incrementally new data in the training set, at each new step.

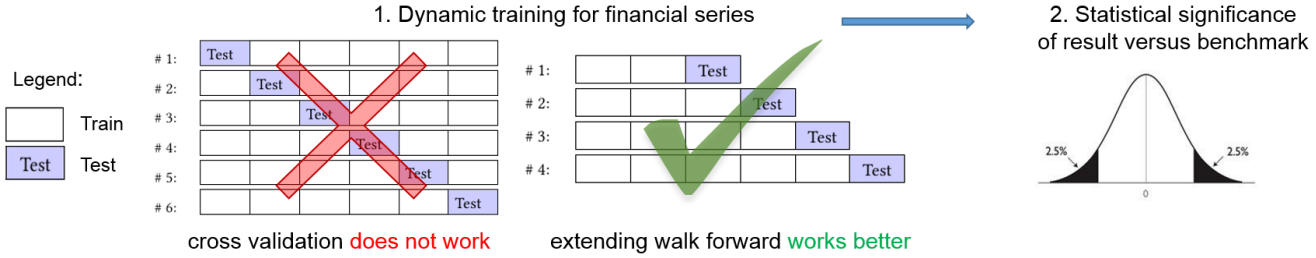

Figure 5: Overall training process

Algorithm 1 Adversarial Policy Gradient

Input: initial policy parameters θ, empty replay buffer $\mathcal{D}$
repeat
  Reset replay buffer
  while not Terminal **do**
    Observe observation o and select action $a = \pi_\theta(o)$ with probability p and random action with probability $1 - p$,
    Execute a in the environment
    Observe next observation o', reward r, and done signal d to indicate whether o' is terminal
    Apply noise to next observation o'
    Store (o, a, o') in replay buffer $\mathcal{D}$
    if Terminal **then**
      for however many updates in $\mathcal{D}$ **do**
        Compute final reward R
      end for
      Update network parameter with Adam gradient ascent $\vec{\theta} \longrightarrow \vec{\theta} + \lambda \nabla_{\vec{\theta}} J_{[0,t]}(\pi_{\vec{\theta}})$
    end if
  end while
until Convergence

This method is sometimes referred to as "anchored walk forward" as we have anchored training data. It is also called "extending walk forward" as we progressively extend the training set. The negative effect of using extending training data set is that it adapts slowly to new information. In our experience, since we have limited data to train our DRL model, we use anchored walk forward to make sure we have enough training data. Last but not least, as the test set is always after the training set, walk forward analysis gives less steps compared with cross-validation. In practice, and for our given data set, we train our models from 2000 to the end of 2013 (giving us at least 14 years of data) and use a repetitive test period of one year from 2014 onward. Once a model has been selected, we also test its statistical significance, defined as the difference between the returns of two time series. We therefore do a T-test to validate how different these time series are. The whole process is summarized by Figure 5.

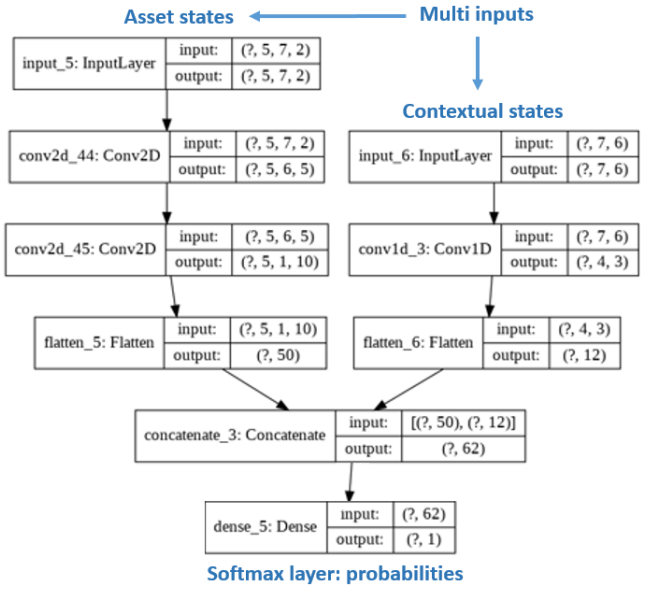

Figure 6: Multi-input DRL network

2.3.2 Model architecture. The states consist in two different types of data: the asset inputs and the contextual inputs.

- Asset inputs are a truncated portion of the time series of financial returns of the volatility targeting models and of the volatility of these returns computed over a period of 20 observations. So if we denote by r_t^i the returns of model i at time t, and by σ_t^i the standard deviation of returns over the

last $d = 20$ periods, asset inputs are given by a 3-D tensor denoted by $A_t = [R_t, V_t]$, with $R_t = \begin{pmatrix} r^1_{t-n} \dots r^1_t \\ \dots \dots \dots \\ r^m_{t-n} \dots r^m_t \end{pmatrix}$, and $V_t = \begin{pmatrix} \sigma^1_{t-n} \dots \sigma^1_t \\ \dots \dots \dots \\ \sigma^m_{t-n} \dots \sigma^m_t \end{pmatrix}$.
This setting with two layers (past returns and past volatilities) is very different from the one presented in Jiang and Liang [33], Liang et al. [40], Zhengyao et al. [60] that uses layers representing open, high, low and close prices, which are not necessarily available for volatility target models. Adding volatility is crucial to detect regime change and is surprisingly absent from these works.

- Contextual inputs are a truncated portion of the time series of additional data that represent contextual information. Contextual information enables our DRL agent to learn the context, and are, in our problem, short-term and long-term risk appetite indices and short-term and long-term macro signals. Additionally, we include the maximum and minimum portfolio strategies return and the maximum portfolio strategies volatility. Similarly to asset inputs, standard deviations is useful to detect regime changes. Contextual observations are stored in a 2D matrix denoted by C_t with stacked past p individual contextual observations. The contextual state reads
$C_t = \begin{pmatrix} c^1_{t-n} \dots c^1_t \\ \dots \dots \dots \\ c^p_{t-n} \dots c^p_t \end{pmatrix}$.

The output of the network is a softmax layer that provides the various allocations. As the dimensions of the assets and the contextual inputs are different, the network is a multi-input network with various convolutional layers and a final softmax dense layer as represented in Figure 6.

2.3.3 Features sensitivity analysis. One of the challenges of neural networks relies in the difficulty to provide explainability about their behaviors. Inspired by computer vision, we present a methodology here that enables us to relate features to action. This concept is based on features sensitivity analysis. Simply speaking, our neural network is a multi-variate function. Its inputs include all our features, strategies, historical performances, standard deviations, contextual information, short-term and long-term macro signals and risk appetite indices. We denote these inputs by X, which lives in $\mathbb{R}^k$ where k is the number of features. Its outputs are the action vector Y, which is an n-d array with elements between 0 and 1. This action vector lives in an image set denoted by $\mathcal{Y}$, which is a subset of $\mathbb{R}^n$. Hence, the neural network is a function $\Phi : \mathbb{R}^k \rightarrow \mathcal{Y}$ with $\Phi(X) = Y$. In order to project the various partial derivatives, we take the L1 norm (denoted by $|.|_1$) of the different partial derivatives as follows: $|\frac{\partial \Phi(X)}{\partial X}|_1$. The choice of the L1 norm is arbitrary but is intuitively motivated by the fact that we want to scale the distance of the gradient linearly.

In order to measure the sensitivity of the outputs, simply speaking, we change the initial feature by its mean value over the last d periods. This is inspired by a "what if" analysis where we would like to measure the impact of changing the feature from its mean value to its current value. In computer vision, the practice is not to use the mean value but rather to switch off the pixel and set it to the black pixel. In our case, using a zero value would not be relevant as this would favor large features. We are really interested here in measuring the sensitivity of our actions when a feature deviates from its mean value as presented in figure 7.

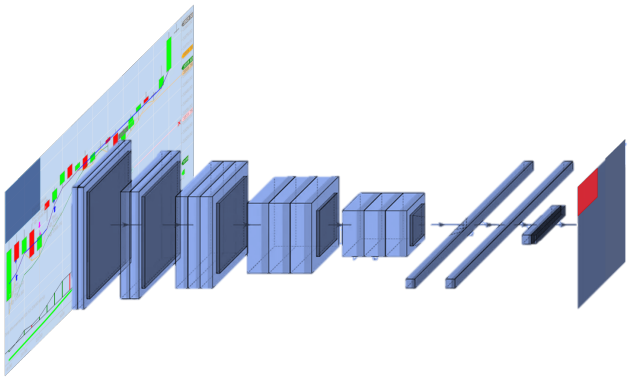

Figure 7: Features sensitivity summary

The resulting value is computed numerically and provides us for each feature a feature importance. We rank these features importance and assign arbitrarily the value 100 to the largest and 0 to the lowest. This provides us with the following features importance plot given below 8. We can notice that the HAR returns and volatility are teh most important features, followed by various returns and volatility for the TYVIX model. Although returns and volatility are dominating among the most important features, macro signals 0d observations comes as the 12th most important feature over 70 features with a very high score of 84.2. The features sensitivity analysis confirms two things: i) it is useful to include volatility features as they are good predictors of regime changes, ii) contextual information plays a role as illustrated by the macro signal.

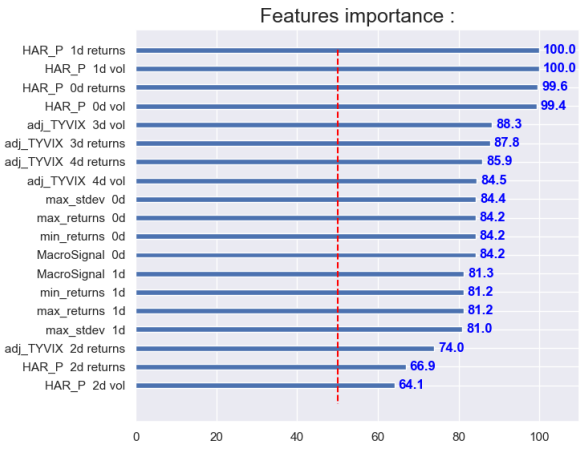

Figure 8: Model explainability

3 OUT OF SAMPLE RESULTS

In this section, we compare the various models: the deep RL model using states with contextual inputs and standard deviation, the deep RL model without contextual inputs and standard deviation, the average strategy, the Markowitz portfolio and the "the winner" strategy. The results are the combination of the 7 distinct test periods: each year from 2014 to 2020. The resulting performance is plotted in Figure 9. We notice that the deepRL model with contextual information and standard deviation is substantially higher than the other models in terms of performance as it ends at 157, whereas other models (the deepRL with no context, the average, the Markowitz and "the winner" model) end at 147.6, 147.8, 145.5, 143.4 respectively. To make such a performance, the DRL model needs to frequently rebalance between the various models (Figure 10) with dominant allocations in GARCH and TYVIX models (Figure 11).

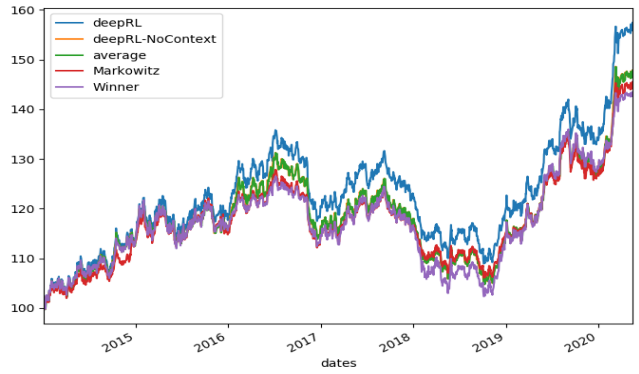

Figure 9: Model comparison

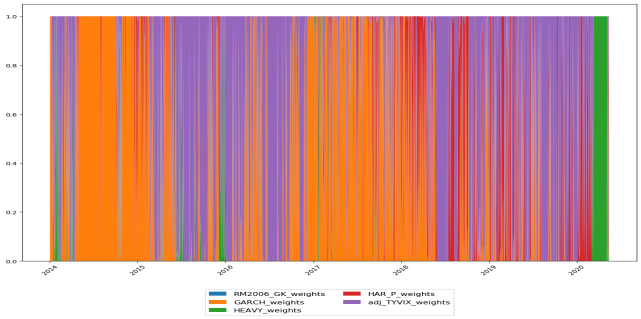

Figure 10: DRL portfolio allocation

3.1 Results description

3.1.1 Risk metrics. We provide various statistics in Table 1 for different time horizons: 1, 3 and 5 years. For each horizon, we put the best model, according to the column's criterion, in bold. The Sharpe and Sortino ratios are computed on daily returns. Maximum drawdown (written as mdd in the table), which is the maximum observed loss from a peak to a trough for a portfolio, is also computed on daily returns. DRL1 is the DRL model with standard deviations and contextual information, while DRL2 is a model with no contextual information and no standard deviation. Overall, DLR1, the DRL model with contextual information and standard deviation, performs better for 1, 3 and 5 years except for three-year maximum drawdown. Globally, it provides a 1% increase in annual net return for a 5-year horizon. It also increases the Sharpe ratio by 0.1 and is able to reduce most of the maximum drawdowns except for the 3-year period. Markowitz portfolio selection and "the winner" strategy, which are both traditional financial methods heavily used by practitioners, do not work that well compared with a naive arithmetic average and furthermore when compared to the DRL model with context and standard deviation inputs. A potential explanation may come from the fact that these volatility targeting strategies are very similar making, the diversification effects non effective.

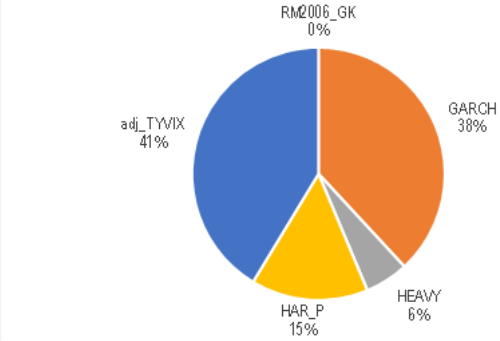

Figure 11: Average model allocation

Table 1: Models comparison over 1, 3, 5 years

	return	sharpe	sortino	mdd	mdd/vol
			1 Year		
DRL1	**22.659**	**2.169**	**2.419**	**- 6.416**	**- 0.614**
DRL2	20.712	2.014	2.167	- 6.584	- 0.640
Average	20.639	2.012	2.166	- 6.560	- 0.639
Markowitz	19.370	1.941	2.077	- 6.819	- 0.683
Winner	17.838	1.910	2.062	- 6.334	- 0.678
			3 Years		
DRL1	**8.056**	**0.835**	**0.899**	- 17.247	**- 1.787**
DRL2	7.308	0.783	0.834	**- 16.912**	- 1.812
Average	7.667	0.822	0.876	- 16.882	- 1.810
Markowitz	7.228	0.828	0.891	- 16.961	- 1.869
Winner	6.776	0.712	0.754	- 17.770	- 1.867
			5 Years		
DRL1	**6.302**	**0.651**	**0.684**	**- 19.794**	**- 2.044**
DRL2	5.220	0.565	0.584	- 20.211	- 2.187
Average	5.339	0.579	0.599	- 20.168	- 2.187
Markowitz	4.947	0.569	0.587	- 19.837	- 2.074
Winner	4.633	0.508	0.526	- 19.818	- 2.095

3.1.2 Statistical significance. Following our methodology described in 5, once we have computed the results for the various walk-forward test periods, we do a T-statistic test to validate the significance of the result. A first and naive T-stat test is to test the statistical significance of the returns difference between the strategies with 0. This is provided in Table 2. Given two models, we test the null hypothesis that the difference of returns between the two models is equal to 0. We provide the T-statistic and, in parenthesis, the p-value. We take a p-value threshold of 5%, and put he cases where we can reject the null hypothesis in bold. Hence, we conclude that the DRL1 model is statistically different from DRL2 and "the winner" models. For the other models, we fail to reject that they are statistically different.

Table 2: T stat and P-values (in parenthesis) for the statistical difference between returns.

Returns	DRL2	Average	Markowitz	Winner
DRL1	**2.7 (0.7%)**	0.2 (85%)	1.5 (14.2%)	**2 (4.9%)**
DRL2		0 (99.8%)	0.4 (71.5%)	0.7 (48.2%)
Average			0.1(95.4%)	0.1 (92.7%)
Markowitz				0.2 (84.1%)

If we do the test on the returns running average (computed as $(\sum_{u=0}^{t} r_u/t)$ for various times t) , the conclusions of the Student test results are very different as shown in table3. For a p-value threshold of 5%, we conclude that all the models are statistically different, except when comparing the DRL2 and Average models or the Average and "the winner" models. For the latter, we fail to reject that they are statistically different. These results on the running average are quite intuitive as we are able to distinguish the curve in Figure 9.

Table 3: T-statistics and P-values (in parenthesis) for running average returns difference

Avg Return	DRL2	Average	Markowitz	Winner
DRL1	**72.1 (0%)**	**14 (0%)**	**44.1 (0%)**	**79.8 (0%)**
DRL2		1.2 (22.3%)	**24.6 (0%)**	**10 (0%)**
Average			**7.6(0%)**	0.9 (38.7%)
Markowitz				**-13.1 (0%)**

3.1.3 Results discussion. It is interesting to understand how the DRL model achieves such a performance as it provides an additional 1% annual return over 5 years, and an increase in Sharpe ratio of 0.10. This is done simply by selecting the right strategies at the right time. We notice that the DRL model selects the GARCH model quite often and, more recently, the HAR and HEAVY model (Figure 10). When targeting a given volatility level, capital weights are inversely proportional to the volatility estimates. Hence, lower volatility estimates give higher weights and in a bullish market give higher returns. Conversely, higher volatility estimates drive capital weights lower and have better performance in a bearish market. The allocation of these models evolve quite a bit as shown by Figure 12, which plots the rank of the first 5 models.

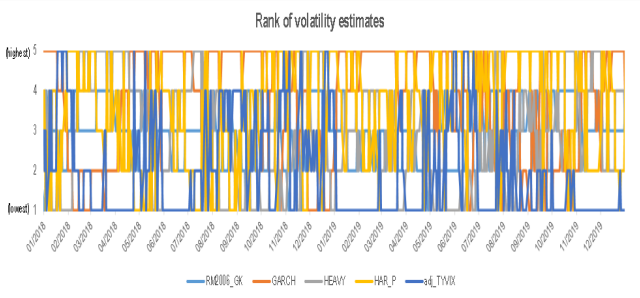

Figure 12: Volatility estimates rank

We can therefore test if the DRL model has a tendency to select volatility targeting models that favor lower volatility estimates. If we plot the occurrence of rank by dominant model for the DRL model, we observe that the DRL model selects the lowest volatility estimate model quite often (38.2% of the time) but also tends to select the highest volatility models giving a U shape to the occurrence of rank Figure as given in 13. This U shape confirms two things: i) the model has a tendency to select either the lowest or highest volatility estimates models, which are known to perform best in bullish markets or bearish markets (however, it does not select these models blindly as it is able to time when to select the lowest or highest volatility estimates); ii) the DRL model is able to reduce maximum drawdowns while increasing net annual returns as seen in Table 1. This capacity to simultaneously increase net annual returns and decrease maximum drawdowns indicates a capacity to detect regime changes. Indeed, a random guess would only increase the leverage when selecting lowest volatility estimates, thus resulting in higher maximum drawdowns.

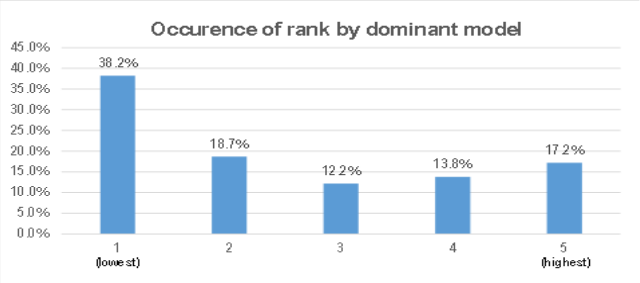

Figure 13: Occurrence of rank for the DRL model

3.2 Benefits of DRL

The advantages of context based DRL are numerous: (i) by design, DRL directly maps market conditions to actions and can thus adapt to regime changes, (ii) DRL can incorporate additional data and be a multi-input method, as opposed to more traditional optimization methods.

3.3 Future work

As nice as this may look, there is room for improvement as more contextual data and architectural networks choices could be tested.

4 CONCLUSION

In this work, we propose a model-free RL approach to select volatility targeting models that can be interpreted as a mix of model-free and model-based RL. The last model-free step enables us to select the allocation in each model. These models aim at describing and simplifying the volatility behavior. Thanks to the addition of volatility and context, we are able to find an efficient allocation among these volatility targeting models. The profitability and ability of this method to reduce risk are verified as we are able to overperform the various benchmarks. The use of successive training and testing sets enables us to stress test the robustness of the resulting agent. Features sensitivity analysis confirms the importance of volatility and contextual variables and explains the DRL agent's better performance. Last but not least, statistical tests validate that results are statistically significant.

REFERENCES

[1] E. Benhamou. 2002. *Option pricing with Levy Process.* Finance 0212006. University Library of Munich, Germany. https://ideas.repec.org/p/wpa/wuwpfi/0212006.html

[2] E. Benhamou. 2018. Trend without hiccups: a Kalman filter approach. *ssrn.1808.03297* (2018). arXiv:1808.03297 [q-fin.TR]

[3] E. Benhamou. 2019. Connecting Sharpe ratio and Student t-statistic, and beyond. *ArXiv* (2019). arXiv:1808.04233 [q-fin.ST]

[4] E. Benhamou and O. Croissant. 2007. Local Time for the SABR Model: Connection with the 'Complex' Black Scholes and Application to CMS and Spread Options.

[5] E. Benhamou, E. Gobet, and M. Miri. 2010. Expansion formulas for European options in a local volatility model. *International Journal of Theoretical and Applied Finance* 13, 4 (June 2010), 603–634.

[6] E. Benhamou, E. Gobet, and M. Miri. 2010. Time Dependent Heston Model. *SIAM J. Financial Math.* 1 (2010), 289–325.

[7] E. Benhamou, E. Gobet, and M. Miri. 2012. Analytical formulas for a local volatility model with stochastic rates. *Quantitative Finance* 12, 2 (2012), 185–198.

[8] E. Benhamou and B. Guez. 2018. Incremental Sharpe and other performance ratios. *Journal of Statistical and Econometric Methods* 2018 (2018).

[9] E. Benhamou, B. Guez, and N. Paris. 2019. Omega and Sharpe ratio. *ArXiv* (2019). arXiv:1911.10254 [q-fin.RM]

[10] E. Benhamou, D. Saltiel, B. Guez, and N. Paris. 2019. Testing Sharpe ratio: luck or skill? *ArXiv* (2019). arXiv:1905.08042 [q-fin.RM]

[11] E. Benhamou, D. Saltiel, J.-J. Ohana, and J. Atif. 2021. Detecting and adapting to crisis pattern with context based Deep Reinforcement Learning. In *International Conference on Pattern Recognition (ICPR).* IEEE Computer Society.

[12] E. Benhamou, D. Saltiel, J. J. Ohana, J. Atif, and R. Laraki. 2021. Deep Reinforcement Learning (DRL) for Portfolio Allocation. In *Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track*, Y. Dong, G. Ifrim, D. Mladenić, C. Saunders, and S. Van Hoecke (Eds.). Springer International Publishing, Cham, 527–531.

[13] E. Benhamou, D. Saltiel, S. Ungari, and R. L. Abhishek Mukhopadhyay, Jamal Atif. 2021. Knowledge discovery with Deep RL for selecting financial hedges. In *AAAI: KDF.* AAAI Press.

[14] E. Benhamou, D. Saltiel, S. Ungari, and A. Mukhopadhyay. 2020. Bridging the gap between Markowitz planning and deep reinforcement learning. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS): PRL.* AAAI Press.

[15] E. Benhamou, D. Saltiel, S. Ungari, and A. Mukhopadhyay. 2020. Time your hedge with Deep Reinforcement Learning. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS): FinPlan.* AAAI Press.

[16] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine. 2017. Combining Model-Based and Model-Free Updates for Trajectory-Centric Reinforcement Learning. In *PMLR (Proceedings of Machine Learning Research, Vol. 70)*, D. Precup and Y. W. Teh (Eds.). PMLR, International Convention Centre, Sydney, Australia, 703–711.

[17] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel. 2018. Model-Based Reinforcement Learning via Meta-Policy Optimization. arXiv:1809.05214 http://arxiv.org/abs/1809.05214

[18] M. Deisenroth, D. Fox, and C. Rasmussen. 2014. Gaussian processes for data-efficient learning in robotics and control.

[19] M. Deisenroth and C. E. Rasmussen. 2011. PILCO: A model-based and data-efficient approach to policy search.

[20] M. P. Deisenroth, C. E. Rasmussen, and D. Fox. 2011. Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning.

[21] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft. 2016. Learning and policy search in stochastic dynamical systems with bayesian neural networks.

[22] J. Dias, J. Vermunt, and S. Ramos. 2015. Clustering financial time series: New insights from an extended hidden Markov model. *European Journal of Operational Research* 243 (06 2015), 852–864.

[23] A. Dreyer and S. Hubrich. 2017. Tail Risk Mitigation with Managed Volatility Strategies.

[24] F. Ebert, C. Finn, S. Dasari, A. Xie, A. X. Lee, and S. Levine. 2018. Visual Foresight: Model-Based Deep Reinforcement Learning for Vision-Based Robotic Control.

[25] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine. 2018. Model-Based Value Estimation for Efficient Model-Free Reinforcement Learning. arXiv:1803.00101 http://arxiv.org/abs/1803.00101

[26] F. Freitas, A. De Souza, and A. Almeida. 2009. Prediction-based portfolio optimization model using neural networks. *Neurocomputing* 72 (06 2009), 2155–2170.

[27] Y. Gal, R. McAllister, and C. E. Rasmussen. 2016. Improving PILCO with Bayesian Neural Network Dynamics Models.

[28] L. R. Glosten, R. Jagannathan, and D. E. Runkle. 1993. On the Relation between the Expected Value and the Volatility of the Nominal Excess Return on Stocks. *Journal of Finance* 48, 5 (1993), 1779–1801.

[29] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor.

[30] J. B. Heaton, N. G. Polson, and J. H. Witte. 2017. Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry* 33, 1 (2017), 3–12.

[31] A. Hocquard, S. Ng, and N. Papageorgiou. 2013. A Constant-Volatility Framework for Managing Tail Risk. *The Journal of Portfolio Management* 39 (2013), 28–40.

[32] M. Janner, J. Fu, M. Zhang, and S. Levine. 2019. When to Trust Your Model: Model-Based Policy Optimization. arXiv:1906.08253 http://arxiv.org/abs/1906.08253

[33] Z. Jiang and J. Liang. 2016. Cryptocurrency Portfolio Management with Deep Reinforcement Learning. arXiv:1612.01277

[34] L. P. Kaelbling, M. L. Littman, and A. P. Moore. 1996. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research* 4 (1996), 237–285.

[35] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowsi, S. Levine, R. Sepassi, G. Tucker, and H. Michalewski. 2019. Model-Based Reinforcement Learning for Atari.

[36] D. Kingma and J. Ba. 2014. Adam: A Method for Stochastic Optimization.

[37] V. Kumar, E. Todorov, and S. Levine. 2016. Optimal control with learned local models: Application to dexterous manipulation. , 378–383 pages.

[38] S. Levine and V. Koltun. 2013. Guided Policy Search. , 9 pages. http://proceedings.mlr.press/v28/levine13.html

[39] X. Li, Y. Li, Y. Zhan, and X.-Y. Liu. 2019. Optimistic Bull or Pessimistic Bear: Adaptive Deep Reinforcement Learning for Stock Portfolio Allocation.

[40] Liang et al. 2018. Adversarial Deep Reinforcement Learning in Portfolio Management. arXiv:1808.09940 [q-fin.PM]

[41] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. 2016. Continuous control with deep reinforcement learning.

[42] Y. Liu, Q. Liu, H. Zhao, Z. Pan, and C. Liu. 2020. Adaptive Quantitative Trading: an Imitative Deep Reinforcement Learning Approach.

[43] H. Markowitz. 1952. Portfolio Selection. *The Journal of Finance* 7, 1 (March 1952), 77–91.

[44] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (02 2015), 529–33.

[45] T. M. Moerland, J. Broekens, and C. M. Jonker. 2020. Model-based Reinforcement Learning: A Survey. arXiv:2006.16712 [cs.LG]

[46] A. Nagabandi, G. Kahn, R. Fearing, and S. Levine. 2018. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. , 7559–7566 pages. https://doi.org/10.1109/ICRA.2018.8463189

[47] S. Niaki and S. Hoseinzade. 2013. Forecasting S&P 500 index using artificial neural networks and design of experiments. *Journal of Industrial Engineering International* 9 (02 2013).

[48] D. Noureldin and N. Shephard. 2012. Multivariate high-frequency-based volatility (HEAVY) models. , 907–933 pages.

[49] J. Oh, X. Guo, H. Lee, R. Lewis, and S. Singh. 2015. Action-Conditional Video Prediction using Deep Networks in Atari Games.

[50] R. Perchet, R. Leote de Carvalho, T. Heckel, and P. Moulin. 2016. Predicting the Success of Volatility Targeting Strategies: Application to Equities and Other Asset Classes. , 21–38 pages.

[51] V. Pong, S. Gu, M. Dalal, and S. Levine. 2018. Temporal Difference Models: Model-Free Deep RL for Model-Based Control. arXiv:1802.09081 http://arxiv.org/abs/1802.09081

[52] K. Salhi, M. Deaconu, A. Lejay, N. Champagnat, and N. Navet. 2015. Regime switching model for financial data: empirical risk analysis.
[53] R. S. Sutton and A. G. Barto. 2018. *Reinforcement Learning: An Introduction*. The MIT Press, The MIT Press.
[54] H. van Hasselt, M. Hessel, and J. Aslanides. 2019. When to use parametric models in reinforcement learning? arXiv:1906.05243 http://arxiv.org/abs/1906.05243
[55] H. Wang and X. Y. Zhou. 2019. Continuous-Time Mean-Variance Portfolio Selection: A Reinforcement Learning Framework. arXiv:1904.11392 [q-fin.PM]
[56] Z. Xiong, X.-Y. Liu, S. Zhong, H. Yang, and A. Walid. 2019. Practical Deep Reinforcement Learning Approach for Stock Trading.
[57] Y. Ye, H. Pei, B. Wang, P.-Y. Chen, Y. Zhu, J. Xiao, and B. Li. 2020. Reinforcement-Learning Based Portfolio Management with Augmented Asset Movement Prediction States. In *AAAI*. AAAI, New York, 1112–1119. https://aaai.org/ojs/index.php/AAAI/article/view/5462
[58] P. Yu, J. S. Lee, I. Kulyatin, Z. Shi, and S. Dasgupta. 2019. Model-based Deep Reinforcement Learning for Financial Portfolio Optimization.
[59] K. Zheng, Y. Li, and W. Xu. 2019. Regime switching model estimation: spectral clustering hidden Markov model.
[60] Zhengyao et al. 2017. Reinforcement Learning Framework for the Financial Portfolio Management Problem. arXiv:1706.10059v2