# Augmented Factor Investing with Reinforcement Learning

Rui Dai Marshall Guan Ziyue Su Shaohan Wang

## Contents

## 1 Introduction

Portfolio optimization has been a main concern in the financial industry for a long time. One of the most popular theory in this area is the Modern Portfolio Theory (MPT) developed by Harry Markowitz (1952), in which the optimal portfolio is selected based on the mean and variance of the asset returns, as well as constraints specified by investors. However, as the financial market demonstrates increasing complexity, the assumptions of MPT became more difficult to held so that its applicability is limited. With the rapid development of machine learning theories, a large number of researches have been done on their application on portfolio optimization, and Reinforcement Learning (RL) has been shown to have outstanding performance. The RL algorithm trains an intelligent agent to take actions according to its environments to maximize rewards, which perfectly fits the settings of portfolio optimization.

In addition, using mean and variance only accounts for the "technical" aspect of investing, but fundamentals are also significant factors when it comes to stocks investing. Therefore, this project combines augmented fundamentals factors such as momentum with both the Markowitz framework and reinforcement learning and performs portfolio optimization on selected stocks. A comparison is given to discuss the performance of the models.

## 2 Data

### 2.1 Stocks

In order to produce clear results of portfolio optimization, our team selected the top 20 stocks from the Standard & Poor 500 index in terms of market capitalization so that they have sufficient liquidity. However, some of the top 20 stocks have missing prices of fundamentals data so that we need to extend to the next place. After careful inspection of available data sources, we decided to implement the models in this project on the following 20 stocks:

The stock prices are acquired from Yahoo Finance and the optimization is performed from 2011 to 2019 to avoid the influence of the COVID-19 pandemic in 2020 which is not the concern of this project.

| Ticker | Company Name | Ticker | Company Name |
|---|---|---|---|
| AAPL | Apple Inc. | CMCSA | Comcast Corporation Class A |
| MSFT | Microsoft Corporation | XOM | Exxon Mobil Corporation |
| AMZN | Amazon.com Inc. | ADBE | Adobe Inc. |
| GOOG | Alphabet Inc. Class C | VZ | Verizon Communications Inc. |
| JNJ | Johnson & Johnson | INTC | Intel Corporation |
| UNH | UnitedHealth Group Incorporated | PFE | Pfizer Inc. |
| V | Visa Inc. Class A | CSCO | Cisco Systems Inc. |
| NVDA | NVIDIA Corporation | NFLX | Netflix Inc. |
| PG | Procter & Gamble Company | KO | Coca-Cola Company |
| MA | Mastercard Incorporated Class A | T | AT&T Inc. |

Table 1: A list of selected stocks

## 2.2 Fundamentals

For further analysis and to construct the factor model, we select six different fundamental variables: return on assets (ROA), return on equity (ROE), earnings per share (EPS), operating profitability (OP), accrual-based operating profitability (A) and cash-based operating profitability (CBOP). The fundamental data is acquired from SimFin. The sample fundamental data time frame is from June 2010 to December 2019. The choice of the fundamental variables is driven by Dashan Huang et al. (2017). These fundamental variables mainly reflect the performance of individual companies and are widely applied by investors for corporate valuation.

- ROA: one of the simplest measures for corporate performance. It is calculated by company's net income over total assets and reflects the company's actual profitability in terms of its assets. This indicator also shows how the company makes profits by utilizing total assets.
- ROE: another common measure for company performance. It is calculated by company's net income over average shareholder's equity and indicates the corporate actual profitability in terms of its shareholder's equity.
- EPS: widely used for measuring how well the company makes profits from each outstanding share of its stock and is calculated by company's net income over total outstanding shares of the common stock.
- OP: obviously, the company's total incomes generated from all its business and products, except the interest and taxes. It is mainly used as an indicator of the corporate's operation performance at the given period.
- A: one company profitability measure and represents the profit generated from the accrual adjustment earnings.
- CBOP: another measure of how well the company could derive profits except from the accounting accruals cash flows.

We have successfully extracted the fundamental data of return on assets, return on equity, earnings per share, and accrual-based operating profitability (also names as "payables & accruals" on SinFin) from SimFin website. To obtain the operating profitability, we applied the formula

$$\text{OP} = \text{Revenue} - \text{Operating expenses}$$

Similarly, cash-based operating profitability is calculated by,

$$\text{CBOP} = \text{OP} - \text{A}$$

# 3 Markowitz Portfolio Optimization with Momentum Factor

## 3.1 Motivation

The modern portfolio theory (MPT), developed by Harry Markowitz (1952), who was awarded the Nobel Prize later because of this pioneered achievement in portfolio selection field, has been widely used in the finance industry for portfolio optimization. As Markowitz (1952) states, the portfolio selection should be mainly based on risk-reward ratio (or so-call mean variance characteristics), which is opposite to simply considering the individual asset's risk-reward characteristics (Mangram, 2013). The basic idea conducted by MPT is that the optimal or so-called "efficient" portfolio should be the one maximizing the expected portfolio returns given a certain level of risk (also known as the volatility of portfolio) by assigning different weight to each selected asset. The key component during portfolio analysis is the trade-off between the expected value and the volatility of the returns data, which is characterized by risk aversion coefficients $\gamma$. One of the benefits of conducting MPT is that the investors could diversify the portfolio risk by preventing a highly concentrated portfolio on a few or single assets.

The input variables for the Markowitz portfolio optimization are the expected values $\mu$ and covariance matrix $\Sigma$ of asset returns. Obviously, the estimation of these two inputs will have significant influence on the results of portfolio optimization. Since a simple estimation by descriptive statistics may fail to capture the underlying evolution of the assets, our team combines a momentum factor model on fundamentals to estimate the expected return of the assets.

Our goal is to determine the optimized portfolio values given the optimal weight for each asset at each specific time point (i.e. each quarter) between the selected time frame 2011 and 2019. First, we fit the momentum factor model trained by the selected assets' fundamental information and their simple net returns. Then, apply the fitted factor model to estimate the asset expected returns at each quarter and treat it as input variable $\mu$. We could also obtain the covariance matrix from the actual returns data quarterly and treat it as another input variable $\Sigma$. In the end, we perform the Markowitz portfolio optimization setup to obtain the optimal portfolio selection with the value adjusted quarterly.

## 3.2 Momentum Factor Model on Fundamentals

### 3.2.1 Momentum Factor

Nowadays, momentum has become a thriving idea in finance industry when investors are going to construct the portfolio by taking the trend of the asset price movement into account. The basic idea conducted by momentum factor or even momentum strategy is to utilize the continuity of the existing market trend and then, the investors are going to enter the corresponding long or short position by assuming this market trend will persist in the same direction. The philosophy is that more money could be made by "buying high and selling higher" than by buying the underpriced stocks and waiting for further market re-evaluation (Barone, 2019). Therefore, on the contrary of traditional fundamental factor models which only considers the performance of the given fundamental variables and the actual returns of the assets, our team plans to add the momentum factor into the model to account for the trend of the fundamental variables' movement in a specified period of time. Inspired by the model introduced by Dashan Huang et al. (2017), the momentum is interpreted as moving averages of fundamentals. Specifically, the momentum (MOM) is calculated by,

$$\mathrm{MOM}_f = (f_t + f_{t-1} + f_{t-2} + f_{t-3})/4$$

where f represents one individual fundamentals variable. In our paper, we choose to calculate the moving average effect by averaging the fundamentals at every four periods (i.e. four quarters).

### 3.2.2 Fundamental Factor Model with Momentum

As discussed in subsection 2.2, we have selected six fundamental variables as our matrix of factor loading matrix B from SimFin website and apply them to train the fundamental factor model: return on assets (ROA), return on equity (ROE), earnings per share (EPS), operating profitability (OP), accruals (A) and cash-based operating profitability (CBOP).

The fundamental factor model with momentum factor explicit formula is given by,

$$\mu = \alpha + \beta_{ROA}\text{MOM}_{ROA} + \beta_{ROE}\text{MOM}_{ROE} + \beta_{EPS}\text{MOM}_{EPS}$$

$$+\beta_{OP}\text{MOM}_{OP} + \beta_{A}\text{MOM}_{A} + \beta_{CBOP}\text{MOM}_{CBOP}$$

where $\mu$ is the expected return of an asset and $\beta$ is the corresponding coefficients of the momentum factors.

We trained the factor model at every time period t separately by cross-sectional regression along with the least square method given in the formula,

$$\min_{f_1(t),\ldots,f_m(t)} \sum_{i=1}^{n} \left( r_i(t) - \sum_{j=1}^{m} \beta_{ij}(t) f_j(t) \right)^2$$

To take momentum factor account into the traditional fundamental factor model, we could simply calculate the fundamental factor loading matrix B's moving average every four quarters at each individual time point (i.e. each quarter) from 2011 to 2019. Then, we treat the loading matrix B's moving average as our regressor X and the actual asset expected returns as our target variable Y into the regression setup. Therefore, at every time period t, we will obtain the optimal fundamental "factor returns" vector $f(t)$. Then, we could simply apply the fitted factor model to estimate the expected returns $\hat{\mu}_i$ of each asset across the time periods. Finally, we will treat all the estimated return vectors as the input variables $\mu$ into the Markowitz portfolio optimization to determine the portfolio value $\mu^T w$ and adjust the portfolio weights quarterly.

## 3.3 Covariance Matrix

Another critical input of the Markowitz portfolio optimization is the covariance matrix $\Sigma$ of the assets as we need to balance the variance of the portfolio $w^T \Sigma w$ while maximizing the expected portfolio returns $\mu^T w$. One problem is that the number of samples needs to exceed the number of assets included in the portfolio to keep the uniqueness property of the covariance matrix. Since the fundamental data are collected quarterly, it is difficult to gather enough samples to derive the invertible covariance matrix if we simply obtain it from the corresponding quarterly net returns. Therefore, we decided to use daily log returns and make projection to the covariance matrix on quarterly horizons accordingly. The assumption of projecting to a lengthy horizon is that the compounded returns $\widehat{r_i(t)}$ are identically distributed and independent of each other across all assets and time periods.

One of the advantages of applying compounded returns is the summation property. We could simply obtain the expected compounded return $\hat{u}_i$ by averaging the summation of individual return $r_i\hat{(t)}$ for each asset.

This method relies on compounded returns instead of arithmetic returns, and the formula of compounded return for asset i is given by

$$\hat{r}_i(t) = \ln \frac{p_i(t)}{p_{i-1}(t)}$$

The expected compounded return for asset i in time horizon 1 to T is then calculated as,

$$\hat{\mu}_i := \frac{1}{T}\sum_{t=1}^{T} \hat{r}_i(t) = \frac{1}{T} \ln \frac{p_i(t)}{p_i(0)}$$

Next, we calculate the covariance of the compounded returns between asset $i$ and $j$

$$\hat{v}_{i,j} := \frac{1}{T-1} \sum_{t=1}^{T} (\hat{r}_i(t) - \hat{\mu}_i)(\hat{r}_j(t) - \hat{\mu}_j)$$

For projection, assume there are 63 trading days in one quarter. Then, the projected expected compounded returns and covariance of the compounded returns are

$$\ell\hat{\mu}_i = \frac{\ell}{T} \sum_{t=1}^{T} \hat{r}_i(t)$$

$$\ell\hat{v}_{i,j} = \frac{\ell}{T-1} \sum_{t=1}^{T} (\hat{r}_i(t) - \hat{\mu}_i)(\hat{r}_j(t) - \hat{\mu}_j)$$

where $\ell = 63$.

The last step is to transform the obtained compounded returns back to simple returns. The expected return of asset $i$ is given by

$$\mu_i = e^{\ell(\hat{\mu}_i + \frac{1}{2}\hat{v}_{i,i})} - 1$$

And the covariance between returns of asset $i$ and $j$ is given by

$$v_{i,j} = e^{\ell(\hat{\mu}_i + \hat{\mu}_j + \frac{1}{2}(\hat{v}_{i,i} + \hat{v}_{j,j}))}(e^{\ell\, \hat{v}_{i,j}} - 1)$$

The expected returns are only used for calculation of covariance matrix here and are not applied in the Markowitz optimization. The whole covariance matrix $\Sigma$ serves as the second input of the Markowitz portfolio optimization problem.

## 3.4 Markowitz Portfolio Optimization

The setting of the Markowitz portfolio optimization problem has several assumptions. First, we would only consider the portfolio with long position where short selling is not available, i.e. the weights $w_i \geq 0$ across each asset. Secondly, the absolute value of the weight for each asset is lower than 0.5 in order to maintain a reasonable level of diversification (i.e. prevent a highly concentrated portfolio). Implicitly, the weights across all assets are summed up to 1.

Hence, the Markowitz optimization setup constrained by the above assumptions is formulated as follows,

$$\begin{aligned}\underset{w}{\text{maximize}} \quad & \mu^T w - \frac{1}{2}\gamma w^T \Sigma w,\ w \geq 0 \\ \text{subject to} \quad & e^T w = 1, \\ & \text{abs}(w) \leq 0.5,\end{aligned}$$

where $\mu$ is the expected returns predicted by the fitted factor model across each individual time period (i.e. each quarter) above, $w$ is the weight of each asset across all time periods to be optimized, $\gamma$ is the risk aversion factor which is usually in range [2,10] and our team set it to be 2 in this paper. $\Sigma$ is the projected and invertible quarterly covariance matrix of the asset returns with the methodology introduced in the subsection 3.3.

In our paper, we apply the python package "cvxpy" to perform the Markowitz optimization. After the optimization, we would obtain the optimal weights for each asset. Then, we could simply derive the optimized portfolio returns $\mu^T w$ and variance $w^T \Sigma w$. Noted, the optimal portfolio is adjusted every quarter between the selected time frame 2011 and 2019.

## 4 Reinforcement Learning (RL)

### 4.1 RL Concepts

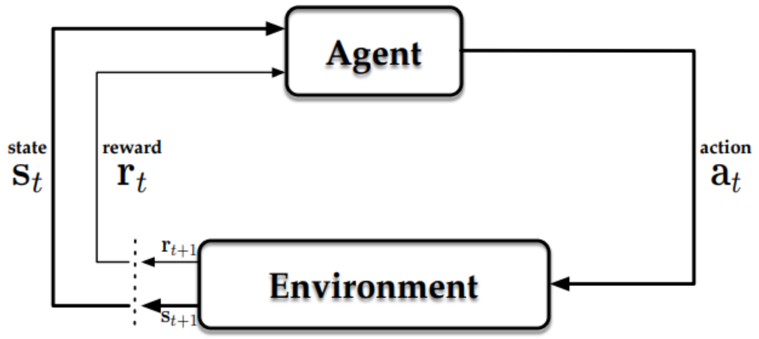

Figure 1: Stochastic dynamical system schematic

In reinforcement learning (RL), an agent (controller) is trained to live in and interact with the environment (system) effectively, as shown in Figure 1. An agent may include at least one of the following components (Silver, 2015):

- Policy: agent's behavior function
- Value function: how good is each state, or state-action pair
- Model: agent's representation of the environment

State space $S$ refers to a set of following possible states that the agents can observe or construct:

- Discrete: $S = s_1, s_2, \dots, s_n$
- Continuous: $S \subseteq R^N$

Action space $A$ refers to a set of following actions that the agent can take:

- Discrete: $A = a_1, a_2, \dots, a_m$
- Continuous: $A \subseteq [c, d]^M$

The agent receives the controlled state $s_t$ and a reward signal $r_t$ associated with the current state. The agent then calculates the action $a_t$ (control signal) and sends it back to the environment. Note that the environment is usually hidden from the agent. The agent can influence the environment and result in different reward signal sequences. In response, the environment transitions to a new state $s_{t+1}$ and the cycle will be repeated (Filos, 2018). In simpler words, the agent takes a state and decides which action to be executed at each iteration; the environment may change according to the behavior of agent. The agent aims to maximize the cumulative reward via perceiving the reward signals to learn the behaviors.

## 4.2 RL Architecture and Training Method for Stock Portfolio

### 4.2.1 Stock Portfolio Environment

The stock portfolio environment is constructed with full information about our selected list of stocks. During training, the environment receives trading decisions from the agent. Based on the trading decisions, which consists of the portfolio weights decision from the agent, the environment calculates the new value of the agent's portfolio. Then, the environment returns the agent with portfolio reward value and information about the market in the next time period.

Specifically, the environment includes:

- Action Space - trading execution:
  The agent sends its portfolio weighting decision as a vector $dv$, that has same dimension as the number of candidate stocks. In addition, the environment always applies a softmax operations on the agent's weighting vector to enforce it sums to 1.

  $$\text{New portfolio weight: } w_i = \frac{e^{dv_i}}{\sum_{s=1}^{20} e^{dv_s}}$$

  With the new portfolio weight $w$, the environment also produces the new portfolio value and returns by calculating the weighted sum of values and returns.

  In order to analyze the agent's decision and training history, all decisions, values, and returns are stored in numerous lists in the environment object.

- State Space - information at each time point:
  The state space includes all the variables that we ask the agent to learn in order to make trading decisions. Since our goal is to explore whether reinforcement learning is

able to improve factor models, the state variables are identical to those in the factor models to guarantee fairness.

Therefore, state information at time $t$ consists of:

1. Fundamental factors: $ROA_{t,s}, ROE_{t,s}, EPS_{t,s}, OP_{t,s}, A_{t,s}, CBOP_{t,s}$ for each stock $s$ at time $t$.
2. Stock return covariances: return covariances between the 20 candidate stocks using a lookback window of 63 days, that is the number of trading days in a quarter.

- Observation Space - environment information exposed to trading agent:
  At each time step $t$, only state information at $t$ is sent to the agent from the environment.

  While the alternative is to send more historic information to the agent, doing so will greatly increase the computational costs of the model. At the same time, using neural network architecture with long term memory, such as Recurrent Neural Network, Long Short Term Memory, and Gated Recurrent Unit, are better implementation that enables the agent to benefit from extended historical information without oversupplying them.

- Reward/Value Function - incentive signal at each step:
  In our model, we use portfolio value as the numerical reward signal for the agent to optimize. Rewarding with portfolio value is straight forward to implement with low computation cost. We can also easily substitute other numerical award signal, such as Sharpe ratio, to account for risk appetite and other investor preferences.

### 4.2.2 Stock Trading Agent and Training Steps

The following algorithm and plot suggest a general approach to build a stock trading agent on a small selection of stock market data. This setup should be classified as a semi-Markov decision process since its outcome is also influenced by other traders.

```
inputs : trading universe of $M$-assets
         initial portfolio vector $w_1 = a_0$
         initial asset prices $p_0 = o_0$
         objective function $\mathcal{J}$
output : optimal agent parameters $\theta_*, \varphi_*$

repeat
    for $t = 1, 2, \dots T$ do
        observe 2-tuple $\langle o_t, r_t \rangle$
        calculate gradients $\nabla_\theta \mathcal{J}(r_t)$ and $\nabla_\varphi \mathcal{J}(r_t)$
        update agent parameters $\theta, \varphi$
            using adaptive gradient optimizers
        get estimate of agent state: $s_t \approx f(\cdot, o_t)$
        sample and take action: $a_t \sim \pi(\cdot|s_t; \varphi)$
          rebalance
    end
until convergence
set $\theta_*, \varphi_* \leftarrow \theta, \varphi$
```

Figure 2: General agent training scheme

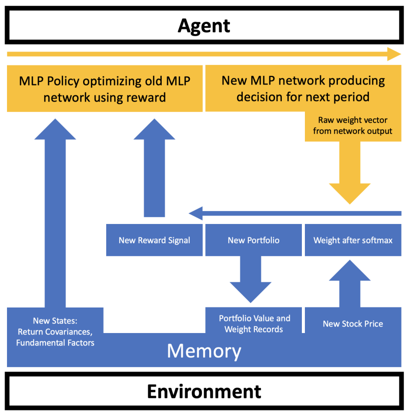
Agent
MLP Policy optimizing old MLP network using reward
New MLP network producing decision for next period
Raw weight vector from network output
New Reward Signal
New Portfolio
Weight after softmax
New States: Return Covariances, Fundamental Factors
Portfolio Value and Weight Records
New Stock Price
Memory
Environment

Figure 3: RL portfolio training pipeline

## 4.3 MLP and Proximal Policy Optimization

### 4.3.1 Multilayer Perceptron Policy

A multilayer perceptron (MLP) network is a class of feedforward artificial neural network (ANN) that utilizes a supervised learning technique called backpropagation for training. Backpropagation computes the gradient in weight space of an ANN, with respect to a loss function[1]. To achieve the goal discussed in subsection 4.1, we make use of an MLP as our policy network. The network takes state vectors as inputs and returns discrete probability distribution over the possible actions. Our MLP consists of four layers:

- An input layer that receives the signal
- Two hidden layers of size 64 that serve as the computational engines
- An output layer that makes a decision or prediction about the input

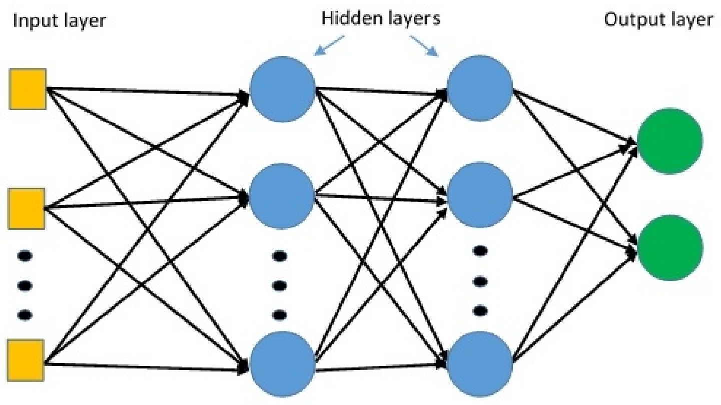

Figure 4: MLP neural network schematic

In machine learning, a perceptron produces a single output based on several real-valued inputs by forming a linear combination using its input weights (A.I. Wiki[2]). We can write it in the following mathematical way:

$$y = \phi(\sum_{i=1}^{n} w_i x_i + b) = \phi(w^T x + b)$$

.

The weights are adjusted so that the error is minimized in the entire output, given by

$$\varepsilon(n) = \frac{1}{2} \sum_j e_j^2(n)$$

Using gradient descent, the change in each weight is

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial v_j(n)} y_i(n)$$

[1]https://en.wikipedia.org/wiki/Backpropagation
[2]https://wiki.pathmind.com/multilayer-perceptron

### 4.3.2 Proximal Policy Optimization

We use Proximal Policy Optimization (PPO) since it enables multiple epochs of minibatch updates. It alternates between sampling data through interaction with the environment, and optimizing a "surrogate" objective function using stochastic gradient ascent (Schulman, 2017). In Trust Region Policy Optimization (TRPO), the "surrogate" objective is maximized in the following way:

$$\underset{\theta}{\text{maximize}} \quad \hat{E}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}\hat{A}_t - \beta KL[\pi_{\theta old}(\cdot|s_t), \pi_\theta(\cdot|s_t)]]$$

PPO restricts the policy update at each training step and simplifies the objective of TRPO by introducing a clipping term to the objective function[3], as follows:

$$L^{CLIP}(\theta) = \hat{E}_t[min(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}\hat{A}_t, clip(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}, 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

We use PPO because it is stable, reliable, tune, easier to implement, and have better overall performance. Besides, it outperforms other online policy gradient methods, and overall strikes a favorable balance between sample complexity, simplicity, and wall-time (Schulman, 2017).

## 4.4 Model Designs

With the settings above, we produce two Reinforcement Learning models:

1. Reinforcement Learning Factor Model:
   In this model, the agent makes decision based on both fundamental factors and return covariances. Its performance should be compared against that of the factor momentum model.

2. Reinforcement Learning Free Model:
   In this model, the agent makes decision only based on return covariances. We could consider it as a benchmark reinforcement learning model that trades with technical information.

[3] http://finrl.org/

## 5 Results

### 5.1 Factor Model with Momentum Against Equal Weight Benchmark

The final optimal portfolio is quarterly adjusted and as showed in Figure 5, the optimal portfolio's performance is fairly good as the expectation. Besides, as showed in Figure 6, compared with the equal weight benchmark portfolio, the risk of the optimal portfolio is also relatively lower at most time points.

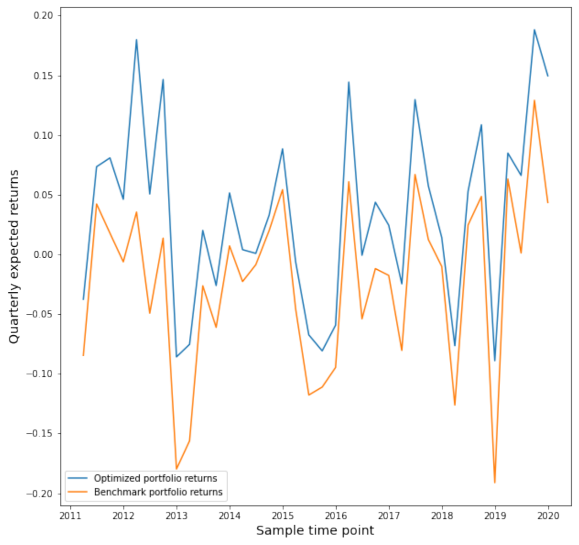

Figure 5: Markowitz Portfolio Return

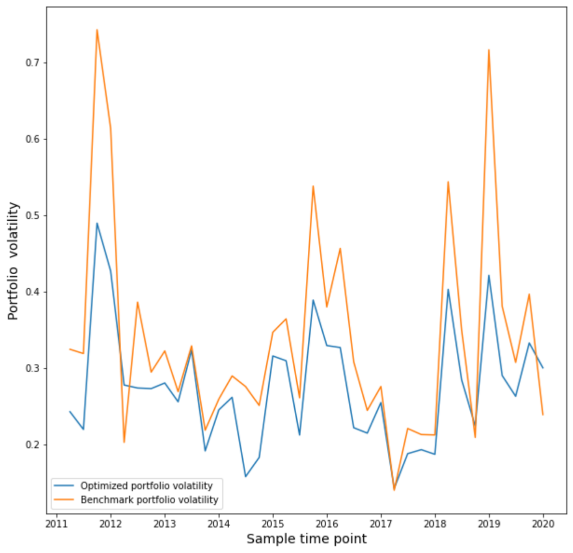

Figure 6: Markowitz Portfolio Volatility

### 5.2 Goodness of Optimization Across Training Period (2011 Q1 - 2018 Q4)

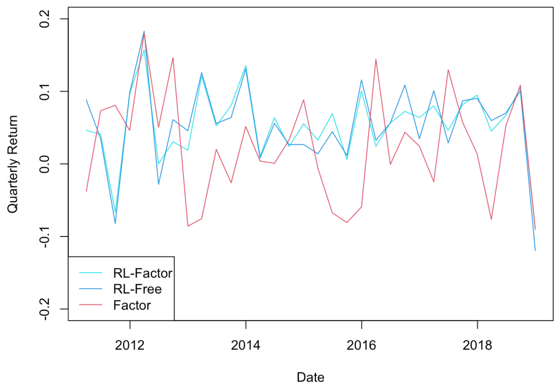

Figure 7: Quarterly returns of different models

The first step of our results analysis is to learn how are the trained reinforcement models' fits over the training period from 2011 Q1 to 2018 Q4. From the quarterly return plot (Figure 7), we see that both reinforcement learning models are able to generate higher and more stable returns than the factor model.

From the numerical results, we notice the reinforcement learning models have higher Sharpe Ratio than factor model (with annual risk free rate = 2%). The excess return appears to be higher and the volatility appears to be lower. Also notice that the excess returns of the reinforcement learning models are much higher than that of the factor model. We believe the over-performance is attributed to the fact that we directly uses portfolio value as reward signal while training the reinforcement learning models.

Between the two reinforcement learning models, we notice that the addition of company fundamental factors could help reduce the volatility of the fitted models and lead to major improvements to risk adjusted return.

| Models | Sharpe Ratio | Excess Return | Volatility |
|---|---|---|---|
| Factor reinforcement learning | 1.919 | 0.195 | 0.101 |
| Free reinforcement learning | 1.653 | 0.196 | 0.118 |
| Factor model with momentum | 0.475 | 0.070 | 0.147 |

Table 2: Model statistics (annualized)

In conclusion, we see the complex architecture of reinforcement learning agent allows the models to capture more information about the stocks, and therefore produce better investment decision during the training period.

## 5.3 Backtesting Period (2019)

To study the out-of-sample performance of our reinforcement learning factor model, we test the model returns over year 2019 with the factor model as a benchmark.

We constructs the backtesting results by collecting the portfolio weighting decision each model made during 2019. We record the daily return based on the weighting an rebalance the weights at the end of each quarter, when new fundamental factor becomes available. The key statistics from the backtesting period are displayed as follows:

| Measurement | RL Factor | RL Free | Factor Model |
|---|---|---|---|
| Annual return | 35.58% | 33.83% | 28.22% |
| Annual volatility | 13.97% | 14.28% | 13.28% |
| Sharpe ratio | 2.25 | 2.11 | 1.94 |
| Max drawdown | -8.53% | -9.25% | -5.88% |
| Sortino ratio | 3.4 | 3.15 | 2.89 |
| Skew | -0.28 | -0.33 | -0.49 |
| Kurtosis | 2.68 | 2.67 | 3.42 |

Table 3: Backtesting period statistics (annualized)

### 5.3.1 Return Analysis

From the return plot, we see that the reinforcement learning model consistently outperforms the factor model over 2019. At most months, the weight assigned by reinforcement learning generates more return than factor model.

Since both models have similar volatility (RL-Factor: 13.97%, Factor: 13.28%), the exceptional return of the reinforcement learning model becomes the primary contributor to its higher Sharpe Ratio. In addition, the reinforcement learning model is less left-skewed than the factor model. We believe this result suggests that the direct portfolio value optimization approach in the reinforcement learning model extends its advantage over the Markowitz optimization in the factor model to the out-of-sample period.

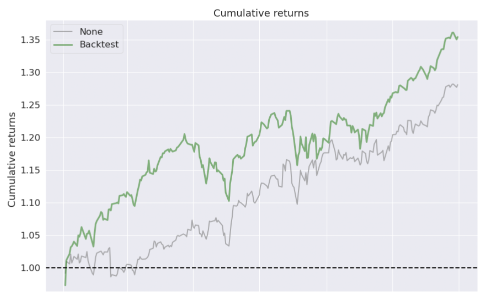

Figure 8: Cumulative return comparison

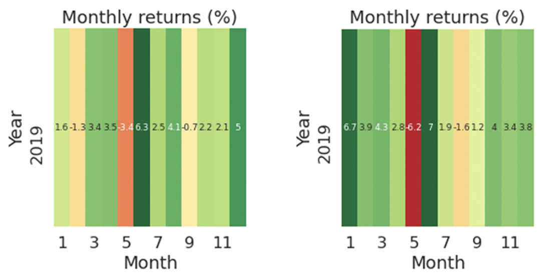

Figure 9: Monthly return comparison

### 5.3.2 Drawdown and Negativity Analysis

Another important aspect about our portfolio is our model's performance and resilience to stress. By studying the drawdowns and underwater periods of the factor model and reinforcement model, we notice that, despite the factor model's maximum drawdown is less than that of the reinforcement learning model, its drawdown are more frequent and volatile.

The finding is also supported by the Sortino ratios of both models (RL-factor: 3.4, Factor: 2.89). Based on these measurements, we see that reinforcement learning model is a safer option during 2019 when stocks are under stress. Sortino ratio is given by

$$\text{Sortino Ratio} = (r - r_f)/\text{Volatility of Downside}$$

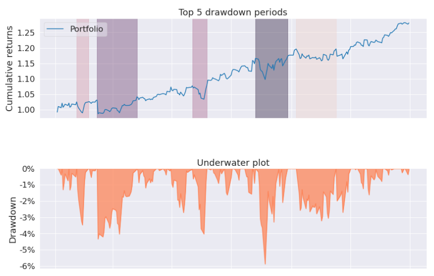

Figure 10: Factor model drawdown

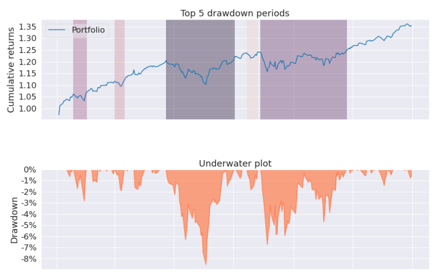

Figure 11: RL Factor model drawdown

## 6 Conclusion

In this paper, we have conducted two theories to perform the portfolio optimization based on 20 selected stock returns.

For Markowitz Portfolio Optimization part, we combine the fundamental factor investing along with the “traditional” modern portfolio theory to perform the portfolio selection. In order to utilize the trend of the continuity of fundamental variables’ movement, we add momentum factor into the fundamental model. In Markowitz optimization setup, we add several assumptions to constrain the portfolio selection to make sure the optimal weights across assets are reasonable enough to prevent a highly concentrated portfolio, which may lead to a bankrupt. The optimal portfolio outperforms an equal weight benchmark portfolio both in terms of volatility and returns.

For Reinforcement Learning part, we keep the information carried in company’s fundamental factors. By directly optimizing numerical reward signals, the reinforcement learning model is able to capture non-linear and hard-to-identify relationships among returns, factors, and return distributions. The Reinforcement generates high returns without sacrificing risk and volatility.

By examining the model’s out-of-sample performance, we find that reinforcement learning model achieves consistent high return and great downturn performance. The daily-measurements of our backtesting simulates public critics in the actual market where real-time performances matter greatly to investors. That makes reinforcement learning model a great tool for investment professionals managing popular products, such as alpha-generating index or exchange-traded funds.

## 7 Limits and Future Work

### Shock

We concentrated our study over the period between 2010-2019, one of the longest economic expansion period. We notice that the portfolio optimization might be out of control under the extreme cases (ex. 2020 stock market crash) and we might not obtain the desired optimized results as expected under these situations.

### Trading Timing

Our models rely heavily on timely access of company’s new fundamental data, which are released throughout an extended period. To apply our models in real-world portfolio managements, we need to develop features that account for the lags between company’s earning announcements.

### Interpretability

Even with superior performance, the reinforcement learning model is a black box that lacks explanatory power, which is a major drawback in finance. In the future, we could try to use techniques such as PCA, transfer learning, NLP, and sensitivity analysis to interpret the models.

## Mid and Small Capitalization Stocks

In this report, we focus our effort on some of the largest companies listed on S&P 500. It would be very meaningful if we can study how would our model performs on less-known stocks, which are more risky but produce more excess returns.

## Multiple Asset Classes

In future, our team might plan to perform the portfolio optimization on more types of assets combined with the stocks, such as commodity derivatives securities or fixed income securities. In higher-dimension, we could also include more fundamental factors into the factor model to account for more accurate estimation combined with principal component analysis method. Besides, while training the factor model by regression method, we may also consider other techniques such as generalized least squares, which may better interpret the residual effect under regression setting. Moreover, the reinforcement learning model is a great tool to absorb high-dimensional data and extract complex relationships.

## Complex Numerical Reward and Wider Constraints

We can test a wide a range of other numerical reward signals. In addition, we can further improve our models by allowing short-selling or borrowing. The expansion of our model settings could allow the models to adapt for investors with different preferences.

## Factor Engineering

Besides using reinforcement learning to produce returns based on factors, we can use the model to create new factors. Such factors could not only help investor create portfolio, but also help company managements to understand and control operating performance better.

## References

1. Barone, Adam. (2019). *Introduction to Momentum Trading.* https://www.marottaonmoney.com/wp-content/uploads/2020/05/Introduction-to-Momentum-Trading.pdf

2. Filos, Angelos. (2018). *Reinforcement Learning for Portfolio Management.*

3. Harry Markowitz. (1952). *Portfolio Selection.* The Journal of Finance. https://cowles.yale.edu/sites/default/files/files/pub/mon/m16-all.pdf

4. Huang, Dashan and Zhang, Huacheng and Zhou, Guofu and Zhu, Yingzi (2017)., *Twin Momentum: Fundamental Trends Matter.* Available at SSRN 2894068.

5. Liu, Xiao-Yang and Yang, Hongyang and Chen, Qian and Zhang, Runjia and Yang, Liuqing and Xiao, Bowen and Wang, Christina Dan. (2020). *FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance.* https://arxiv.org/pdf/2011.09607.pdf

6. Mangram, Myles. (2013). *A Simplified Perspective of the Markowitz Portfolio Theory.* Global Journal of Business Research. 7. https://www.researchgate.net/publication/256034486_A_Simplified_Perspective_of_the_Markowitz_Portfolio_Theory

7. Schulman, John. (2017). *Proximal Policy Optimization Algorithms.* https://arxiv.org/pdf/1707.06347.pdf

8. Silver, David. (2015). *Introduction to Reinforcement Learning.* http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/intro_RL.pdf.