Capstone Project Report

Team ϵ -greedy-quants

Michael Lee, Nikat Patel, Jose Antonio Alatorre Sanchez

January 3, 2021

Abstract

In this capstone project, we explore the usages of Reinforcement Learning for constructing a financial portfolio and for the enhancement of quantitative investment strategies (QIS). In particular, we explore the usage of Policy Gradient Methods (PGM) due to their ability to handle continuous action spaces. PGM are used to create a model-free agent that selects portfolio weights for a portfolio of QIS. The PGM which our group explored are: REINFORCE, REINFORCE with Baseline, Actor-Critic, and Actor-Critic with Eligibility Traces.

Contents

1 Introduction

Trading quantitative finance strategies are based on financial fundamentals using a mathematical model: $\boldsymbol{F(I) \rightarrow R}$. $\boldsymbol{F}$ is defined as a function that maps information, $\boldsymbol{I}$, into $\boldsymbol{R}$. $\boldsymbol{R}$ is usually is a price forecast which can also be a trading signal *(i.e., a buy or a sell indicator)*. Once the mathematical model produces $\boldsymbol{R}$, the quantitative investment strategy needs to transform $\boldsymbol{R}$ into an action. From a portfolio management perspective, this is known as an allocation of capital. Taking recommendations and price forecasts and converting them into actions requires an additional layer of logic. If this layer of logic is executed manually, then the machine learning model is not considered to be automated and therefore is not extensible or adaptable. If we were to automate this layer by machine learning, then we would be able to factor in transaction costs, overlapping transaction against alternative strategies, controlling for draw-downs, sharing signal information between strategies, etc. Therefore, the allocation between quantitative strategies is a problem of interest to financial investors who leverages quantitative strategies for portfolio management. In this capstone project, we plan to address this problem by building a machine learning algorithm for portfolio allocation using reinforcement learning.

Reinforcement learning (RL) is an area of machine learning in which learning of an agent occurs as a result of its own actions and interaction with the environment; the agent has to both exploit its current knowledge and explore new knowledge in order to get a better reward. Every action that the agent takes on every state can be described as **policy** $\boldsymbol{\pi(a|s) = P(A_t = a|s)}$. Most of the Reinforcement Learning algorithms can be classified in two categories:

1. Methods that try to find the optimal policy $\boldsymbol{\pi(a|s)}$ by estimating the value of each action at each state i.e. $\boldsymbol{Q(a, s)}$. These are known as Action-Value methods.

2. Methods that try to find the optimal policy $\boldsymbol{\pi(a|s)}$ by directly estimating it without necessary estimating the state-value action $\boldsymbol{Q(a|s)}$, these methods are known as Policy Gradient Methods (PGM). One of the main advantages of using Policy Gradient Methods is that they can work with environments with a lot of actions, or as an extreme case, with a continuous action space.

Applying the concepts of reinforcement learning to portfolio management, the portfolio manager is considered the agent. The actions are all the possible allocations between strategies, the environment is the stock market, and the cumulative reward can be customized to the investor's goals. For these reasons, we will focus in this project on using Policy Gradient Methods. Development of reinforcement learning models would allow quantitative investors to improve

their portfolio allocation between different quantitative strategies. Applying artificial intelligence to portfolio management is gaining importance as seen in papers by Moody and Saffell (2001), Dempster and Leemans (2006), Cumming (2015), Deng (2017), Jiang, Xu and Liang (2017), and Honchar (2020). The motivation and primary goal of our project is to use Reinforcement Learning Policy Gradient Methods to build a portfolio of quantitative investment strategies.

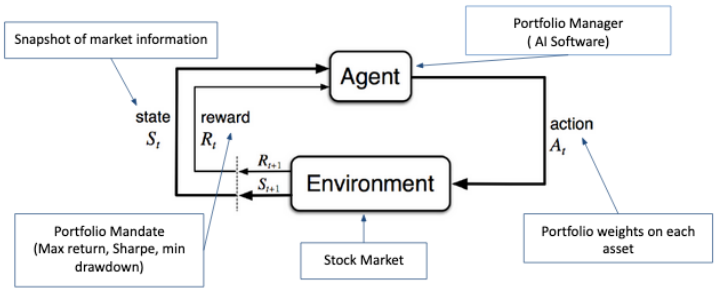

Figure 1: Reinforcement Learning from the perspective of a portfolio manager

For this project, we will narrow portfolio construction of quantitative strategies to **Smart Beta** strategies, one of the most popular quantitative strategies available to financial investors[1]. Smart Beta emphasizes capturing investment factors or market inefficiencies in a rules-based and transparent way. However, there are two main problems in building a portfolio based on Smart Beta strategies. The first problem is that Smart Beta returns are hard to forecast and the second problem is that underlying assets of the strategies can change over time. Since Smart Beta captures a statistical risk premium over time, there is no need to time the strategies and allocations that only focus on risk factors. Therefore, the theoretical solution to this problem is to build an equal risk contribution portfolio.

The reality is that Smart Beta strategies represent a correlation with other risk factors, which in many cases are not taken into account when the financial strategy is constructed. Therefore, having a "full" machine learning algorithm for portfolio construction of Smart Beta strategies will bring an immediate benefit to the investor community by providing a better allocation between Smart Beta strategies under different market scenarios. For the purpose of this project, a "full" machine learning algorithm combines the trading signals with the portfolio allocation.

[1]As of 2018, 91 percent of asset owners have a Smart Beta strategy in their portfolios. Source: https://www.investopedia.com/news/survey-confirms-smart-beta-growth-trajectory/

2 Data Selection and Processing

2.1 Overview

For the project, we will utilize two different sets of data: A control dataset and a real world dataset. The usage of the control dataset is to measure the efficiency of the frameworks in finding optimal actions to a known solution. The real-world dataset will be used to train the reinforcement learning models.

2.2 Control Dataset

For the control dataset, we simulated different assets using a classical geometric Brownian motion process for each of the assets i.e.

$$\boldsymbol{dS_t = \mu S_t dt + \sqrt{\sigma} S_t dB_t}$$

where $\boldsymbol{B_t}$ is a Brownian motion.

2.2.1 Control Data Set Generation

To Generate different simulations of assets we build a class method to each of the environments called

```
build_environment_from_simulated_assets
```

The description of the method can be found in the docstrings.

2.3 Real-World Dataset

We divided the real-world data set in two sub-sets[2]. One set will be use to select features and allocate weights in the portfolio and the other one will be used only to choose features. The portfolio set contains the following ETFs:[3]

- iShares MSCI USA Value Factor ETF **(VLUE)**
- iShares MSCI USA Quality Factor **(QUAL)**
- iShares MSCI USA Momentum Factor **(MTUM)**
- iShares US Size Factor **(SIZE)**
- iShares SP500 Minimum Volatility ETF **(USMV)**
- iShares MSCI EM Minimum Volatility **(EEMV)**
- iShares MSCI EAFE Minimum Volatility **(EFAV)**

[2]Data provided by barchart.com
[3]https://www.ishares.com/us/strategies/smart-beta-investing

We choose this set of ETFs to have a well known, investable set of simple systematic strategies. This set contains all the factor strategies (value, size, momentum, volatility) from iShares [4] with more than 5 years of daily data. The second subset contains the assets that we will use only to extract features:

- iShares 7-10 years US Treasuries **(IEF)**
- iShares Gold Trust **(IAU)**
- SPDR SP 500 ETF Trust **(SPY)**
- DB US Dollar Index **(UUP)**
- Investment Grade Corp Bond iShares Iboxx **(LQD)**

2.3.1 Real-World Data Input to Model

In terms of data requirements for our model, the assets used for our model should consist of daily time bars of closing price and all assets should have the same start and end time range. This data should saved as a separate parquet file for each asset and stored in the `data_env` directory in our Github repository. When preparing our data, the time series are de-meaned. We built a class method called

`build_environment_from_dirs_and_transform`

to load the asset price data from the `data_env` directory into our environment and persist data transformations for further training. Please refer to our User Guide for more details of the data requirements and an example of how the real-world data is loaded into our models.

2.3.2 Processing Data As Features

Once the real-world data is loaded into our models, we process the data before added them as features to our model. Since time series data are nonstationary, we process the data so that they become stationary. First, we detrend each time series using a Double Exponential Smoothing model (this is equivalent to a Holt-Winters smoothing model without the seasonal component) as seen in Figure 2 (Ng, 2019). The trend and level coefficients of the double exponential smoothing model are scaled and added as features. From the detrended time series, we then compute the residuals and demean the returns add the scaled data as features to our model as well. The QQ Plot in Figure 3 shows that although the residuals of a detrended MTUM ETF time series has a left skew, the majority of the residuals in the [-1, 1] theoretical quantile range are normally distributed. Likewise, the QQ Plots of the other ETFs in the portfolio (please refer to Appendix B, Section 8.1) also show normally distributed residuals in the [-1, 1] theoretical quantile range. This gives us confidence that our model can properly use the real-world time series data.

[4] https://www.ishares.com/us/strategies/smart-beta-investing

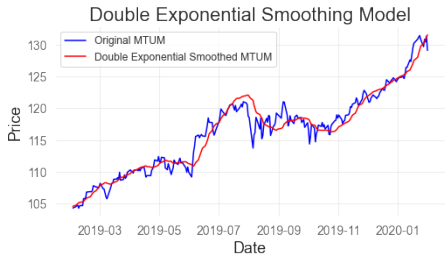

Figure 2: Double Exponential Smoothing of MTUM time series

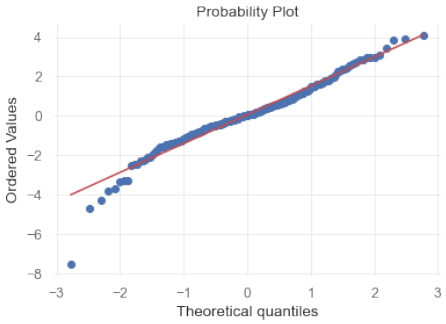

Figure 3: QQ Plot of MTUM detrended residuals

We also plotted an ACF (autocorrelation function) and PACF (partial autocorrelation function) plot of the residuals of the detrended MTUM time series, as seen in Figures 4 and 5 respectively. The ACF plot shows the correlation of a time series with all of its lagged values while the PACF plot shows the correlation of the residuals with its next lag. If there are any high correlations in the PACF plot, it suggests that there could be hidden information in the residual that can be modeled by the following lag and we should keep that lag as a feature.[5] Based on the PACF plot in Figure 5 as well as the PACF plots of the other ETFs in Appendix B, Section 8.1, there is only a significant spike at the lag-0 and lag-1 autocorrelation positions (some ETFs may show a statically significant autocorrelation at other lags but only lag-0 and lag-1 are statistically significant across all 7 ETFs). As a result, we only include one lag to the features in our model.

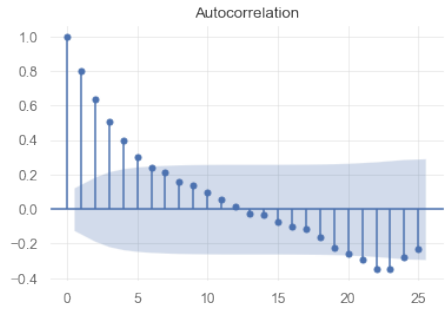

Figure 4: ACF Plot of detrended MTUM residuals

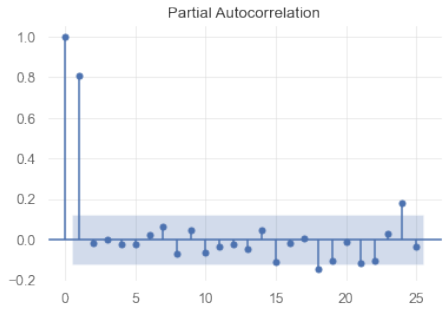

Figure 5: PACF Plot of detrended MTUM residuals

[5] Please refer to https://towardsdatascience.com/significance-of-acf-and-pacf-plots-in-time-series-analysis-2fa11a5d10a8 for more discussion about the significance of ACF and PACF plots for time series data

3 Tested Models

As mentioned in the introduction, the objective of the capstone is to build a full machine learning model portfolio of quantitative strategies. To achieve this task we will use Reinforcement Learning, and within Reinforcement Learning we will use Policy Gradient Methods. Policy Gradient Methods focus on estimating directly the Policy $\boldsymbol{\pi(a|s)}$ rather than the action value function, usually referred in Reinforcement Learning as $\boldsymbol{Q(s,a)}$. An advantage of Policy Gradient methods is that it only requires that the policy be parametric in any way as long as $\boldsymbol{\pi(a|s,\theta)}$ is differential with respect to its parameters. This is an interesting feature as it is not making any restriction on the action or state space. In our case, this is a necessary condition given that we want a full machine learning approach and our action space consists of all the possible combinations of weights $\boldsymbol{w}$ in a portfolio. This is a continuous infinite space which can't be directly estimated using Action-Value methods.

3.1 Policy Gradient Methods

The Policy Gradient Methods that we will explore in this capstone work by defining a performance function $\boldsymbol{J(\theta)}$ and then maximizing it using gradient ascent. By the Policy Gradient theorem, the gradient of $\boldsymbol{J(\theta)}$ is proportional to:

$$\boldsymbol{\nabla J(\theta) \propto \sum_{s} \mu(s) \sum_{a} q_{\pi}(s,a) \nabla \pi(a|s,\theta)}$$

Several algorithms can be used to solve this problem. For this capstone project, we will explore three different Policy Gradient methods: REINFORCE, REINFORCE with baseline, and Actor-Critic method.

3.1.1 REINFORCE

Our first model will be REINFORCE as a baseline and the simplest PG model. The Policy Gradient theorem gives an exact expression proportional to the gradient; all that is needed is some way of sampling whose expectation equals or approximates this expression. Notice that the right-hand side of the Policy Gradient theorem is a sum over states weighted by how often the states occur under the target policy $\boldsymbol{\pi}$; if $\boldsymbol{\pi}$ is followed, then states will be encountered in these proportions. Thus:

$$\begin{aligned}\boldsymbol{\nabla J(\theta)} &\boldsymbol{\propto \sum_{s} \mu(s) \sum_{a} q_{\pi}(s,a) \nabla \pi(a|s,\theta)} \\ &\boldsymbol{= E_{\pi}[\sum_{a} q_{\pi}(S_t,a) \nabla \pi(a|S_t,\theta)]}\end{aligned}$$

We could stop here and instantiate our stochastic gradient-ascent algorithm where $\boldsymbol{\hat{q}}$ is some learned approximation to $\boldsymbol{q_{\pi}}$. This algorithm has been called

an all-actions method because all actions are involved in the update. In the case of REINFORCE, the update at time t involves just A_t, the one action actually taken at time t.

$$\nabla J(\theta) = E_\pi[\sum_a \pi(a|S_t,\theta) q_\pi(S_t,a) \frac{\nabla \pi(a|S_t,\theta)}{\pi(a|S_t,\theta)}]$$

$$= E_\pi[q_\pi(S_t,A_t) \frac{\nabla \pi(a|S_t,\theta)}{\pi(a|S_t,\theta)}]$$

$$= E_\pi[G_t \frac{\nabla \pi(a|S_t,\theta)}{\pi(a|S_t,\theta)}]$$

Where we replaced a by a sample A_t.

The final expression in brackets is exactly what is needed, a quantity that can be sampled on each time step whose expectation is equal to the gradient. Using this sample to instantiate our generic stochastic gradient ascent algorithm yields the REINFORCE update:

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(a|S_t,\theta)}{\pi(a|S_t,\theta)}$$

3.1.2 REINFORCE with Baseline

As a stochastic gradient method, REINFORCE has good theoretical convergence properties. By construction, the expected update over an episode is in the same direction as the performance gradient. This assures an improvement in expected performance for sufficiently small α, and convergence to a local optimum under standard stochastic approximation conditions for decreasing α. However, as a Monte Carlo method, REINFORCE may be of high variance and thus produce slow learning. To improve REINFORCE, the addition of a baseline function can help speed the convergence of the algorithm. The Policy Gradient theorem (Sutton and Barto, Equation 13.5) can be generalized to include a comparison of the action value to an arbitrary baseline $b(s)$:

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a (q_\pi(s,a) - b(s)) \nabla \pi(a|s,\theta)$$

The baseline can be any function, including a random variable as long as it does not vary with a. One natural choice for the baseline is an estimate of the state value $\hat{v}(S_t, w)$; with this baseline the update rule in the gradient ascent will be:

$$\theta_t + 1 = \theta + \alpha(G_t - b(S_t)) \frac{\nabla \pi(a|S_t,\theta)}{\pi(a|S_t,\theta)}$$

3.1.3 Actor-Critic Methods

In REINFORCE with baseline, the learned state-value function estimates the value of the only the first state of each state transition. This estimate sets a baseline for the subsequent return, but is made prior to the transition's action and thus cannot be used to assess that action. In actor–critic methods, on the other hand, the state-value function is applied also to the second state of the transition. The estimated value of the second state, when discounted and added to the reward, constitutes the one-step return,$G_{t:t+1}$ which is a useful estimate of the actual return and thus is a way of assessing the action.

When the state-value function is used to assess actions in this way it is called a critic,and the overall policy-gradient method is termed an actor–critic method. Note that the bias in the gradient estimate is not due to bootstrapping as such; the actor would be biased even if the critic was learned by a Monte Carlo method.

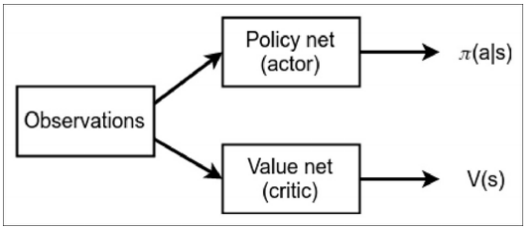

Figure 6: From Deep Reinforcement Learning Hands-On (Lapan, 2018)

The Actor-Critic Methods which we tested are the following:

3.1.4 One step Actor-Critic

One-step actor–critic methods replace the full return with the one-step return (and use a learned state-value function as the baseline) as follows:

$$\theta_{t+1} = \theta_t + \alpha(G_{t:t+1} - \hat{\nu}(S_t, w)\frac{\nabla\pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}$$

$$\theta_{t+1} = \theta_t + \alpha(R_{t+1}\gamma\hat{\nu}(S_{t+1}, w) - \hat{\nu}(S_t, w)\frac{\nabla\pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}$$

As Sutton and Barto (2018) notes, one of the advantages of one-step methods is that they are fully online and incremental, but they avoid the complexities of eligibility traces since they are a special case of the eligibility trace methods that is easier to understand. Pseudocode for one-step actor-critic is listed in Figure 7 below:

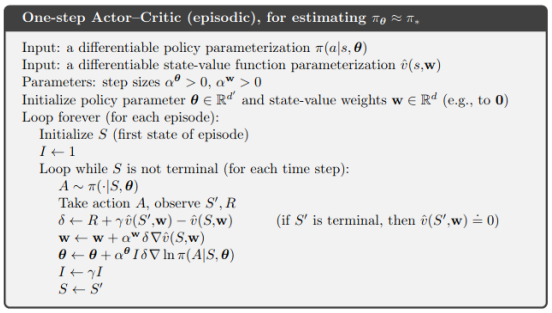

Figure 7: One-step Actor-Critic pseudocode (Sutton and Barto, 2018)

3.1.5 Actor-Critic with Eligibility Traces

The forward view of n-step methods can be generalized by replacing the one-step return with $\boldsymbol{G_{t:t+n}}$ and the forward view of a $\boldsymbol{\lambda}$-return algorithm is replaced by $\boldsymbol{G_t^{\lambda}}$. The backward view of the $\boldsymbol{\lambda}$-return algorithm uses separate eligibility traces for the actor and critic. Pseudocode for the complete algorithm is given in Figure 8 below.

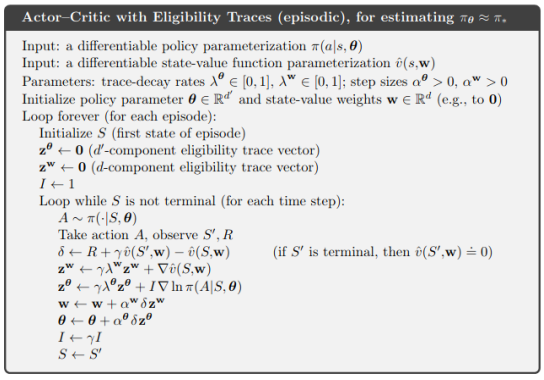

Figure 8: Actor-Critic with Eligibility Traces pseudocode (Sutton and Barto, 2018)

3.1.6 Policy Parameterization

So far we have discussed PG Methods that depend on a parameterization of the action space. For our particular case, we need a continuous parameterization. Our baseline parameterization will be a multivariate Gaussian policy defined as:

$$\boldsymbol{\pi(a|s,\theta)} = \frac{\boldsymbol{exp[-\frac{1}{2}(a-\mu(s,\theta))^T\Sigma_{(s,\theta)}^{-1}(a-\mu(s,\theta))]}}{\boldsymbol{\sqrt{2\pi^k|\Sigma_{(s,\theta)}|}}}$$

Where $\boldsymbol{k}$ is the number of assets in the portfolio.

For the parameterization of the mean and variance, we will try several approaches. First, we will use a linear parameterization where the feature space $\boldsymbol{x(s)}$ will be the same realized volatilites that we are using to build an Equal Risk Contribution (ERC) portfolio. The reason for choosing this initial parameterization is to have a sensitivity of how the algorithm performs when it gets the same information as the benchmark.

$$\boldsymbol{\mu(s, \theta) = \theta_{\mu}^{T} x(s)}$$

$$\boldsymbol{\sigma_{i,j}(s, \theta) = \theta_{\sigma_{i,j}}^{T} x(s)}$$

4 Evaluation and Validation

Each policy will be compared with several benchmark portfolios constructed using traditional portfolio optimization techniques. The portfolio benchmarks that we will use are: maximum return, maximum Sharpe ratio, and minimum volatility. [6]

4.1 Performance Metrics

The performance metrics are computed using the Critical Line Algorithm (CLA). The CLA is a robust alternative to the quadratic solver used to find mean-variance optimal portfolios, that is especially advantageous when we apply linear inequalities.

To evaluate our model, we will split the dataset in train and test sets. [7] After training our models, we will backtest the strategy using the test set. The performance metrics that we will use for the evaluation will be the following:

4.1.1 Maximum Return

The maximum return is a simple performance metric which is based on the maximum return the portfolio. We will utilize the CLA algorithm and compute the entire efficient frontier and extract the maximum return weights.

4.1.2 Sharpe Ratio

The Sharpe Ratio is a measure that indicates the average return minus the risk-free return divided by the standard deviation of return on an investment, also

[6] Benchmark returns are calculated using the public library PyPortfolioOpt. More details can be found in here https://pyportfolioopt.readthedocs.io/en/latest/

[7] Please refer to Section 4.2

known as volatility. We have selected to use the Sharpe Ratio as one of our performance metrics since it measures the performance of our portfolio compared to the risk-free asset, after adjusting for its risk. The Sharpe Ratio characterizes how well the return of an asset compensates the investor for the risk taken.

Sharpe Ratio Formula:

$$\boldsymbol{S_a = \frac{E[R_a - R_b]}{\sigma_a} = \frac{E[R_a - R_b]}{\sqrt{var[R_a - R_b]}}}$$

where $\boldsymbol{R_a}$ is the asset return, $\boldsymbol{R_b}$ is the risk-free return

4.1.3 Volatility

Volatility refers to the amount of uncertainty or risk related to the size of changes in a security's value. A higher volatility means that a security's value can potentially be spread out over a larger range of values. This means that the price of the security can change dramatically over a short time period in either direction. A lower volatility means that a security's value does not fluctuate dramatically, and tends to be more steady.

Volatility Formula:

$$\boldsymbol{\sigma_T = \sigma * \sqrt{T}}$$

where $\boldsymbol{T}$ is time.

4.2 Observed Statistics

To evaluate the performance of our models and the performance of our raw data, we will observe the following statistics in addition to the performance metrics.

4.2.1 Sortino Ratio

The Sortino Ratio measures the risk-adjusted return of the portfolio. The Sortino Ratio is similar to the Sharpe Ratio, but instead factors in a penalization for those returns falling below a target rate of return. Generally, the Sortino ratio is used as a way to compare the risk-adjusted performance of strategies with differing risk and return characteristics.

Sortino Ratio Formula:

$$\boldsymbol{S = \frac{R - T}{DR}}$$

where $\boldsymbol{R}$ is the portfolio average realized return, $\boldsymbol{T}$ is the target rate of return for the investment strategy, $\boldsymbol{DR}$ is the square root of the target semivariance (also known as downside deviation).

4.2.2 Calmar Ratio

Calmar Ratio is a function of the fund's average compounded annual rate of return versus its maximum drawdown. The benefits of the Calmar Ratio is that it changes gradually and serves to smooth out the over achievement and underachievement periods.

Calmar Ratio Formula:

$$\boldsymbol{Calmar\ Ratio} = \frac{\boldsymbol{R_p - R_f}}{\boldsymbol{Maximum\ Drawdown}}$$

where $\boldsymbol{R_p}$ is the portfolio return, $\boldsymbol{R_f}$ is the risk-free return, and $\boldsymbol{(R_p - R_f)}$ is also known as the average annual rate of return.

4.2.3 Draw-Downs

Draw-Downs are a measure of peak-to-trough decline during a specific period of time for an investment. Draw-Downs are important for measuring the historical risk of different investments.

Draw-Downs Formula:

$$\boldsymbol{D(T) = max[\max_{t\varepsilon(0,T)} X(t) - X(T), 0] \equiv [\max_{t\varepsilon(0,T)} X(t) - X(T)]}$$

where $\boldsymbol{T}$ is time and $\boldsymbol{X(T)}$ is the value of the asset at time $\boldsymbol{T}$.

4.3 Training and Testing Data Set

To estimate the performance of our reinforcement learning algorithms, we will create training and testing datasets. These datasets will be used to validate the performance of our models.

The Real-World Dataset ETF consists of data from January 2017 to November 2020. We begin by dividing the Real-World Dataset ETF closing price history into two datasets, the training set which includes ETF closing price history from February 2019 to February 2020, and a test dataset with ETF closing price history from March 2020 to November 2020. We selected the training and testing data sets due to the significant draw-downs observed in December 2018 to January 2019 and February 2020 to March 2020.

Figure 9 shows the Real-World Dataset ETF returns for the training dataset. Figure 10 shows the Real-World Dataset ETF returns for the testing dataset. Due to the significant draw-downs observed in February 2020 and the rapid market bounce starting March 2020, our testing dataset observed significant levels of

returns and volatilities. Moving forward to our results, our performance metric benchmarks will be based on the training dataset, while the backtests performed will be based on the testing dataset.

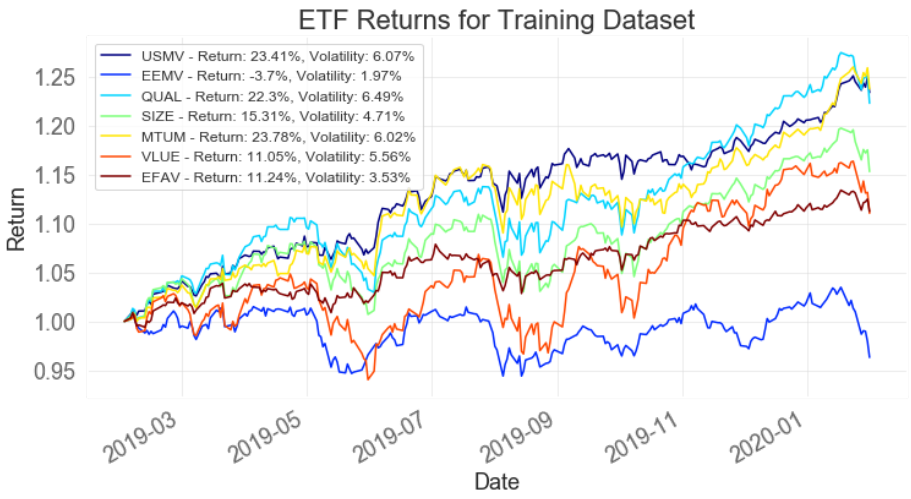

Figure 9: ETF Returns for Training Data

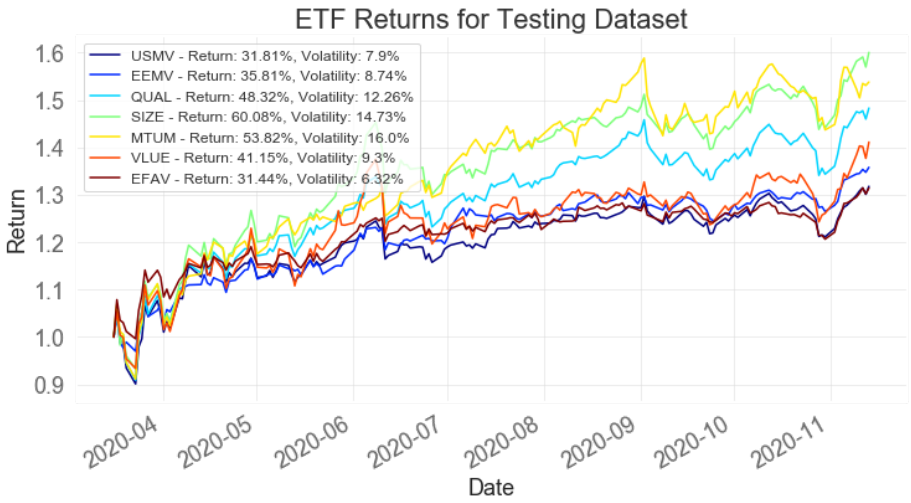

Figure 10: ETF Returns for Test Data

5 Results

5.1 Control Dataset

We tested all our models on a simulated 2-asset environment with a one-step reward defined as follows: on each observation the agent receives a reward $\boldsymbol{R_t}$:

$\boldsymbol{R_t = \lambda \Delta \Pi_t - (1-\lambda) a_t^T \Sigma a_t}$

where:

$\boldsymbol{\Delta \Pi_t}$ = Portfolio return between t and t+1 given action $\boldsymbol{a_t}$ at time 't'

$\boldsymbol{a_t}$ = action at time 't' which is the vector of weights

$\boldsymbol{\Sigma}$ = asset covariance matrix at time 't'

$\boldsymbol{\lambda}$ = Risk aversion parameter

The risk aversion parameter can be adjusted in order to account for the trade-off between risk and reward. Setting $\boldsymbol{\lambda = 0}$ would represent in a minimum volatility reward function, while setting $\boldsymbol{\lambda = 1}$ would represent a maximum return reward function.

The results were consistent across different levels of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. Below is an example of the model convergence using a predefined choice of parameters and for each of the previously discussed reward functions.

Asset	$\boldsymbol{\mu}$	$\boldsymbol{\sigma}$
0	.02	.01
1	.18	.03

Below we present the average weights per epoch at different levels of risk aversion $\boldsymbol{\lambda}$. As expected, we can see that when $\boldsymbol{\lambda = 0}$ (minimum volatility case) the weights converge to 100% investment in the asset with less volatility (Asset 0) and for the cases when and when $\boldsymbol{\lambda = 1}$ (maximum return case) the weights converge to 100% in the asset with higher return (Asset 1).

5.1.1 Model Convergence

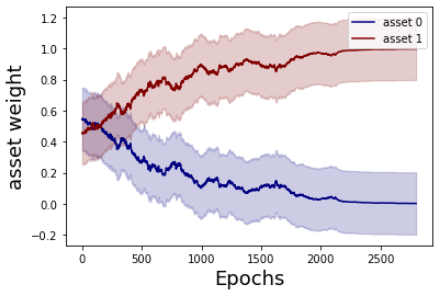

Figure 11: REINFORCE, $\boldsymbol{\lambda = 1}$

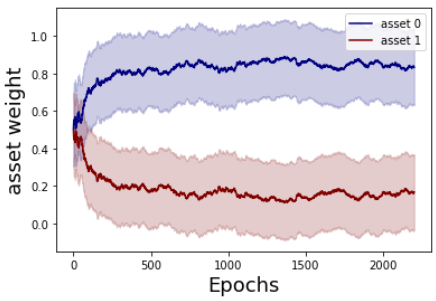

Figure 12: REINFORCE, $\boldsymbol{\lambda = 0}$

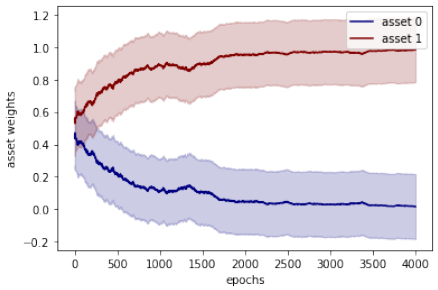

Figure 13: REINFORCE with baseline, $\boldsymbol{\lambda = 1}$

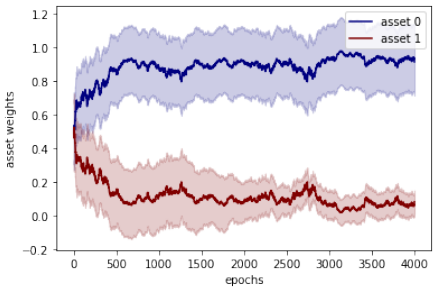

Figure 14: REINFORCE with baseline, $\boldsymbol{\lambda = 0}$

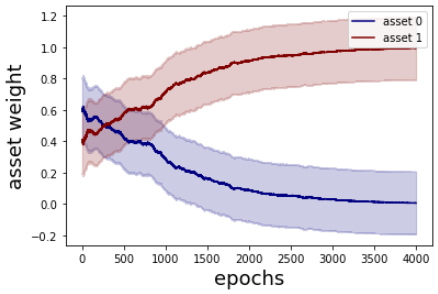

Figure 15: Actor Critic, $\boldsymbol{\lambda = 1}$

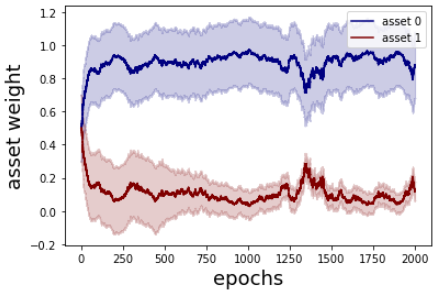

Figure 16: Actor Critic, $\boldsymbol{\lambda = 0}$

5.2 Real Dataset

Now that we have successfully demonstrated the operation of the Policy Gradient Methods with our control dataset, we then test the following Policy Gradient Methods on real-world data: REINFORCE, REINFORCE with Baseline, Actor-Critic, and Actor-Critic with Eligibility traces.

5.2.1 Two Asset Portfolio

We begin our evaluation of the real-world data by testing the PG Methods on a two asset portfolio containing MTUM (the higher return asset) and EFAV (the lower volatility asset) to check for model convergence and correct operation. For each algorithm, we train our models for five different cases of $\boldsymbol{\lambda}$: 0, 0.2, 0.5, 0.8, and 1 to see how these algorithms select a portfolio which balances between different levels of risk and reward. We discuss our findings for the $\boldsymbol{\lambda} = 0$ (minimum volatility) and $\boldsymbol{\lambda} = 1$ (maximum return) cases as well as the impact of $\boldsymbol{\lambda}$ in asset allocation in this section. Please refer to Appendix B, Section 8.2 for the results of the $\boldsymbol{\lambda} = 0.2$, 0.5, and 0.8 cases.

REINFORCE
$\boldsymbol{\lambda = 0}$ Case (Minimum Volatility)

Figure 17 shows how the REINFORCE algorithm selects a portfolio with low volatility (identified in blue) for a risk aversion parameter $\boldsymbol{\lambda = 0}$. This is expected since $\boldsymbol{\lambda = 0}$ represents the minimum volatility portfolio. The stable evolution of backtests over the epoch training demonstrate that the model has converged on an optimal solution. Comparing our training backtest to the benchmark portfolios, the backtest shows that the portfolio selected by REINFORCE has a volatility (3.53%) which is lower than the minimum volatility benchmark's volatility (3.95%).

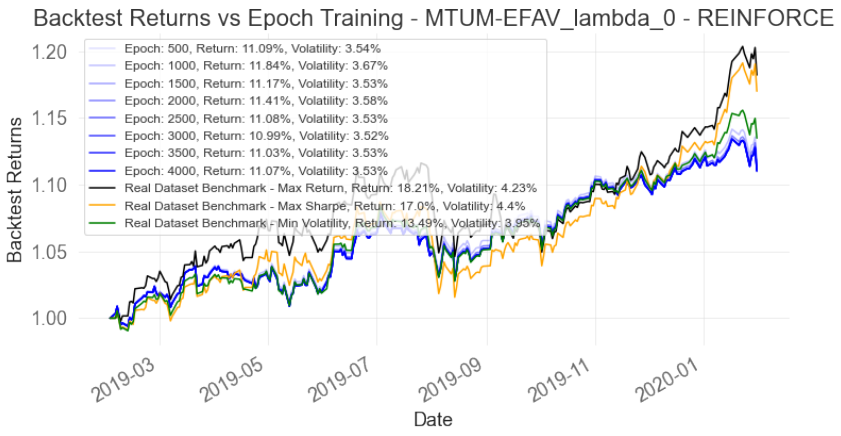

Figure 17: Two Asset REINFORCE Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0}$

Figure 18 shows the contribution of the reward and volatility components of the reward function. The components of the reward function are normalized so that they both have approximately the same magnitude at the start of the model training. Since $\boldsymbol{\lambda = 0}$, the reward function in our case is represented as $\boldsymbol{R_t = -a_t^T \Sigma a_t}$ (negative magnitude of the volatility component).

Figure 19 shows the asset weight distribution over the model training. The model converges on the correct solution for a minimum volatility reward function with $\boldsymbol{\lambda = 0}$, consistently allocating 100% of the portfolio in EFAV (the lower volatility asset) after 2500 epochs of training.

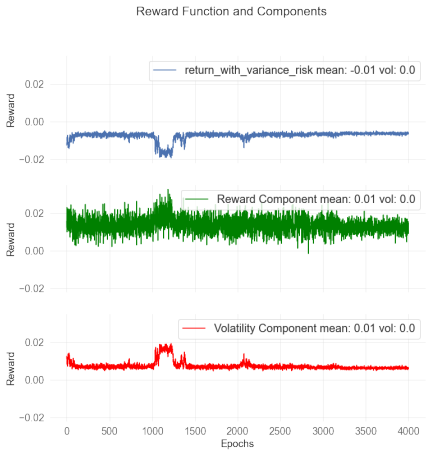

Figure 18: Two Asset REINFORCE Reward Function $\boldsymbol{\lambda = 0}$

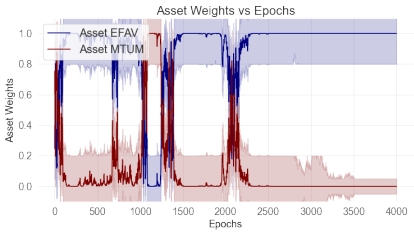

Figure 19: Two Asset REINFORCE Asset Weights $\boldsymbol{\lambda = 0}$

Using the asset weights from the training period, we perform a backtest on the test dataset and compare the backtest return to the benchmarks as seen in Figure 20. We see that the backtest has a volatility (4.1%) lower than the benchmark minimum volatility case (4.23%). This demonstrates that our REINFORCE model is performing as expected to find the minimum volatility portfolio for a two asset portfolio.

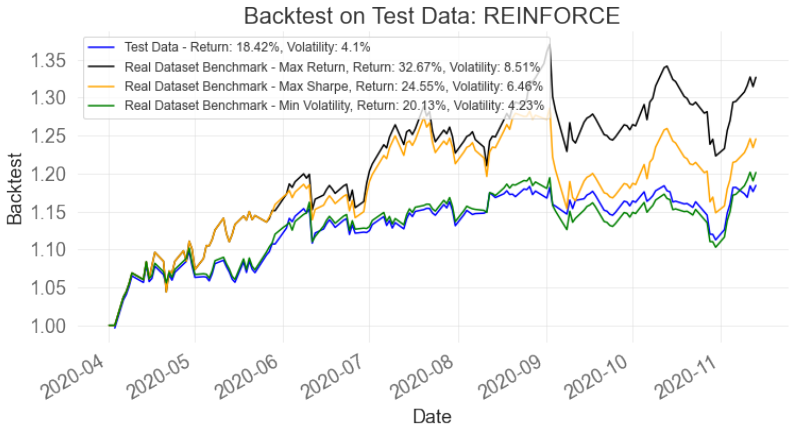

Figure 20: Two Asset REINFORCE Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0}$

$\boldsymbol{\lambda = 1}$ Case (Maximum Return)

In Figure 21, we demonstrate the operation of the REINFORCE algorithm using a risk aversion parameter of $\boldsymbol{\lambda = 1}$. This represents the maximum return reward function, and the backtest of the training data shows that the return we obtained is the maximum return (24.03%), exceeding the returns of maximum return benchmark portfolio (18.21% return). We also note that the model has converged during model training, as evidenced by the consistent backtest results across the last few sets of training epochs.

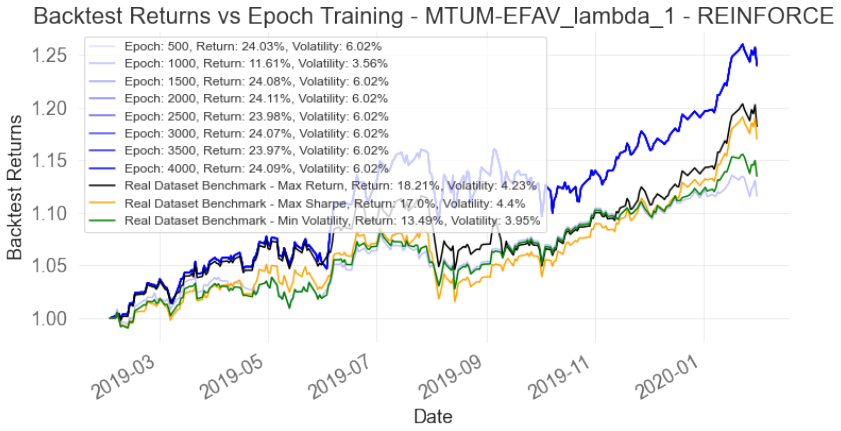

Figure 21: Two Asset REINFORCE Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

Figure 22 shows the contribution of the reward and volatility components of the reward function. Since $\boldsymbol{\lambda = 1}$, the reward function in our case is represented as $\boldsymbol{R_t = \Delta\Pi_t}$ (the return component) without any contribution from the volatility component.

Figure 23 shows the asset weight distribution over the model training. The model converges on the correct solution, consistently allocating 100% of the portfolio in MTUM (the higher return asset) after approximately 1300 epochs of training.

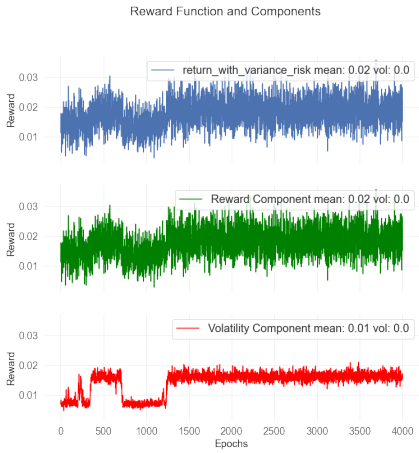

Figure 22: Two Asset REINFORCE Reward Function $\boldsymbol{\lambda = 1}$

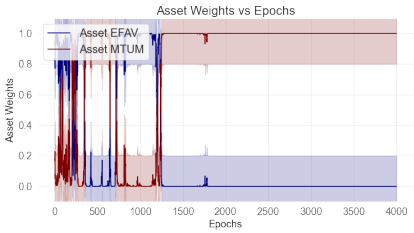

Figure 23: Two Asset REINFORCE Asset Weights $\boldsymbol{\lambda = 1}$

In Figure 24, we perform a backtest on the test dataset and note that it has a higher return (54.08%) then the three benchmarks. This demonstrates that our REINFORCE model can return a maximum return portfolio with two assets on a test dataset it has not been trained on. We also note that the volatility of our test backtest is 13.06%, so the maximum return does come at the expense of a higher volatility.

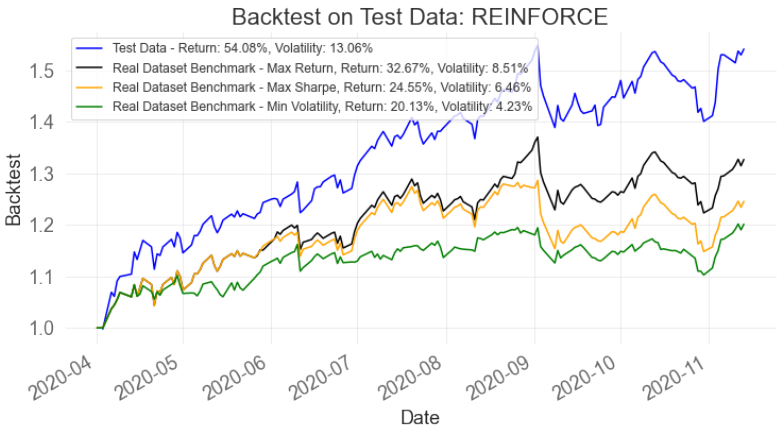

Figure 24: Two Asset REINFORCE Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

Asset Allocation based on $\boldsymbol{\lambda}$

In Figure 19 and Figure 23, we show the evolution of the asset weights over the course of the model training based on the $\boldsymbol{\lambda} = \mathbf{0}$ and $\boldsymbol{\lambda} = \mathbf{1}$ cases. In Appendix B, Section 8.2 we show the evolution of the asset weights for the $\boldsymbol{\lambda} = \mathbf{0.2}$ (Figure 105), $\boldsymbol{\lambda} = \mathbf{0.5}$ (Figure 121) and $\boldsymbol{\lambda} = \mathbf{0.8}$ (Figure 137) cases. For $\boldsymbol{\lambda} = \mathbf{0.2}$, the REINFORCE algorithm converges and picks EFAV, the lower volatility asset. This is expected as the volatility component is favored over the reward component.

However, for $\boldsymbol{\lambda} = \mathbf{0.5}$ the REINFORCE algorithm alternates between choosing EFAV or MTUM, or a portion of the two assets. This reflects the algorithm's balance between the return and volatility components and since those two components of the reward function have an equal weight, neither component is heavily favored.

For the $\boldsymbol{\lambda} = \mathbf{0.8}$ case, the algorithm alternates between choosing EVAV or MTUM but eventually settles on choosing MTUM, the higher return asset. The algorithm considers both the reward and volatility components, but ultimately the reward component is weighted more so it chooses the higher return asset.

Based on these results with the two asset portfolio, it appears that setting a $\boldsymbol{\lambda}$ value greater than 0 and less than 0.5 would ultimately result in a minimum volatility portfolio, while setting $\boldsymbol{\lambda}$ greater than 0.5 and less than 1 would eventually result in a maximum return portfolio. At $\boldsymbol{\lambda} = \mathbf{0.5}$, the REINFORCE algorithm is continuously trying to balance between risk and reward and as a result, the asset weights are not stable over the training period.

REINFORCE with Baseline

$\boldsymbol{\lambda = 0}$ Case (Minimum Volatility)

In Figure 25, the REINFORCE with Baseline algorithm selects a low volatility portfolio (3.53%) for a risk aversion parameter $\boldsymbol{\lambda = 0}$ during model training. Although there is some variation in the backtests at the start of the model training (as evidenced by the light blue backtest returns in the plot), the subsequent backtests with the dark blue backtests are fairly stable and consistent.

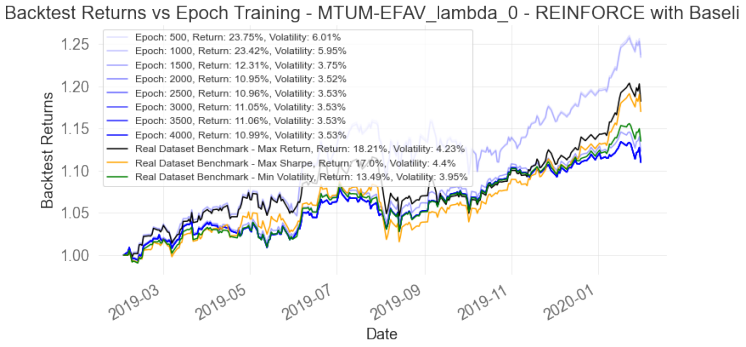

Figure 25: Two Asset REINFORCE with Baseline Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0}$

Figure 26 shows that the backtest of the test dataset has a volatility of 4.11%, which is below the benchmark minimum volatility portfolio's volatility of 4.23%. Please refer to Appendix B, Section 8.2 for the reward function and the asset weight plots.

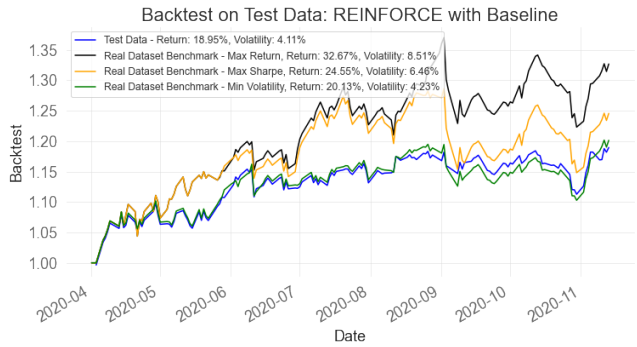

Figure 26: Two Asset REINFORCE with Baseline Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0}$

$\boldsymbol{\lambda = 1}$ Case (Maximum Return)

Figure 27 shows that with a maximum return reward function ($\boldsymbol{\lambda = 1}$), the REINFORCE with Baseline algorithm selects the highest return portfolio (24.03%) for the training data set, with a higher return than the maximum return benchmark (18.21%). This conclusion is also shown with the test dataset in Figure 28, where the test benchmark has a higher return (54.34%) than the maximum return benchmark (32.67%). Please refer to Appendix B, Section 8.2 for the reward function and the asset weight plots.

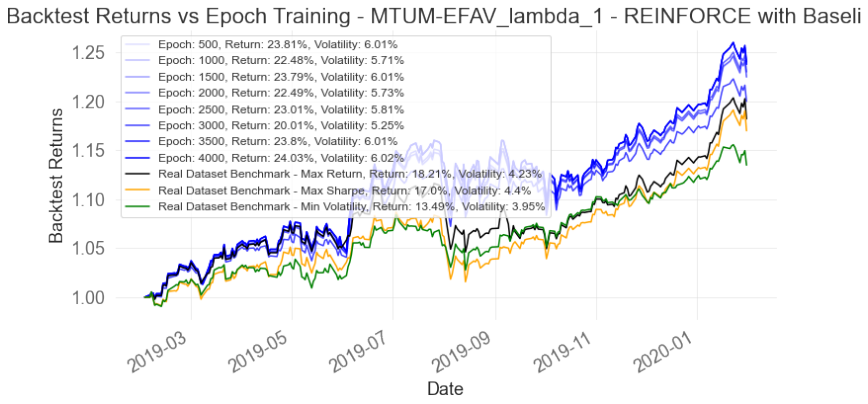

Figure 27: Two Asset REINFORCE with Baseline Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

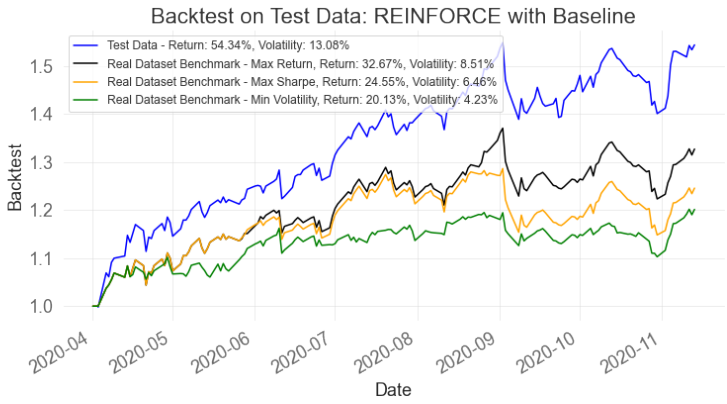

Figure 28: Two Asset REINFORCE with Baseline Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

Actor-Critic

$\boldsymbol{\lambda = 0}$ Case (Minimum Volatility)

In Figure 29, the Actor-Critic algorithm converges on a portfolio with a volatility (3.53%) lower than the minimum volatility benchmark portfolio's volatility (3.95%). Figure 30 shows that the Actor-Critic algorithm also selects the minimum volatility portfolio for the test dataset, with a backtest volatility of 4.12% as compared to the minimum volatility benchmark's volatility of 4.23% in the test dataset. Please refer to Appendix B, Section 8.2 for the reward function and the asset weight plots.

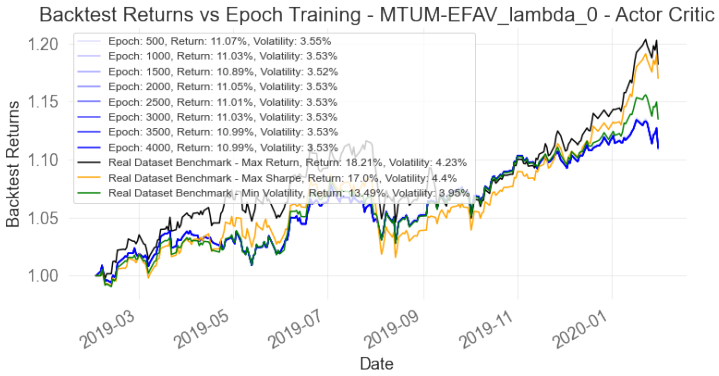

Figure 29: Two Asset Actor-Critic Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0}$

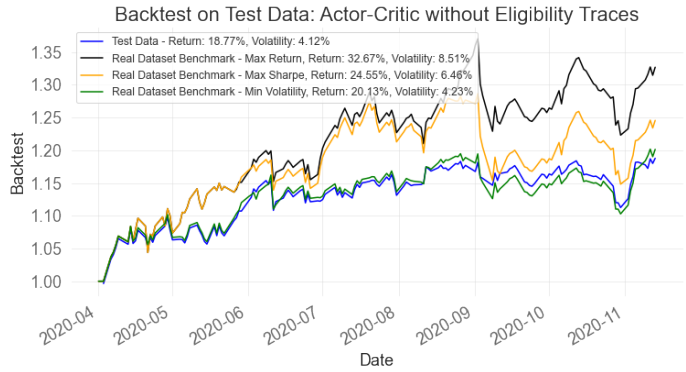

Figure 30: Two Asset REINFORCE with Baseline Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0}$

$\boldsymbol{\lambda = 1}$ Case (Maximum Return)

Figure 31 shows that with a maximum return reward function ($\boldsymbol{\lambda = 1}$), the Actor-Critic algorithm selects a maximum return portfolio (selected portfolio's return is 23.79%, while the maximum return benchmark's return is 18.21%). When we apply the weights learned from the training dataset to the test dataset in Figure 32 and perform a backtest, the backtest has a return of 52.87%, which is higher than the test dataset maximum return benchmark's return of 32.67%. Please refer to Appendix B, Section 8.2 for the reward function and the asset weight plots.

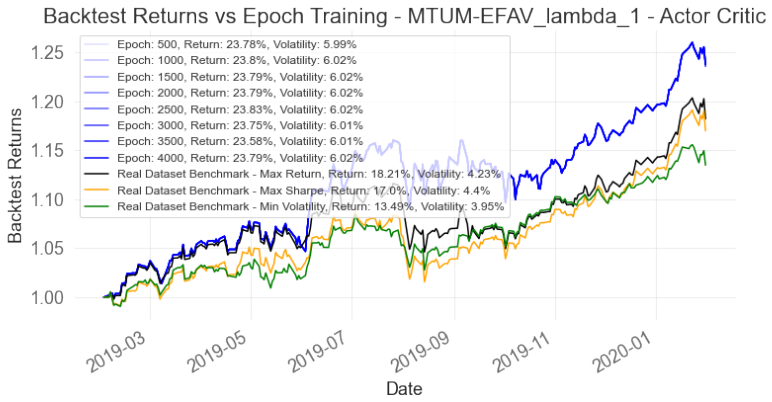

Figure 31: Two Asset Actor-Critic Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

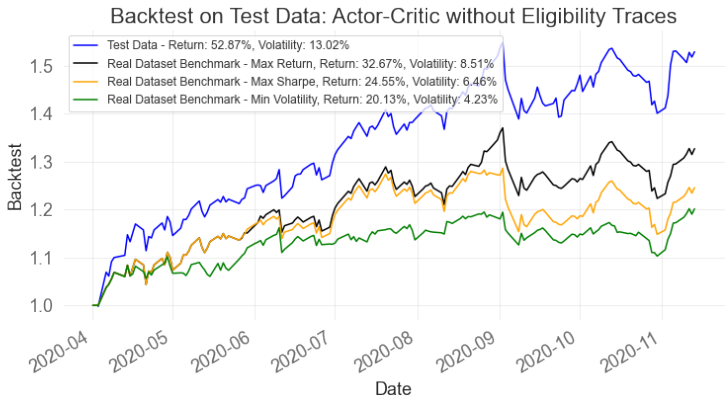

Figure 32: Two Asset Actor-Critic Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

Actor-Critic with Eligibility Traces

$\boldsymbol{\lambda = 0}$ Case (Minimum Volatility)

In Figure 33, the Actor-Critic with Eligibility Traces algorithm converges on a minimum volatility portfolio for a risk aversion parameter $\boldsymbol{\lambda = 0}$ during model training. The training backtest has volatility of 3.53% across the majority of the training epochs, and this is lower than the minimum volatility benchmark of 3.95%. Figure 34 shows that the Actor-Critic with Eligibility Traces algorithm selects the minimum volatility portfolio for the test dataset (volatility of 4.1% while the minimum volatility benchmark has a volatility of 4.23%) Please refer to Appendix B, Section 8.2 for the reward function and the asset weight plots.

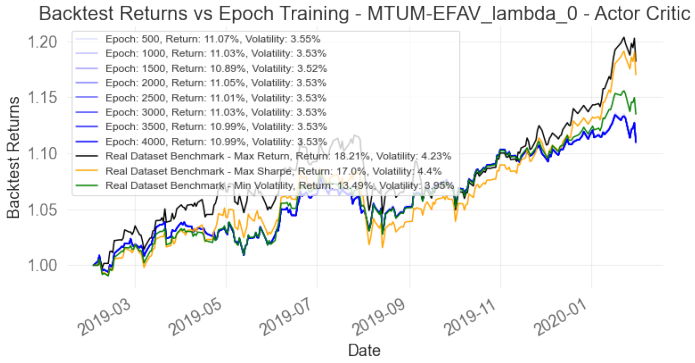

Figure 33: Two Asset Actor-Critic Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0}$

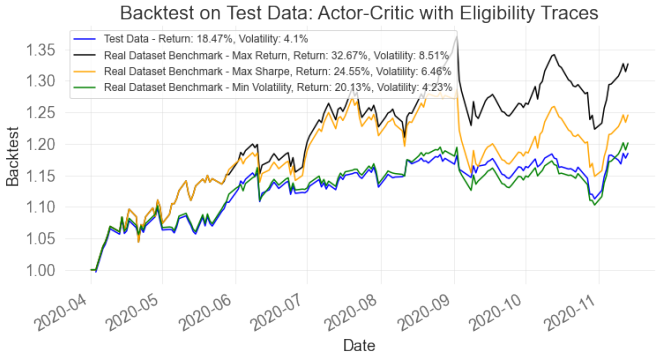

Figure 34: Two Asset Actor-Critic with Eligibility Traces Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0}$

$\boldsymbol{\lambda = 1}$ Case (Maximum Return)

Figure 35 shows that with a max return reward function ($\boldsymbol{\lambda = 1}$), the Actor-Critic with Eligibility Traces algorithm selects a portfolio with a return of 23.79%, which is higher than the maximum return benchmark's return (18.21%). Figure 36 shows that a backtest of the test dataset has a return of 53.81% while the maximum return benchmark has a return of 32.67%. Please refer to Appendix B, Section 8.2 for the reward function and the asset weight plots.

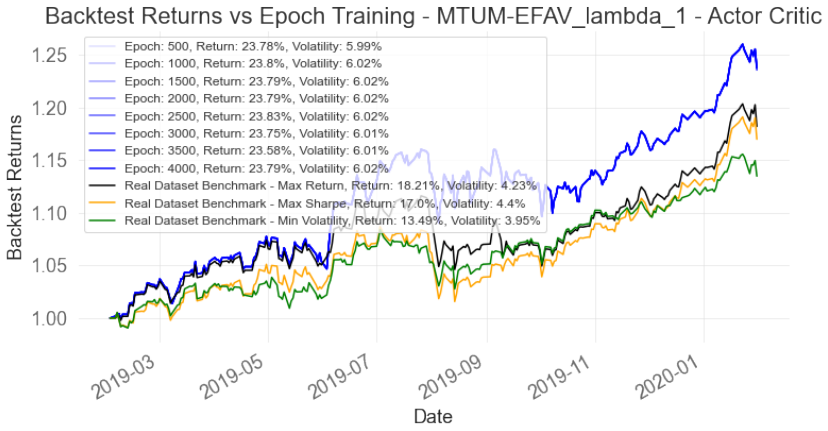

Figure 35: Two Asset Actor-Critic with Eligibility Traces Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

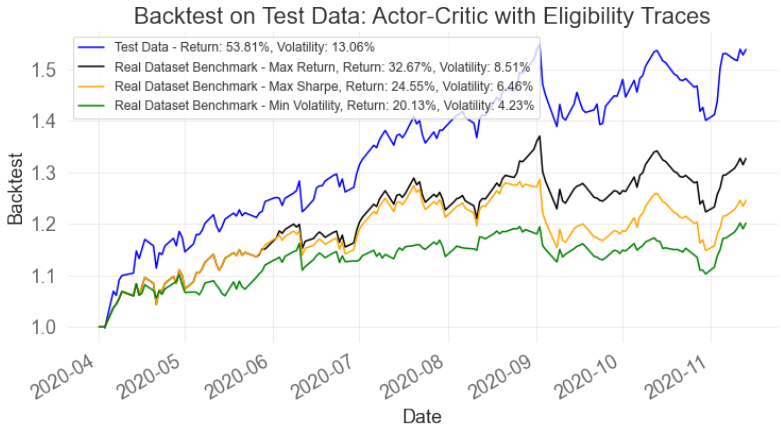

Figure 36: Two Asset Actor-Critic with Eligibility Traces Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

5.2.2 Full Portfolio

Now that we have tested our Policy Gradient Methods on a portfolio of two real assets, we now test our Policy Gradient Methods on our full portfolio of ETFs. For each algorithm, we train our models for five different cases of $\boldsymbol{\lambda}$: 0, 0.2, 0.5, 0.8, and 1 to see how these algorithms select a portfolio which balances between different levels of risk and reward. We discuss our findings for the $\boldsymbol{\lambda} = 0$ (minimum volatility), $\boldsymbol{\lambda} = 0.5$, and $\boldsymbol{\lambda} = 1$ (maximum return) cases as well as the impact of $\boldsymbol{\lambda}$ in asset allocation in this section. Please refer to Appendix B, Section 8.3 for the results of the $\boldsymbol{\lambda} = 0.2$ and 0.8 cases.

REINFORCE
$\boldsymbol{\lambda} = 0$ Case (Minimum Volatility)

Figure 37 shows how the REINFORCE algorithm selects a portfolio with low volatility (identified in blue) for a risk aversion parameter $\boldsymbol{\lambda} = \mathbf{0}$. This is expected since $\boldsymbol{\lambda} = \mathbf{0}$ represents the minimum volatility portfolio. The stable evolution of backtests over the epoch training demonstrate that the model has converged on an optimal solution. Comparing our training backtest to the benchmark portfolios, the backtest shows that the portfolio selected by REINFORCE has a volatility (3.52%) lower than the benchmark minimum volatility portfolio (4.62%).

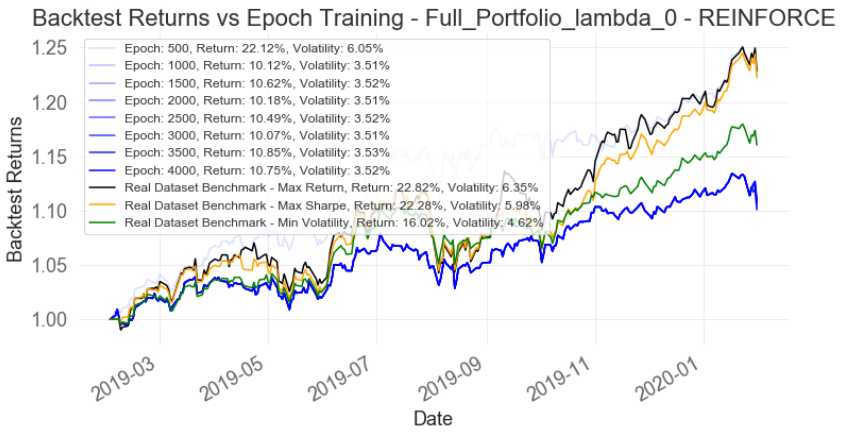

Figure 37: Full Portfolio REINFORCE Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda} = \mathbf{0}$

Figure 38 shows the contribution of the reward and volatility components of the reward function. As previously discussed, the components of the reward

function are normalized so that they both have approximately the same magnitude at the start of the model training. Since $\boldsymbol{\lambda} = \mathbf{0}$, the reward function in this case is represented as the negative magnitude of the volatility component.

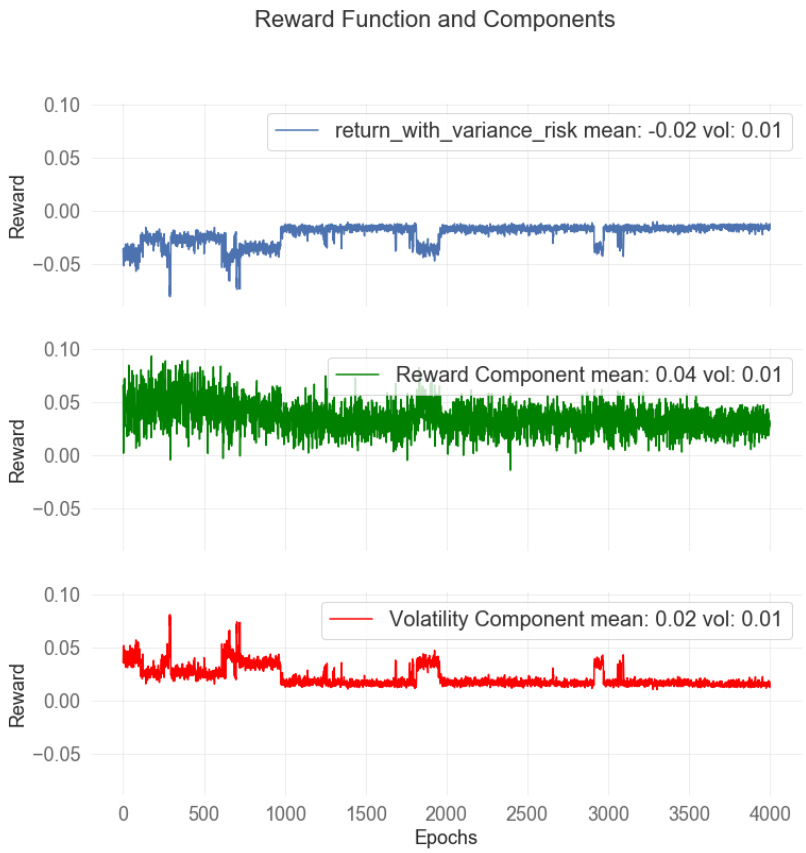

Figure 38: Full Portfolio REINFORCE Reward Function $\boldsymbol{\lambda} = \mathbf{0}$

Figure 39 shows the evolution of the asset weight distribution over the model training. Although the asset weights evolve during the model training the algorithm generally chooses to place 100% of the portfolio in EFAV, which is one of the lowest volatility assets in our portfolio. As shown in Figure 37, the backtest results are stable during the model training.

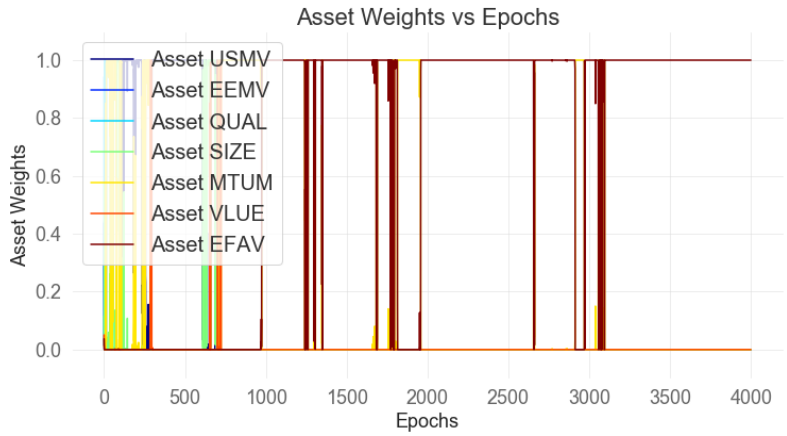

Figure 39: Full Portfolio REINFORCE Asset Weights $\boldsymbol{\lambda = 0}$

Using the asset weights from the training period, we perform a backtest on the test dataset and compare the backtest return to the benchmarks as seen in Figure 40. We see that the backtest has a volatility (4.27%) lower than the benchmark minimum volatility case (4.39%). This demonstrates that our REINFORCE model is performing as expected to find the minimum volatility portfolio for our full portfolio dataset.

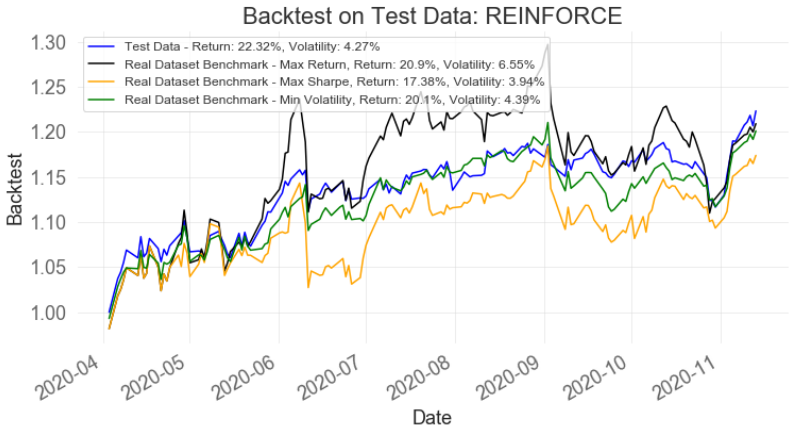

Figure 40: Full Portfolio REINFORCE Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0}$

$\boldsymbol{\lambda = 0.5}$ Case

Figure 41 shows how the REINFORCE algorithm selects a portfolio with low volatility (identified in blue) for a risk aversion parameter $\boldsymbol{\lambda = 0.5}$. This parameter represents a tradeoff between the the minimum volatility portfolio ($\boldsymbol{\lambda = 0}$) and maximum return portfolio ($\boldsymbol{\lambda = 1}$), where both components of the reward function (reward and volatility) have an equal weight. The stable evolution of backtests over the epoch training demonstrate that the model has converged on an optimal solution. Comparing our training backtest to the benchmark portfolios, the backtest shows that the portfolio selected by REINFORCE in the training data appears to have a volatility and return close to the benchmark maximum return portfolio (the backtest has a return of 22.33% and a volatility of 6.06%, while the benchmark has a return of 22.82% and a volatility of 6.35%).

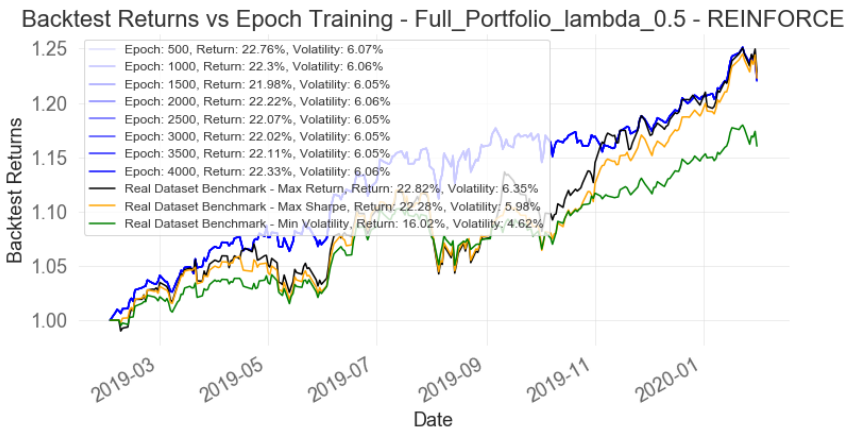

Figure 41: Full Portfolio REINFORCE Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.5}$

Figure 42 shows the contribution of the reward and volatility components of the reward function. The components of the reward function are normalized so that they both have approximately the same magnitude at the start of the model training. Since $\boldsymbol{\lambda = 0.5}$, the reward function in our case is represented as $\boldsymbol{R_t = 0.5\Delta\Pi_t - 0.5a_t^T\Sigma a_t}$ (the reward function depends on both the reward component and the volatility component equally).

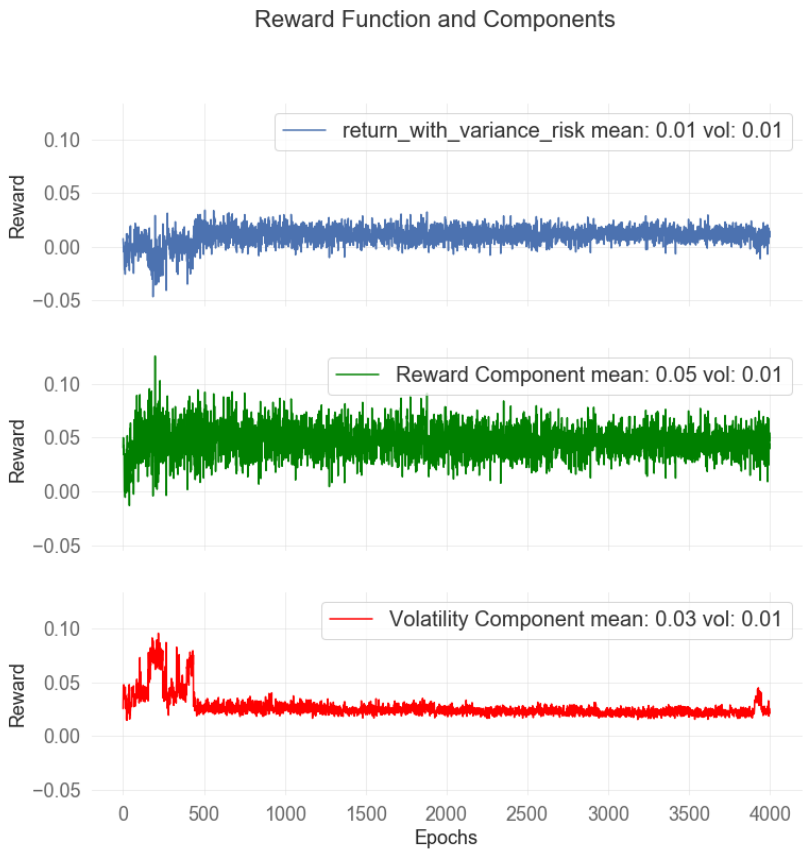

Figure 42: Full Portfolio REINFORCE Reward Function $\boldsymbol{\lambda = 0.5}$

Figure 43 shows the asset weight distribution over the model training. In general, the model allocates 100% of the portfolio in USMV, which is one of the higher return assets in our portfolio, although the volatility of this ETF in the training dataset is also higher than many of the other ETFs. Although the asset weights evolve during the model training, the training backtest in Figure 41 is fairly stable over the course of the model training.

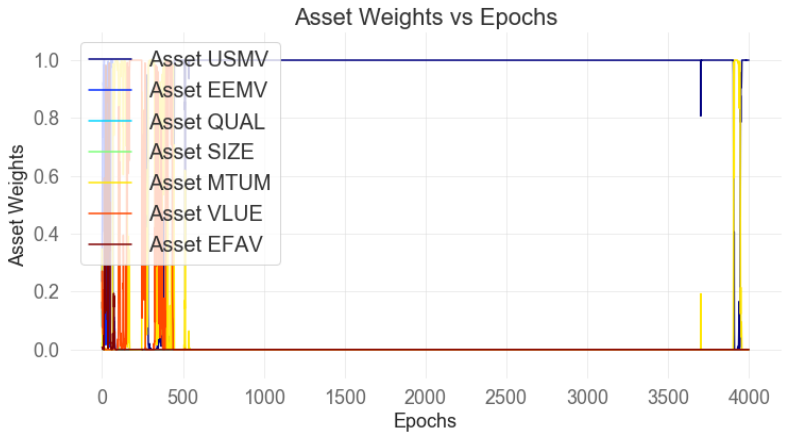

Figure 43: Full Portfolio REINFORCE Asset Weights $\boldsymbol{\lambda = 0.5}$

Using the asset weights from the training period, we perform a backtest on the test dataset and compare the backtest return to the benchmarks as seen in Figure 44. We see that the backtest has a volatility (5.55%) higher than the benchmark minimum volatility case (4.39%) but lower than the benchmark maximum return case (6.55%). The backtest also has a return of 30.47%, which is higher thn the benchmark maximum return case of 20.9%. This demonstrates that in the $\boldsymbol{\lambda = 0.5}$ case, the REINFORCE algorithms finds a portfolio which tries to maximize the return while controlling the volatility.

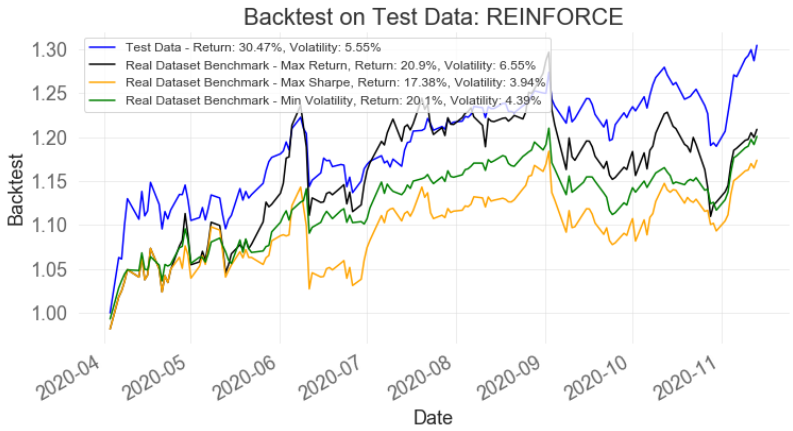

Figure 44: Full Portfolio REINFORCE Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.5}$

$\boldsymbol{\lambda = 1}$ Case (Maximum Return)

In Figure 45, we demonstrate the operation of the REINFORCE algorithm using a risk aversion parameter of $\boldsymbol{\lambda = 1}$. This represents the maximum return reward function. The backtest of the training data is fairly stable after the first 500 epochs, which indicates model convergence and the return is 22.36% (this is approximately the same as the maximum return benchmark, which has a return of 22.82

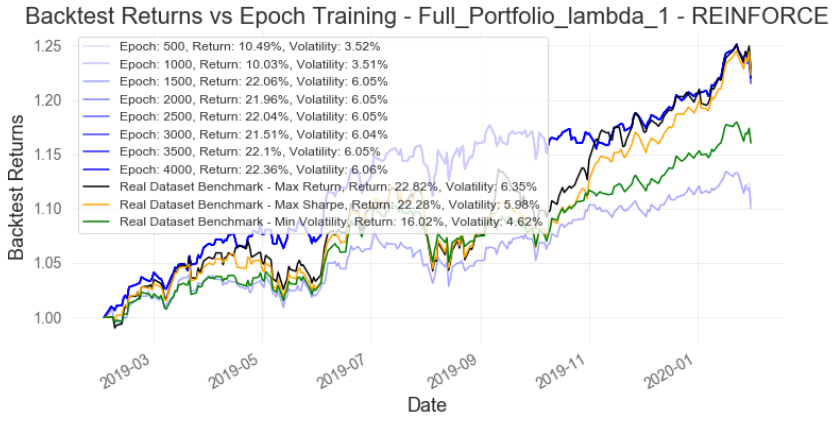

Figure 45: Full Portfolio REINFORCE Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

Figure 46 shows the contribution of the reward and volatility components of the reward function. Since $\boldsymbol{\lambda = 1}$, the reward function in our case is represented by return component without any contribution from the volatility component.

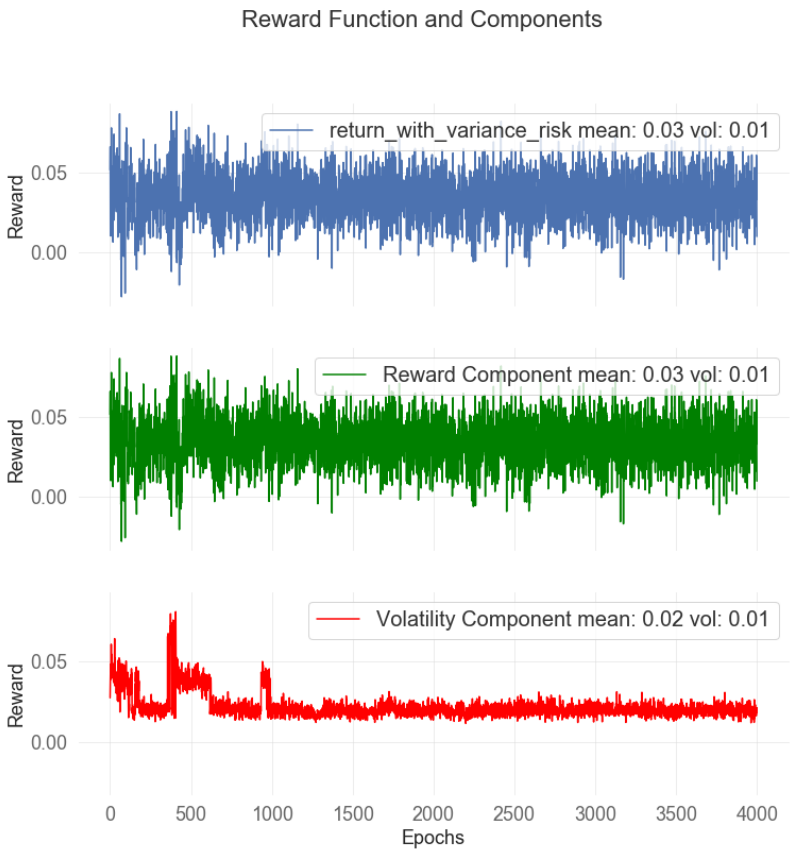

Figure 46: Full Portfolio REINFORCE Reward Function $\boldsymbol{\lambda = 1}$

Figure 47 shows the asset weight distribution over the model training. The model allocates 100% of the assets to USMV, which is one of the higher return ETFs in our portfolio.

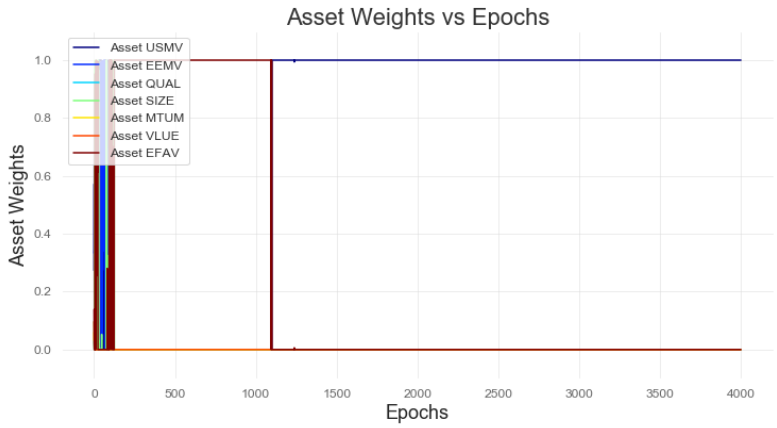

Figure 47: Full Portfolio REINFORCE Asset Weights $\boldsymbol{\lambda = 1}$

In Figure 48, we perform a backtest on the test dataset and note that it has a higher return (30.84%) then the maximum return benchmark portfolio (20.9% return). This demonstrates that our REINFORCE model can return a maximum return portfolio given a portfolio of ETFs.

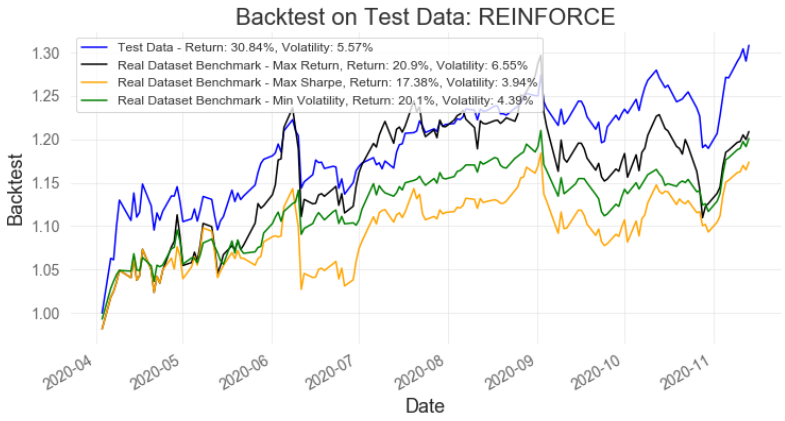

Figure 48: Full Portfolio REINFORCE Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

Asset Allocation based on $\boldsymbol{\lambda}$

In Figure 39, Figure 43, and Figure 47, we discussed the asset weights the REINFORCE algorithm allocated for the $\boldsymbol{\lambda = 0}$, $\boldsymbol{\lambda = 0.5}$, and $\boldsymbol{\lambda = 1}$ cases respectively. In Appendix B, Section 8.3 we show the asset weights for the $\boldsymbol{\lambda = 0.2}$ and $\boldsymbol{\lambda = 0.8}$ cases. For $\boldsymbol{\lambda = 0.2}$ (Figure 164), the reward function favors minimizing the volatility component over maximizing the reward component. The REINFORCE algorithm's asset allocation evolves over the course of the training period, but eventually converges on allocating the full portfolio in EEMV, which has the lowest volatility of all the ETFs in the portfolio.

For the $\boldsymbol{\lambda = 0.8}$ case (Figure 186), the algorithm alternates between choosing MTUM, USMV, or QUAL and ultimately ends with choosing USMV. These three ETFs are the top three ETFs in our portfolio for having the highest returns in the training period. As the return component of the reward function is favored over the volatility component in the $\boldsymbol{\lambda = 0.8}$ case, it makes sense that the REINFORCE algorithm is allocating the higher return assets in the portfolio.

Based on these results with the full portfolio of ETFs, it appears that setting a $\boldsymbol{\lambda}$ value greater than 0 and less than 0.5 would result in a minimum volatility portfolio, while setting $\boldsymbol{\lambda}$ greater than 0.5 and less than 1 would result in a maximum return portfolio. At $\boldsymbol{\lambda = 0.5}$, although there is a balance between risk and reward, in the full portfolio scenario it appears that the REINFORCE algorithm selects a higher return asset, at the expense of a higher volatility. This is a finding different from the two-asset portfolio with $\boldsymbol{\lambda = 0.5}$, where there was no clear preference between the maximum return asset and the minimum volatility asset.

REINFORCE with Baseline

$\boldsymbol{\lambda = 0}$ Case (Minimum Volatility)

In Figure 49, the REINFORCE with Baseline algorithm selects a low volatility portfolio (3.52%) for a risk aversion parameter $\boldsymbol{\lambda = 0}$ during model training (this is a lower volatility then the minimum volatility benchmark portfolio's volatility). With the exception of one epoch training result, the other backtest returns are fairly stable. Figure 50 shows that the REINFORCE with baseline algorithm selects the minimum volatility portfolio for the test dataset (the backtest has a volatility of 4.29%, which is below the benchmark minimum volatility portfolio's volatility of 4.39%). Please refer to Appendix B, Section 8.3 for the reward function and the asset weight plots.

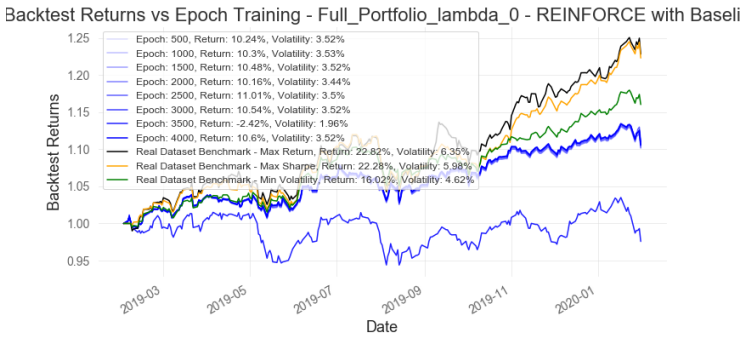

Figure 49: Full Portfolio REINFORCE with Baseline Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0}$

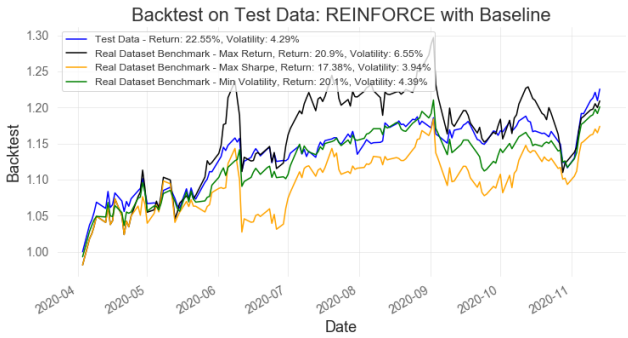

Figure 50: Full Portfolio REINFORCE with Baseline Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0}$

$\lambda = 0.5$ Case

In Figure 51, the REINFORCE with Baseline algorithm tries to balance risk versus reward and selects a portfolio which appears to be similar to the maximum return or maximum Sharpe benchmarks. The backtest of the training set has a return of 22.16% and a volatility of 6.06%. In Figure 52, we apply the asset weights learned in the model training to the test dataset. The backtest has a return of 29.86% and a volatility of 5.52%; the return is higher than the benchmark returns but the volatility is within the range of the three benchmark volatilities. Please refer to Appendix B, Section 8.3 for the reward function and the asset weight plots.

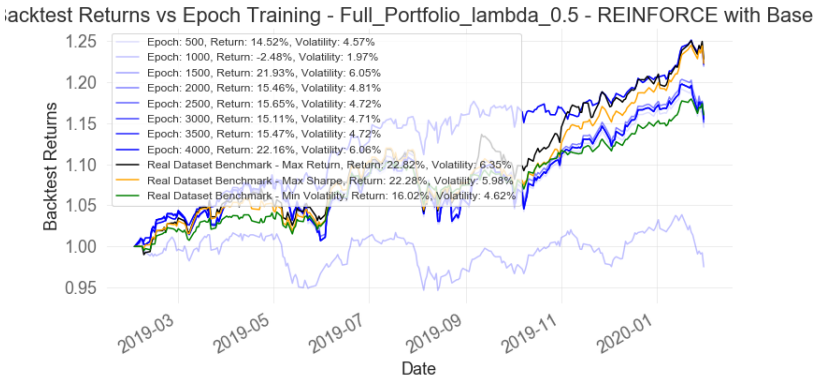

Figure 51: Full Portfolio REINFORCE with Baseline Training Data Backtest Return vs. Benchmark Returns, **$\lambda = 0.5$**

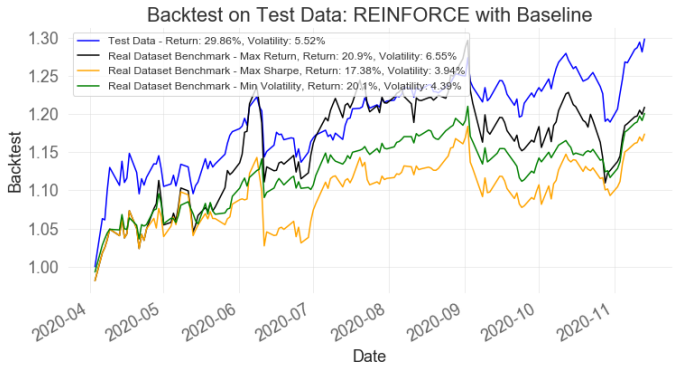

Figure 52: Full Portfolio REINFORCE with Baseline Test Data Backtest Return vs. Benchmark Returns, **$\lambda = 0.5$**

$\boldsymbol{\lambda = 1}$ Case (Maximum Return)

Figure 53 shows that with a maximum return reward function ($\boldsymbol{\lambda = 1}$), the REINFORCE with Baseline algorithm selects a portfolio which maximizes return for the training data set. The training backtest's return (22.0%) is slightly lower than the maximum return benchmark's return (22.82%) and similar to the training data backtest return in the ($\boldsymbol{\lambda = 0.5}$) case. Figure 54 shows that the test benchmark has a higher return (30.03%) than the maximum return benchmark's return (20.9%). Please refer to Appendix B, Section 8.3 for the reward function and the asset weight plots.

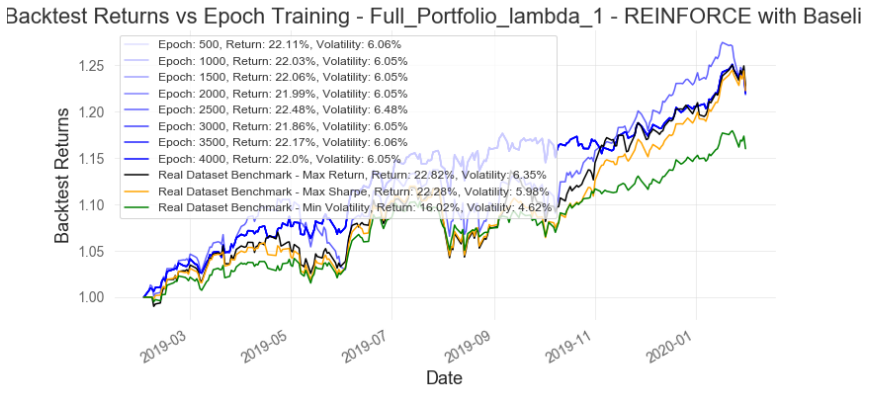

Figure 53: Full Portfolio REINFORCE with Baseline Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

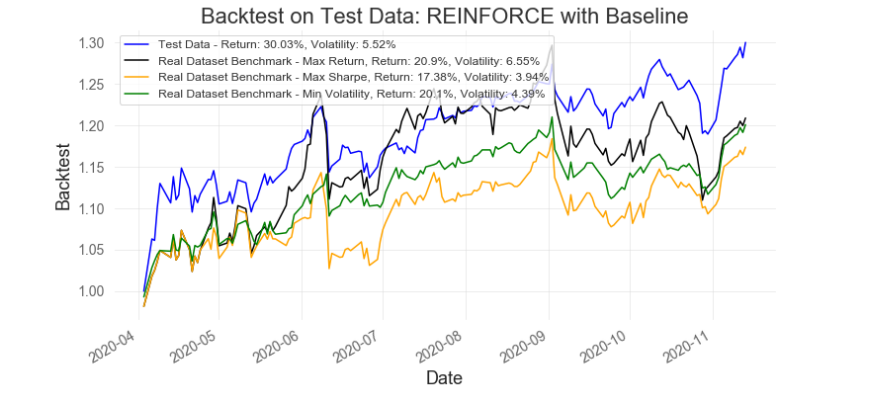

Figure 54: Full Portfolio REINFORCE with Baseline Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

Actor-Critic

$\boldsymbol{\lambda = 0}$ Case (Minimum Volatility)

In Figure 55, the Actor-Critic algorithm converges on a portfolio with a volatility of 3.52%. This is a lower volatility than the minimum volatility benchmark portfolio's volatility (4.62%). Figure 56 shows that the Actor-Critic algorithm also selects the minimum volatility portfolio for the test dataset, with a backtest volatility of 4.28% as compared to the minimum volatility benchmark of 4.39% in the test dataset. Please refer to Appendix B, Section 8.3 for the reward function and the asset weight plots.

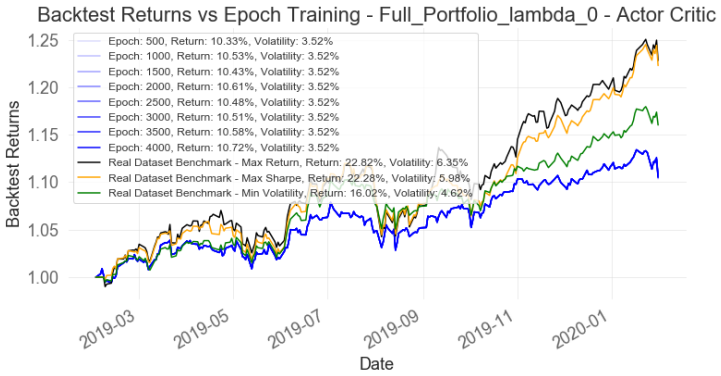

Figure 55: Full Portfolio Actor-Critic Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0}$

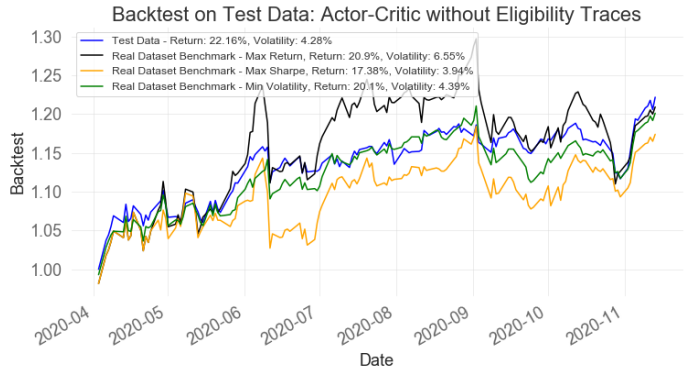

Figure 56: Full Portfolio REINFORCE with Baseline Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0}$

$\lambda = 0.5$ Case

In Figure 57, the Actor-Critic algorithm seeks to balance reward versus risk by selecting a portfolio which has a return of 21.96% and a volatility of 6.05%. This is close to the maximum return and maximum Sharpe benchmarks. Although there is some variation in the backtests over the model run, the backtests across different epochs are similar to each other in terms of return and volatility. In Figure 58, the backtest on the test dataset has a return of 22.16% and a volatility of 4.28%. This is a higher return than the maximum return benchmark and the volatility is lower than the minimum volatility benchmark. Please refer to Appendix B, Section 8.3 for the reward function and the asset weight plots.

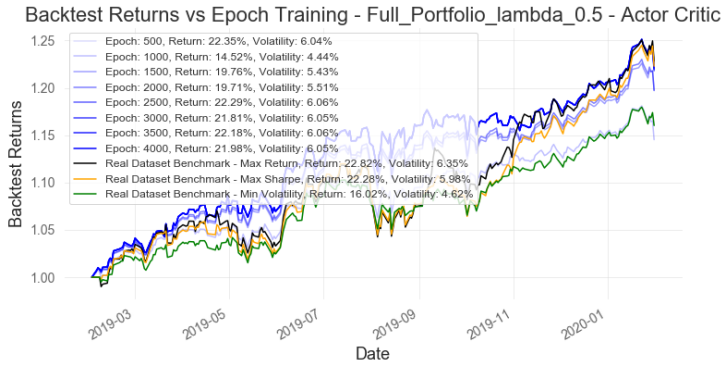

Figure 57: Full Portfolio Actor-Critic Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.5}$

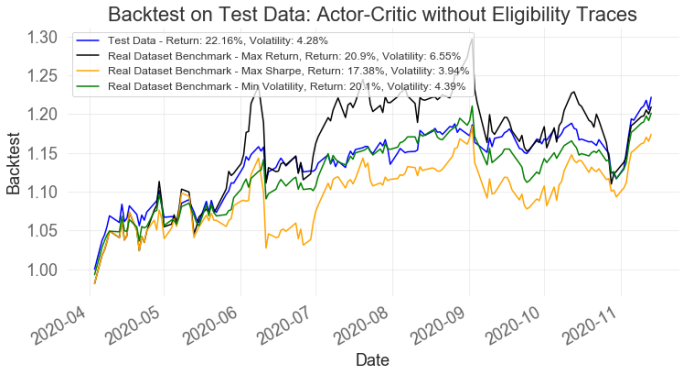

Figure 58: Full Portfolio REINFORCE with Baseline Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.5}$

$\boldsymbol{\lambda = 1}$ Case (Maximum Return)

Figure 59 shows that the Actor-Critic algorithm selects a portfolio with a return of 21.83%. This is slightly lower than the returns of the maximum return and the maximum Sharpe benchmarks. The backtests are fairly stable over the course of the model training. When we perform a backtest on the test dataset in Figure 60, the portfolio has a return of 29.23%, which is higher than the maximum return benchmark of 20.9%. Please refer to Appendix B, Section 8.3 for the reward function and the asset weight plots.

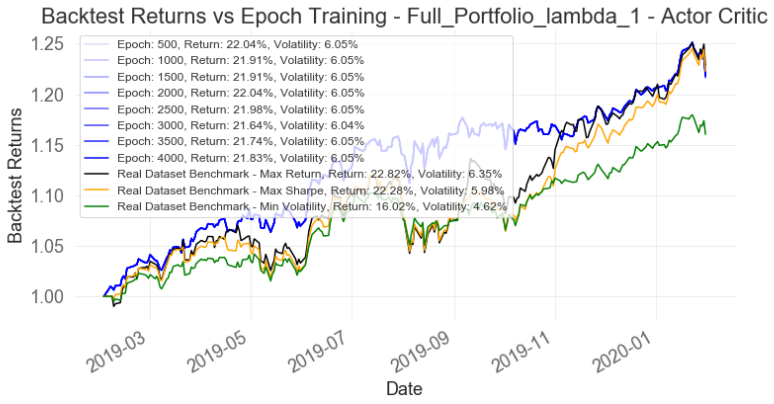

Figure 59: Full Portfolio Actor-Critic Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

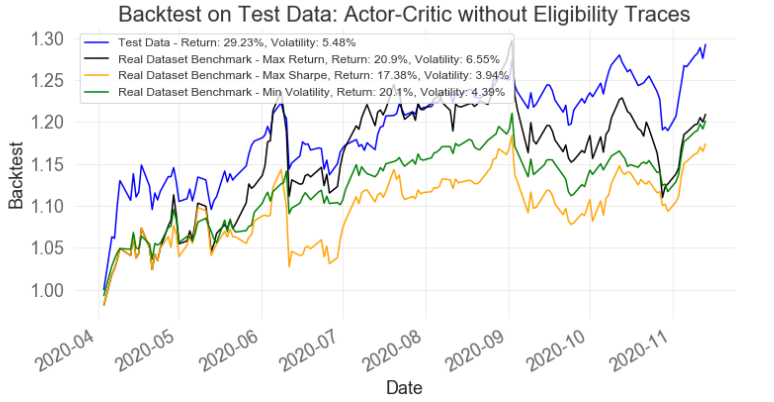

Figure 60: Full Portfolio Actor-Critic Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

Actor-Critic with Eligibility Traces

$\boldsymbol{\lambda = 0}$ Case (Minimum Volatility)

In Figure 61, the Actor-Critic with Eligibility Traces algorithm selects a portfolio which has a training backtest volatility of 3.52% across the majority of the training epochs; this is lower than the minimum volatility benchmark of 4.62%. Figure 62 shows that the Actor-Critic with Eligibility Traces algorithm selects the minimum volatility portfolio for the test dataset (volatility of 4.4% is similar to the volatility of the the minimum volatility benchmark, 4.39%) Please refer to Appendix B, Section 8.3 for the reward function and the asset weight plots.

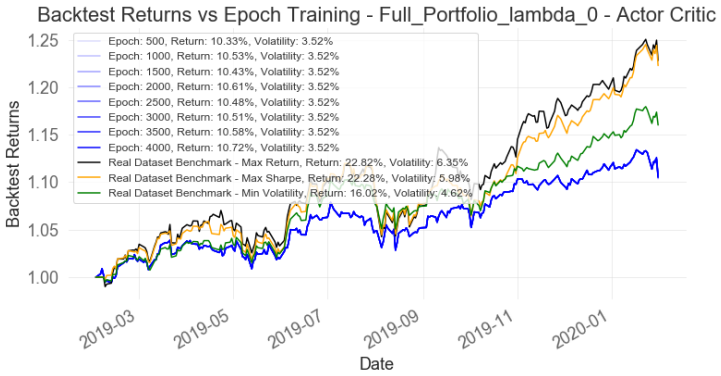

Figure 61: Full Portfolio Actor-Critic Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0}$

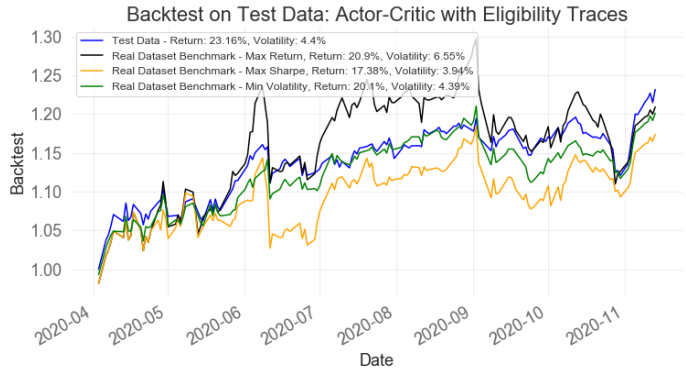

Figure 62: Full Portfolio Actor-Critic with Eligibility Traces Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0}$

$\lambda = 0.5$ Case

In Figure 63, the Actor-Critic with Eligibility Traces algorithm tries to balance risk versus reward and selects a portfolio which appears to be similar to the maximum return or maximum Sharpe benchmarks. The backtest of the training set has a return of 22.26% and a volatility of 6.06%. In Figure 64, we perform a backtest on the test dataset and the return is 30.56% and the volatility is 5.57%; the return is higher than the benchmark returns and the volatility is within the range of the three benchmark volatilities. Please refer to Appendix B, Section 8.3 for the reward function and the asset weight plots.

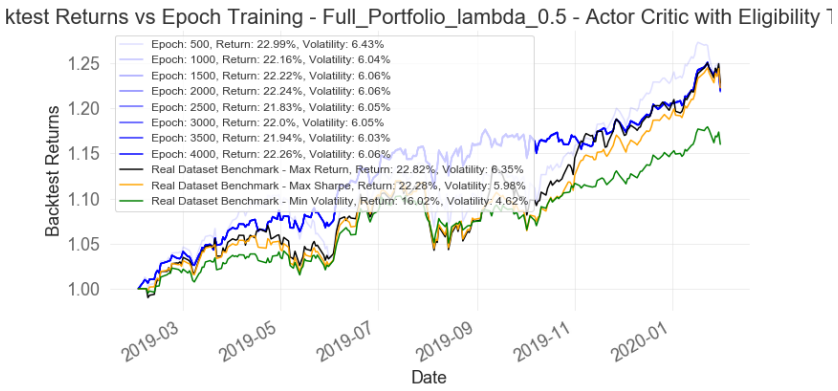

Figure 63: Full Portfolio Actor-Critic with Eligibility Traces Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.5}$

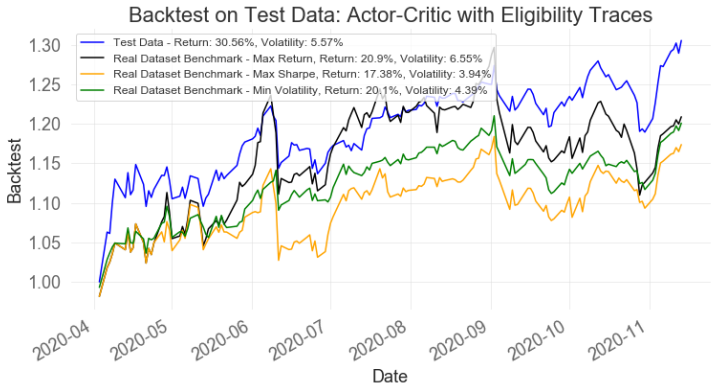

Figure 64: Full Portfolio Actor-Critic with Eligibility Traces Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.5}$

$\boldsymbol{\lambda = 1}$ Case (Maximum Return)

Figure 65 shows that with a max return reward function ($\boldsymbol{\lambda = 1}$), the Actor-Critic with Eligibility Traces algorithm selects a portfolio with a return of 21.83% and a volatility of 6.05%. This return is slightly lower than but close to the maximum return benchmark. The backtest evolution over the course of the model training is stable. In Figure 66, the backtest on the test dataset has a return (30.15%) higher than the maximum return benchmark (20.9%). Please refer to Appendix B, Section 8.3 for the reward function and the asset weight plots.

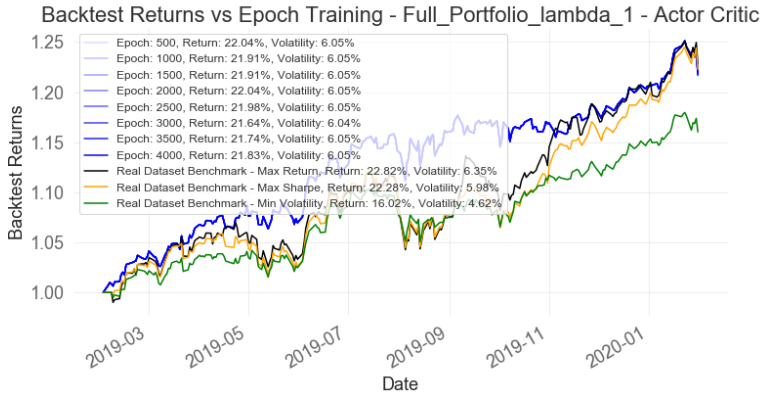

Figure 65: Full Portfolio Actor-Critic with Eligibility Traces Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

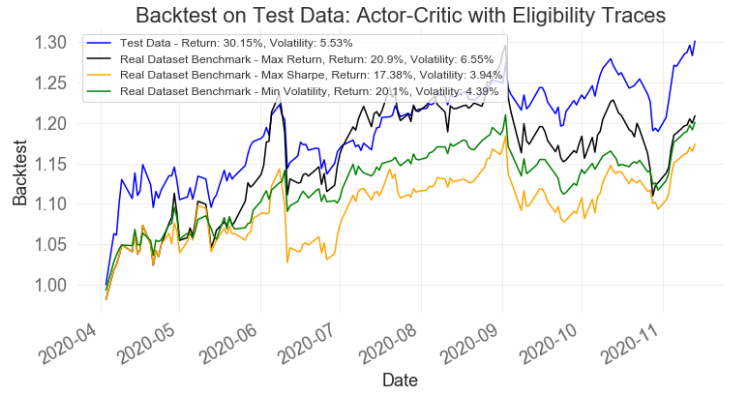

Figure 66: Full Portfolio Actor-Critic with Eligibility Traces Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 1}$

5.3 Statistics

Now that we have shown the results of our dataset, we will be evaluating the statistical components of our dataset. The statistical components will be based on the mentioned performance metrics and observed statistics.[8] To generate our statistical components we will be using the public library of QuantStats. [9]

5.3.1 Real World Dataset Statistics

In Figures 67 and 68, respectively, we evaluate each ETF's statistical components for the training dataset (February 2019 to February 2020) and the testing dataset (March 2020 to November 2020).

	Sharpe Ratio	Sortino Ratio	Calmar Ratio	Max DrawDown
USMV	2.37	3.45	5.86	-0.04
EEMV	0.31	0.49	0.83	-0.07
QUAL	1.70	2.36	3.23	-0.07
SIZE	1.23	1.67	2.16	-0.07
MTUM	1.86	2.63	3.96	-0.06
VLUE	0.80	1.11	1.08	-0.10
EFAV	1.40	1.96	2.42	-0.05

Figure 67: Training Dataset Statistics

[8] Refer to Section 4.1 and 4.2

[9] QuantStats performs portfolio profiling, which allows portfolio managers to understand their performance better by providing them with in-depth analytics and risk metrics. https://github.com/ranaroussi/quantstats

	Sharpe Ratio	Sortino Ratio	Calmar Ratio	Max DrawDown
USMV	1.60	2.59	3.59	-0.14
EEMV	2.21	3.45	8.14	-0.07
QUAL	2.07	3.31	6.30	-0.13
SIZE	2.22	3.54	8.15	-0.13
MTUM	2.07	3.32	6.18	-0.15
VLUE	1.62	2.58	5.25	-0.13
EFAV	1.91	3.32	6.66	-0.08

Figure 68: Test Dataset Statistics

5.3.2 Statistics on Models

In Figures 69 and 70, respectively, we have computed the statistical components from the two asset MTUM-EFAV portfolio algorithm results of using both the training data sets and testing data sets.

		Sharpe Ratio	Sortino Ratio	Calmar Ratio	Max DrawDown
MTUM-EFAV Lambda 0	**REINFORCE**	1.39	1.94	2.42	-0.05
	REINFORCE with Baseline	1.40	1.94	2.42	-0.05
	Actor Critic	1.39	1.93	2.41	-0.05
	Actor Critic with Eligibility Traces	1.36	1.89	2.36	-0.05
MTUM-EFAV Lambda 0.5	**REINFORCE**	1.38	1.92	2.40	-0.05
	REINFORCE with Baseline	1.73	2.43	3.43	-0.05
	Actor Critic	1.70	2.38	3.32	-0.05
	Actor Critic with Eligibility Traces	1.48	2.07	2.66	-0.05
MTUM-EFAV Lambda 1	**REINFORCE**	1.86	2.63	3.94	-0.06
	REINFORCE with Baseline	1.87	2.65	3.97	-0.06
	Actor Critic	1.87	2.65	3.96	-0.06
	Actor Critic with Eligibility Traces	1.88	2.66	3.98	-0.06

Figure 69: MTUM-EFAV Portfolio Training Data Set Statistics

		Sharpe Ratio	Sortino Ratio	Calmar Ratio	Max DrawDown
MTUM-EFAV Lambda 0	**REINFORCE**	1.87	2.88	5.60	-0.06
	REINFORCE with Baseline	1.85	2.85	5.52	-0.06
	Actor Critic	1.88	2.90	5.61	-0.06
	Actor Critic with Eligibility Traces	1.82	2.79	5.40	-0.06
MTUM-EFAV Lambda 0.5	**REINFORCE**	1.80	2.76	5.32	-0.06
	REINFORCE with Baseline	2.61	4.11	8.33	-0.08
	Actor Critic	2.41	3.72	7.19	-0.07
	Actor Critic with Eligibility Traces	2.08	3.20	6.11	-0.06
MTUM-EFAV Lambda 1	**REINFORCE**	2.83	4.40	9.79	-0.10
	REINFORCE with Baseline	2.86	4.46	9.94	-0.10
	Actor Critic	2.83	4.39	9.77	-0.10
	Actor Critic with Eligibility Traces	2.91	4.52	10.14	-0.10

Figure 70: MTUM-EFAV Portfolio Testing Data Set Statistics

In Figures 71 and 72, respectively, we have computed the statistical components from the full portfolio algorithm results of using both the training data sets and testing data sets.

		Sharpe Ratio	Sortino Ratio	Calmar Ratio	Max DrawDown
Full Portfolio Lambda 0	**REINFORCE**	1.33	1.84	2.31	-0.05
	REINFORCE with Baseline	1.81	2.55	3.82	-0.06
	Actor Critic	1.33	1.83	2.31	-0.05
	Actor Critic with Eligibility Traces	1.59	2.22	3.04	-0.04
Full Portfolio Lambda 0.5	**REINFORCE**	2.26	3.26	5.59	-0.04
	REINFORCE with Baseline	2.24	3.23	5.55	-0.04
	Actor Critic	1.29	1.79	2.38	-0.04
	Actor Critic with Eligibility Traces	2.25	3.25	5.58	-0.04
Full Portfolio Lambda 1	**REINFORCE**	1.34	1.85	2.34	-0.05
	REINFORCE with Baseline	1.19	1.42	2.30	-0.05
	Actor Critic	2.21	3.19	5.47	-0.04
	Actor Critic with Eligibility Traces	2.21	3.18	5.48	-0.04

Figure 71: Full Portfolio Training Data Set Statistics

		Sharpe Ratio	Sortino Ratio	Calmar Ratio	Max DrawDown
Full Portfolio Lambda 0	**REINFORCE**	2.10	3.25	6.45	-0.06
	REINFORCE with Baseline	2.85	4.33	9.58	-0.10
	Actor Critic	2.08	3.23	6.40	-0.06
	Actor Critic with Eligibility Traces	2.15	3.30	6.44	-0.06
Full Portfolio Lambda 0.5	**REINFORCE**	2.17	3.39	7.71	-0.07
	REINFORCE with Baseline	2.14	3.34	7.54	-0.07
	Actor Critic	2.06	3.16	6.24	-0.06
	Actor Critic with Eligibility Traces	2.17	3.40	7.73	-0.07
Full Portfolio Lambda 1	**REINFORCE**	2.11	3.28	6.53	-0.06
	REINFORCE with Baseline	2.69	4.40	7.81	-0.05
	Actor Critic	2.10	3.28	7.37	-0.07
	Actor Critic with Eligibility Traces	2.15	3.36	7.62	-0.07

Figure 72: Full Portfolio Testing Data Set Statistics

6 Conclusion

Applying Reinforcement Learning to portfolio management was the objective of this capstone project. We determined that this could be accomplished by using Policy Gradient Methods such as REINFORCE, REINFORCE with baseline, Actor-Critic, and Actor-Critic with Eligibility Traces to build a model-free agent which selects portfolio weights.

We used a reward function which can be balanced between risk and reward by using a risk aversion parameter, $\boldsymbol{\lambda}$. Using a control dataset of simulated data, we were able to demonstrate model convergence for the four Policy Gradient Methods that we tried. After detrending our real-world data and demeaning the returns, we were also able to demonstrate convergence for all four Policy Gradient methods in a portfolio of two real-world assets. Finally, we tested our Policy Gradient Methods on a full portfolio of seven ETFs and the Policy Gradient Methods were able to pick between a minimum volatility portfolio or a maximum return portfolio, depending on the $\boldsymbol{\lambda}$ parameter specified.

In order to validate our models, we split our dataset into a training dataset and testing dataset. In all four Policy Gradient Methods with various risk aversion parameters, the testing dataset resulted in either meeting the portfolio benchmarks of the training dataset or exceeding those benchmarks.

Our statistical evaluation of our real-world dataset gave us insight to the performance of the ETF's in our portfolio by providing in-depth analytics and risk metrics. Our evaluation consisted of both two-real world assets model statistics and the full real world portfolio model statistics. Based on the Sharpe Ratio, Sortino Ratio, Calmar Ratio, and Max Drawdown, the ETFs used in our real-world dataset provided us a diverse distribution of assets to be used for our models and therefore provided us with an opportunity to explore Reinforcement Learning in portfolio management.

For more information regarding our capstone project, please refer to:
Github Repository: https://github.com/nikatpatel/epsilon-greedy-quants
Project Website: https://nikatpatel.github.io/epsilon-greedy-quants/

7 Appendix A: Installation Requirements

The complete list of libraries used for the project can be found in the file requirements.txt in the repository. While most of the libraries are standard and can be installed directly through pip

```
pip install -r requirements.txt
```

There are a few that require special care. Below are further details on their installation.

1. Pytorch : Use the OS matrix described here: https://pytorch.org/get-started/locally/
2. Spinning Up : Needs to be built from source, complete details can be found here: https://spinningup.openai.com/en/latest/user/installation.html
3. PyportfolioOpt: Windows users may require extra steps to install cvxopt, details can be found here: https://pyportfolioopt.readthedocs.io/en/latest/UserGuide.html and https://cvxopt.org/install/
4. QuantStats : Needs to be built from source, complete details can be found here: https://github.com/ranaroussi/quantstats
5. We have also included a DataFeatures class that can generate several technical indicators using the library "Talib". For details on installation please follow: https://mrjbq7.github.io/ta-lib/install.html

8 Appendix B: Supplemental Data and Results

8.1 ETF Detrending

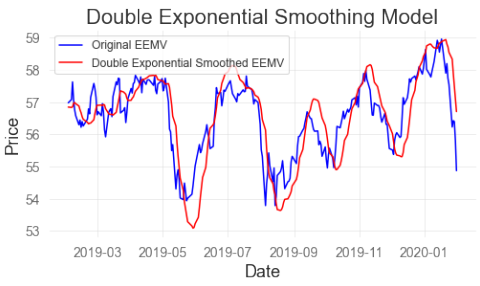

Figure 73: Double Exponential Smoothing of EEMV time series

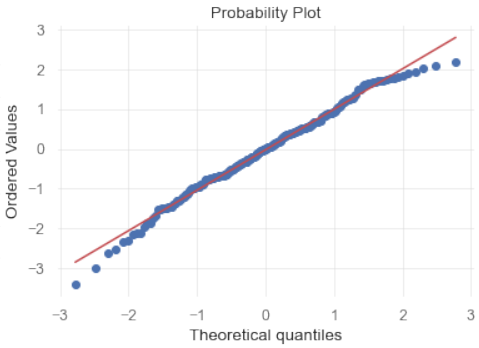

Figure 74: QQ Plot of EEMV detrended residuals

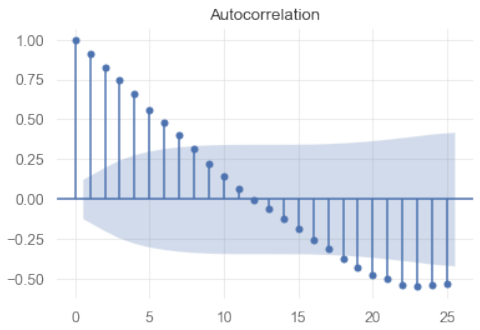

Figure 75: ACF Plot of detrended EEMV residuals

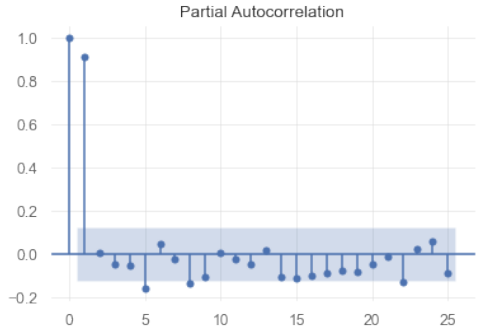

Figure 76: PACF Plot of detrended EEMV residuals

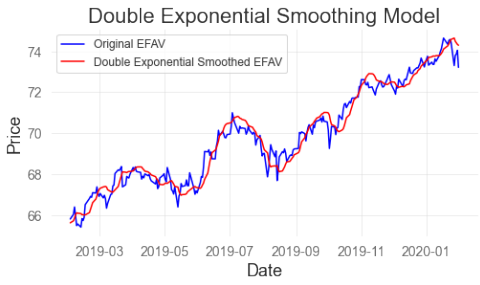

Figure 77: Double Exponential Smoothing of EFAV time series

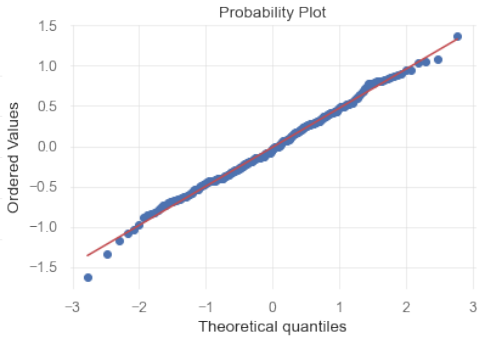

Figure 78: QQ Plot of EFAV detrended residuals

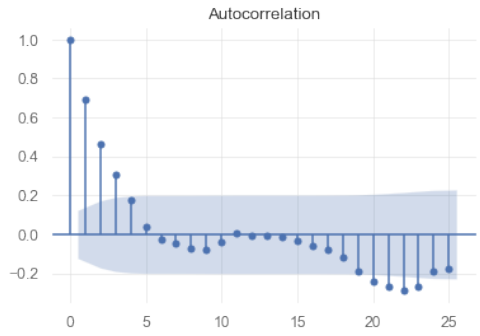

Figure 79: ACF Plot of detrended EFAV residuals

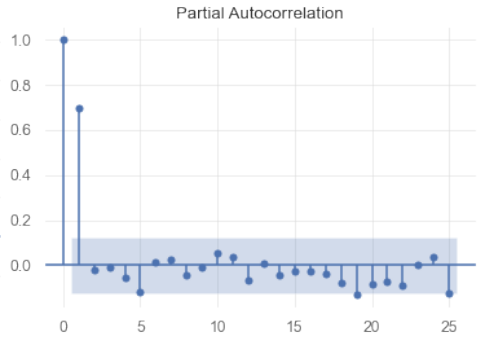

Figure 80: PACF Plot of detrended EFAV residuals

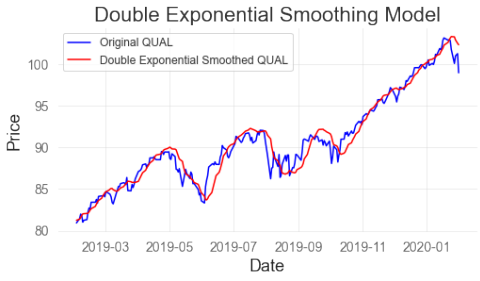

Figure 81: Double Exponential Smoothing of QUAL time series

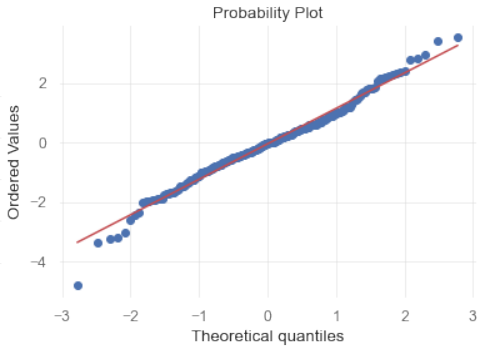

Figure 82: QQ Plot of QUAL detrended residuals

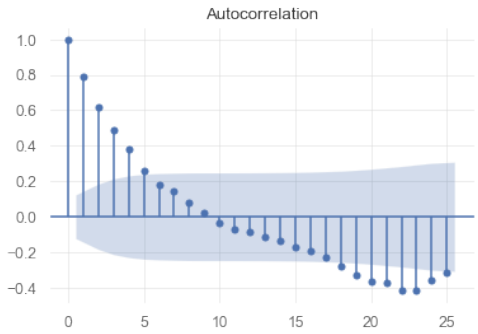

Figure 83: ACF Plot of detrended QUAL residuals

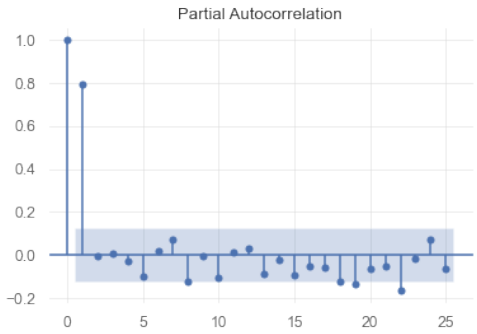

Figure 84: PACF Plot of detrended QUAL residuals

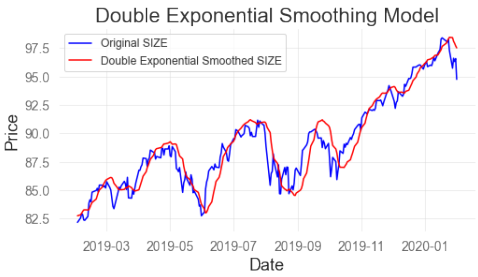

Figure 85: Double Exponential Smoothing of SIZE time series

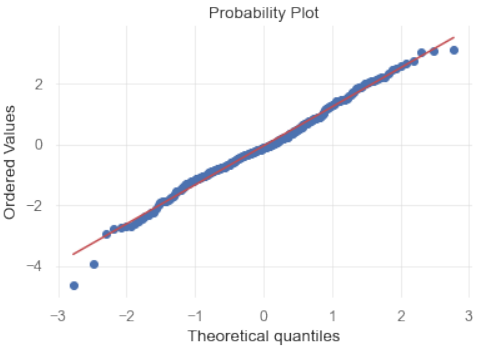

Figure 86: QQ Plot of SIZE detrended residuals

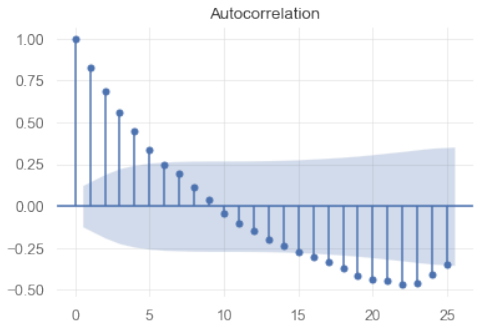

Figure 87: ACF Plot of detrended SIZE residuals

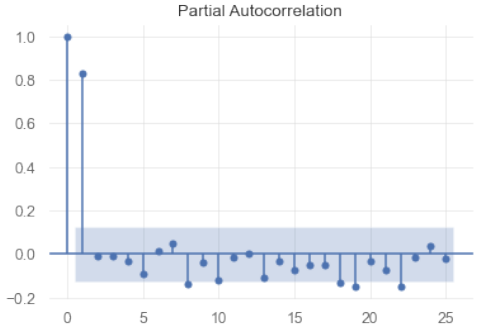

Figure 88: PACF Plot of detrended SIZE residuals

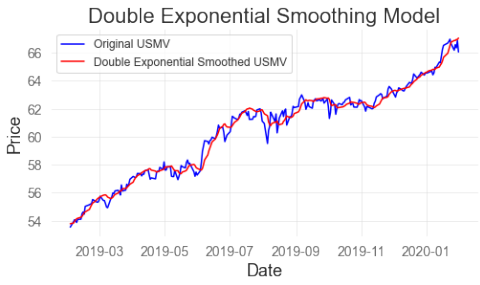

Figure 89: Double Exponential Smoothing of USMV time series

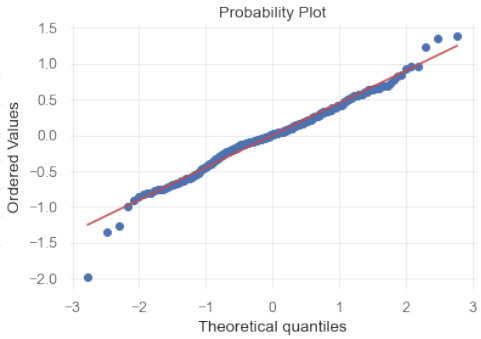

Figure 90: QQ Plot of USMV detrended residuals

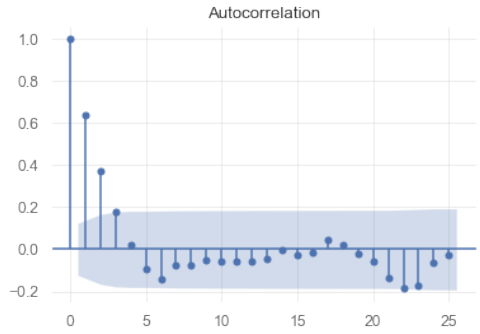

Figure 91: ACF Plot of detrended USMV residuals

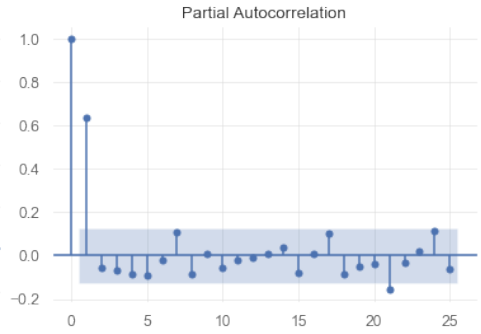

Figure 92: PACF Plot of detrended USMV residuals

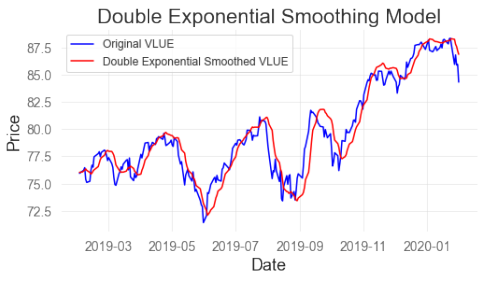

Figure 93: Double Exponential Smoothing of VLUE time series

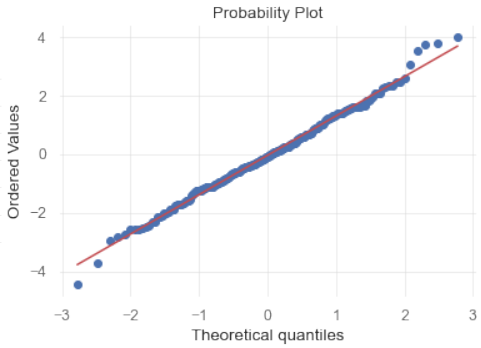

Figure 94: QQ Plot of VLUE detrended residuals

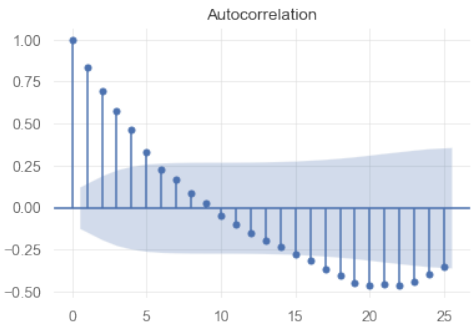

Figure 95: ACF Plot of detrended VLUE residuals

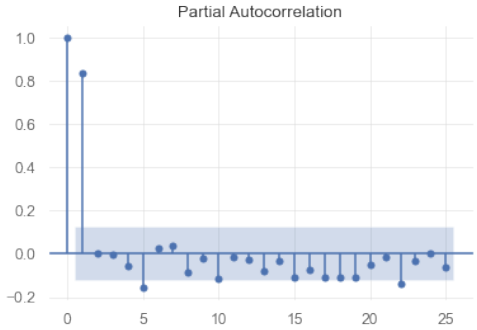

Figure 96: PACF Plot of detrended VLUE residuals

8.2 Two Asset Results

$\boldsymbol{\lambda = 0}$ Case (Minimum Volatility)

REINFORCE with Baseline

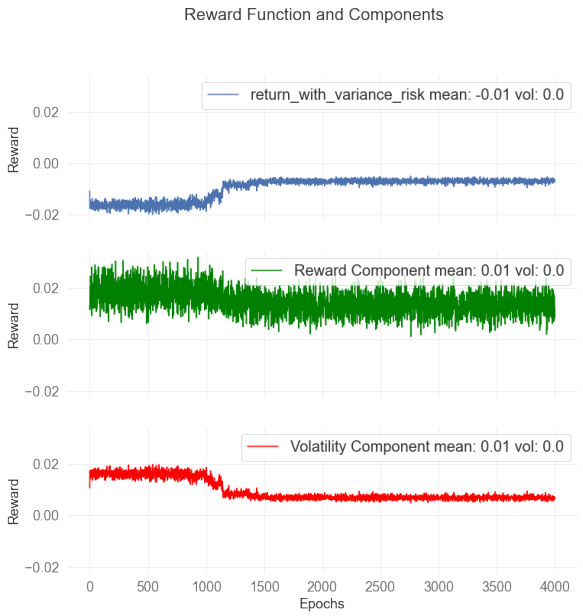

Figure 97: Two Asset REINFORCE with Baseline Reward Function $\boldsymbol{\lambda = 0}$

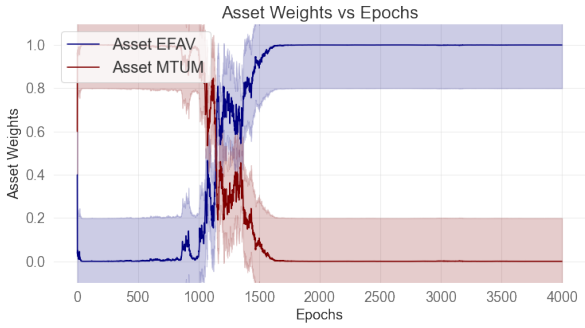

Figure 98: Two Asset REINFORCE with Baseline Asset Weights $\boldsymbol{\lambda = 0}$

Actor-Critic

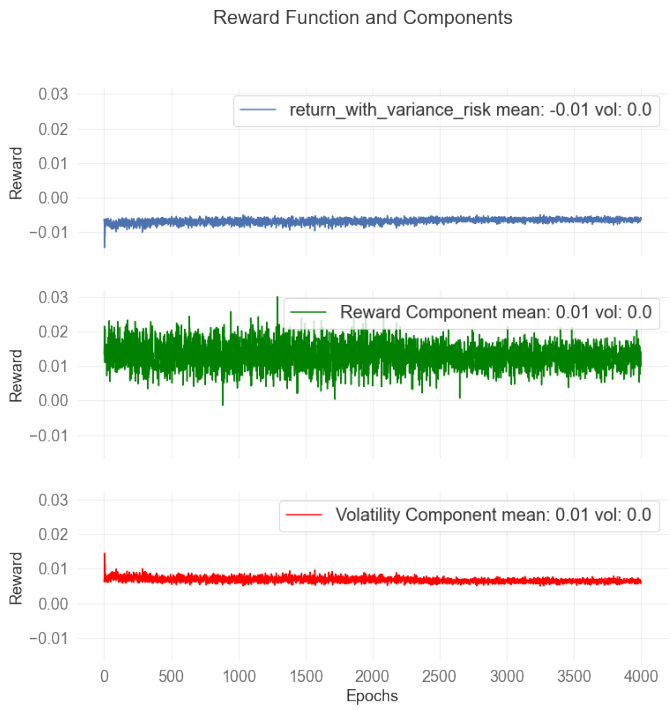

Figure 99: Two Asset Actor-Critic Reward Function $\boldsymbol{\lambda = 0}$

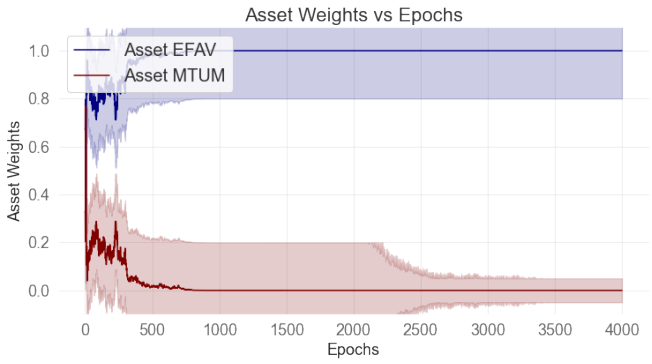

Figure 100: Two Asset Actor-Critic Asset Weights $\boldsymbol{\lambda = 0}$

Actor-Critic with Eligibility Traces

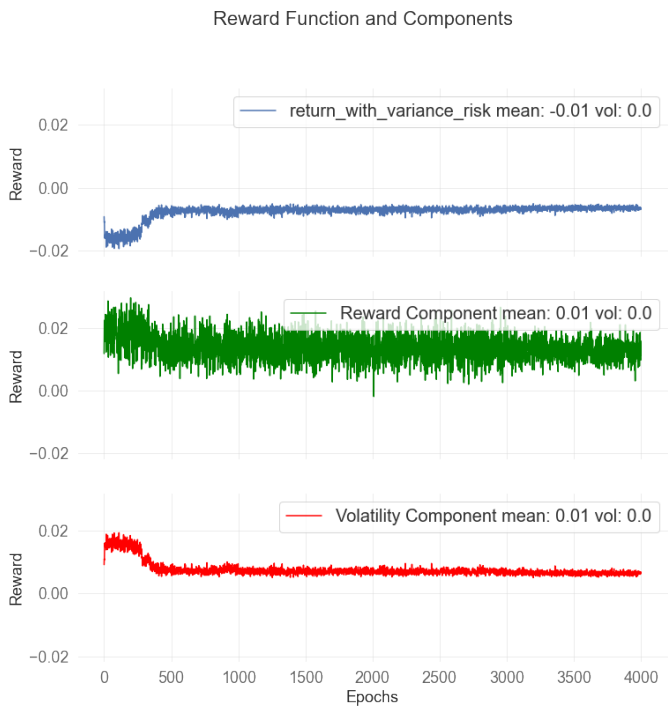

Figure 101: Two Asset Actor-Critic with Eligibility Traces Reward Function $\boldsymbol{\lambda = 0}$

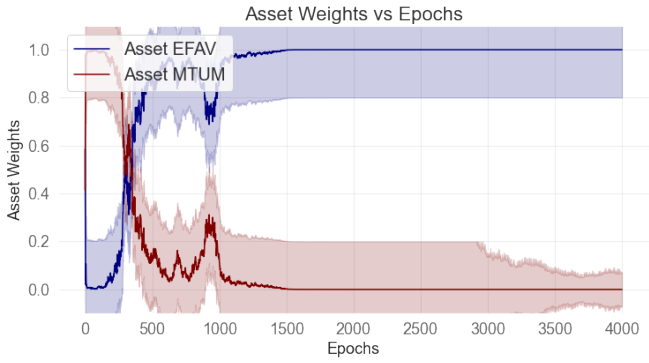

Figure 102: Two Asset Actor-Critic with Eligibility Traces Asset Weights $\boldsymbol{\lambda = 0}$

$\boldsymbol{\lambda = 0.2}$ Case

REINFORCE

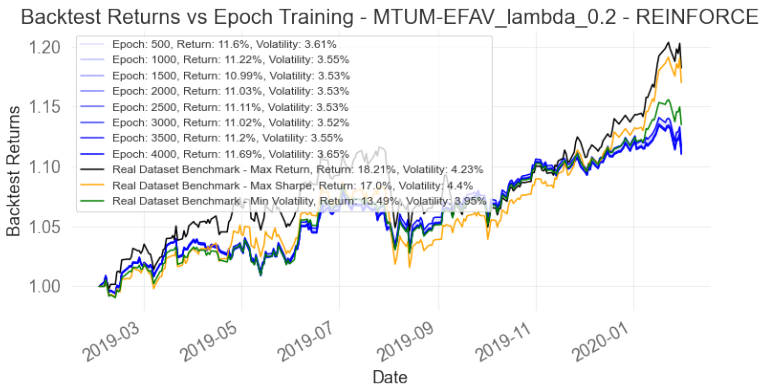

Figure 103: Two Asset REINFORCE Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.2}$

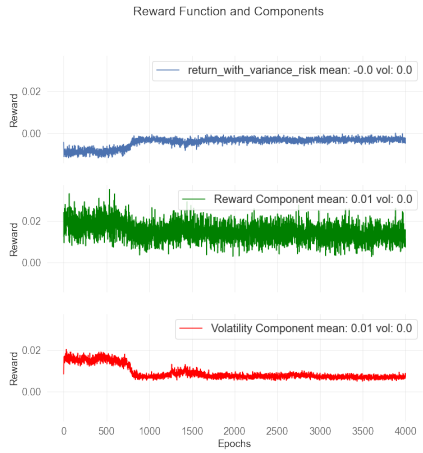

Figure 104: Two Asset REINFORCE Reward Function $\boldsymbol{\lambda = 0.2}$

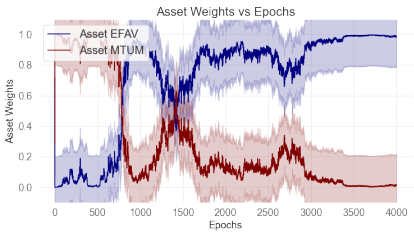

Figure 105: Two Asset REINFORCE Asset Weights $\boldsymbol{\lambda = 0.2}$

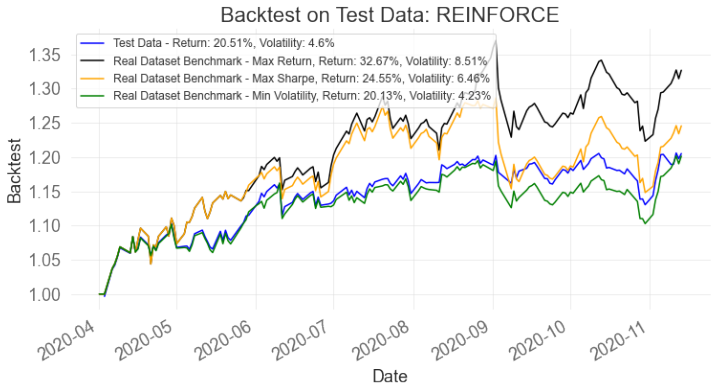

Figure 106: Two Asset REINFORCE Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.2}$

REINFORCE with Baseline

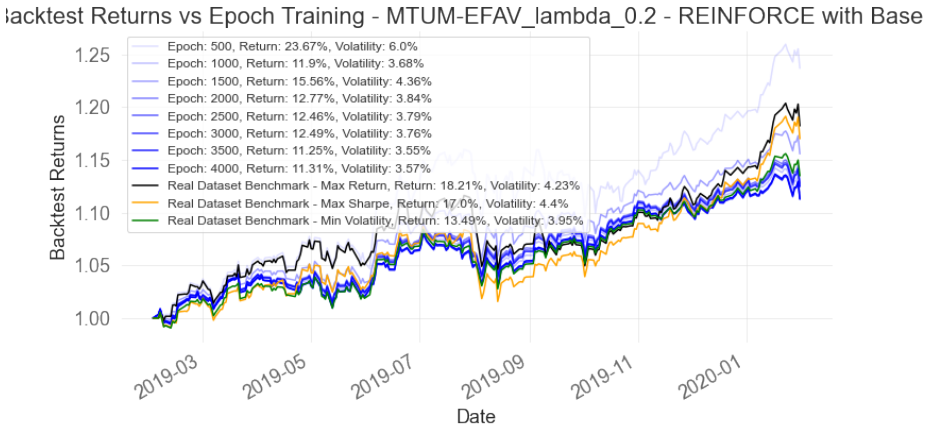

Figure 107: Two Asset REINFORCE Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.2}$

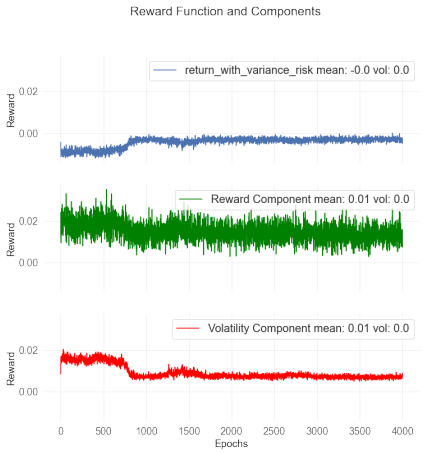

Figure 108: Two Asset REINFORCE with Baseline Reward Function $\boldsymbol{\lambda} = \mathbf{0.2}$

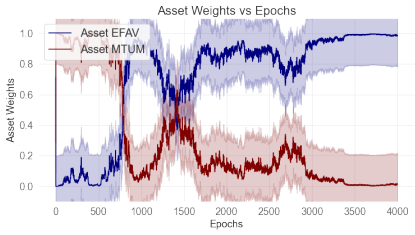

Figure 109: Two Asset REINFORCE with Baseline Asset Weights $\boldsymbol{\lambda} = \mathbf{0.2}$

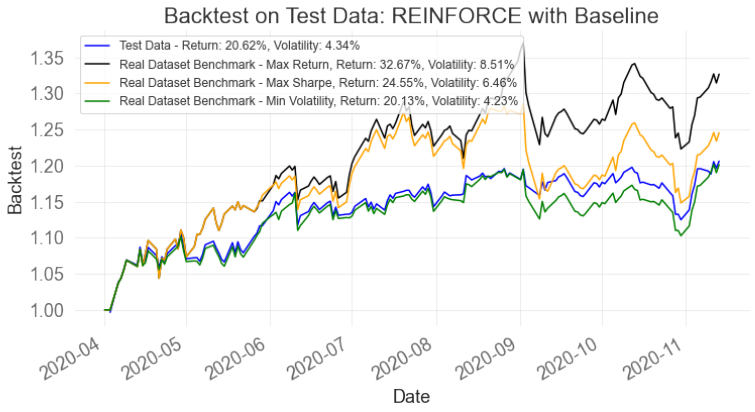

Figure 110: Two Asset REINFORCE with Baseline Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda} = \mathbf{0.2}$

Actor-Critic

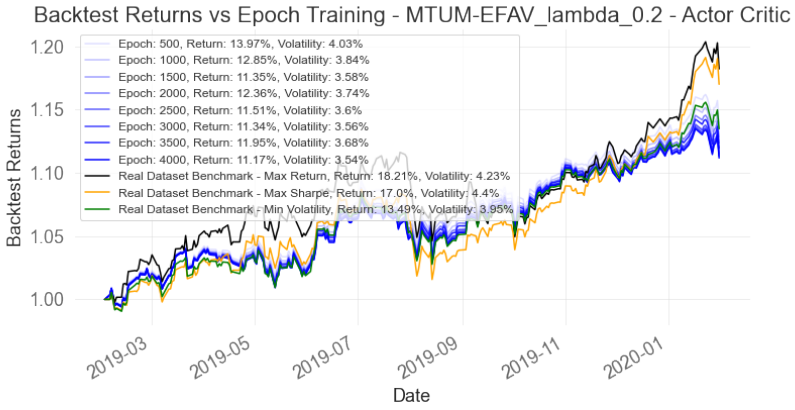

Figure 111: Two Asset Actor-Critic Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.2}$

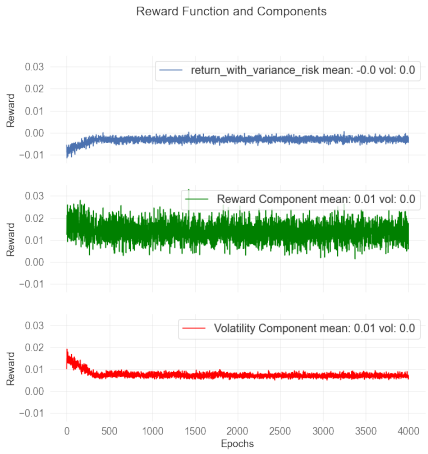

Figure 112: Two Asset Actor-Critic Reward Function $\boldsymbol{\lambda = 0.2}$

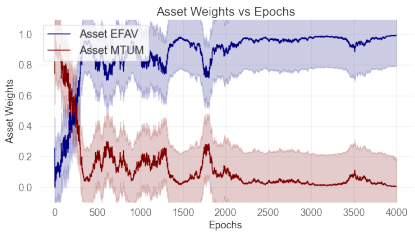

Figure 113: Two Asset Actor-Critic Asset Weights $\boldsymbol{\lambda = 0.2}$

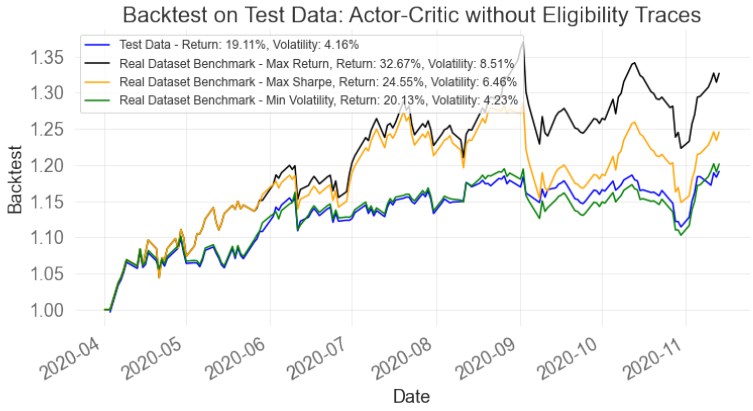

Figure 114: Two Asset Actor-Critic Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.2}$

Actor-Critic with Eligibility Traces

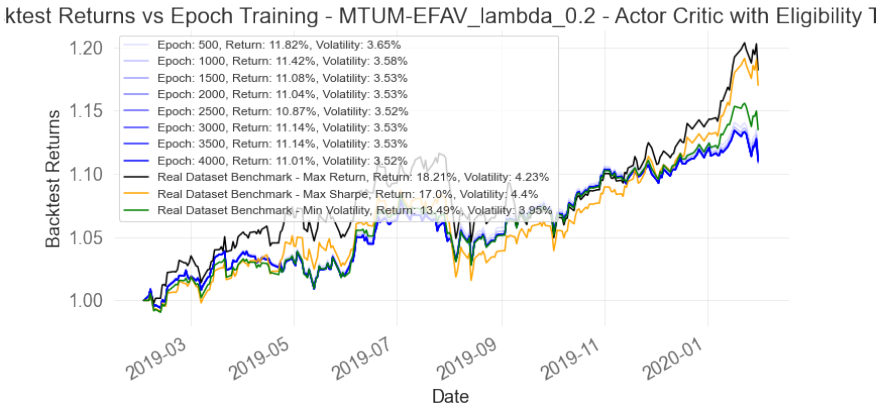

Figure 115: Two Asset Actor-Critic with Eligibility Traces Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.2}$

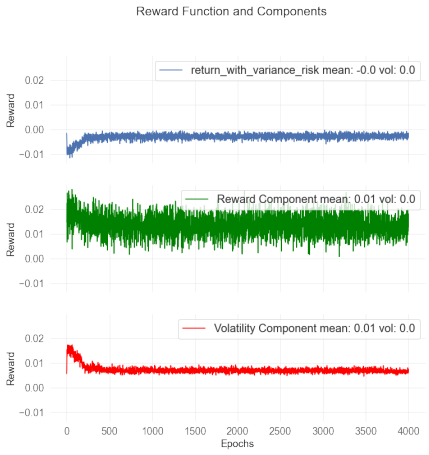

Figure 116: Two Asset Actor-Critic with Eligibility Traces Reward Function $\boldsymbol{\lambda = 0.2}$

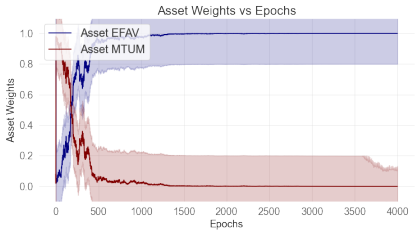

Figure 117: Two Asset Actor-Critic with Eligibility Traces Asset Weights $\boldsymbol{\lambda = 0.2}$

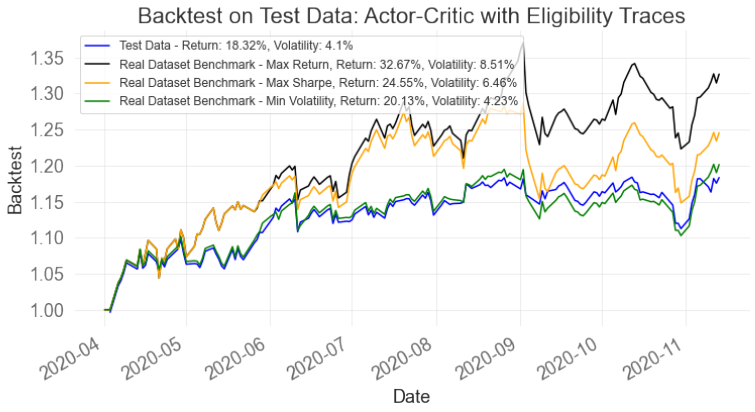

Figure 118: Two Asset Actor-Critic with Eligibility Traces Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.2}$

$\boldsymbol{\lambda = 0.5}$ Case

REINFORCE

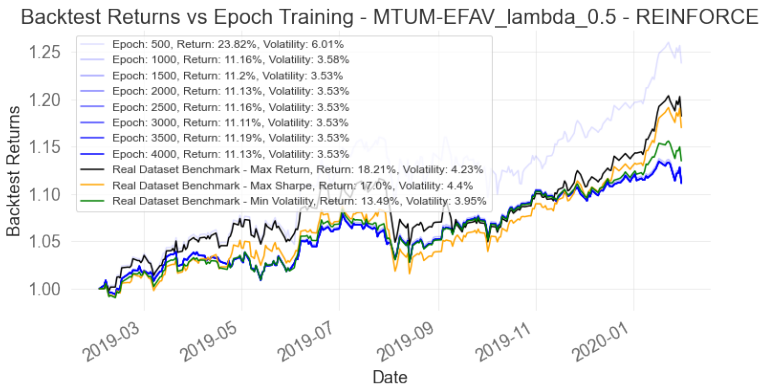

Figure 119: Two Asset REINFORCE Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.5}$

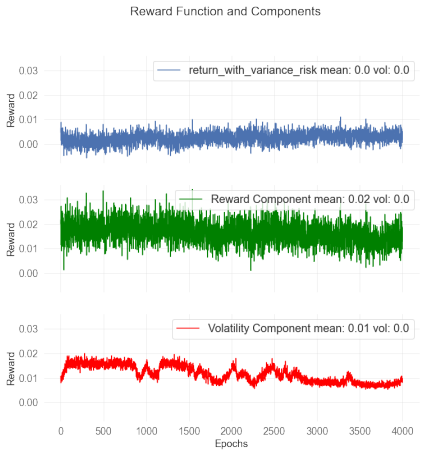

Figure 120: Two Asset REINFORCE Reward Function $\boldsymbol{\lambda = 0.5}$

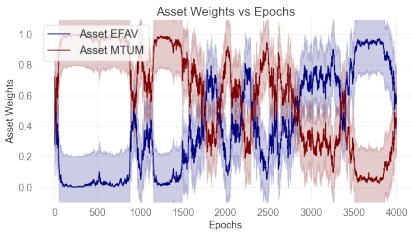

Figure 121: Two Asset REINFORCE Asset Weights $\boldsymbol{\lambda = 0.5}$

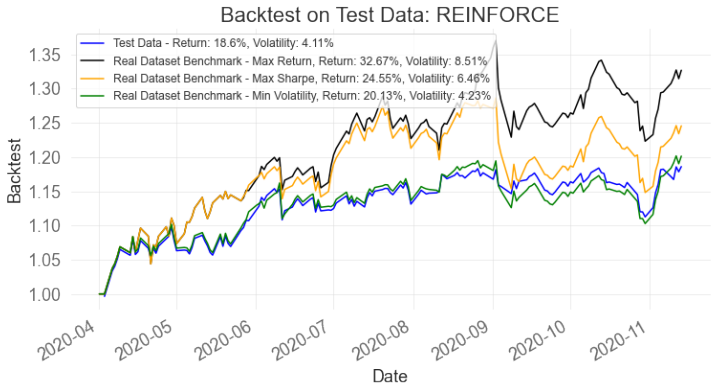

Figure 122: Two Asset REINFORCE Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.5}$

REINFORCE with Baseline

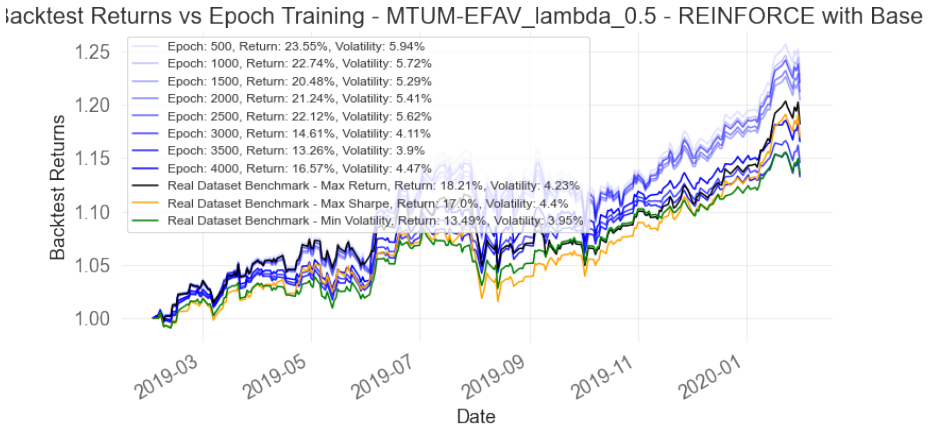

Figure 123: Two Asset REINFORCE Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.5}$

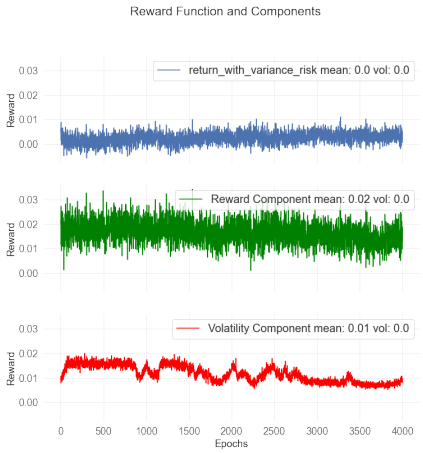

Figure 124: Two Asset REINFORCE with Baseline Reward Function $\boldsymbol{\lambda = 0.5}$

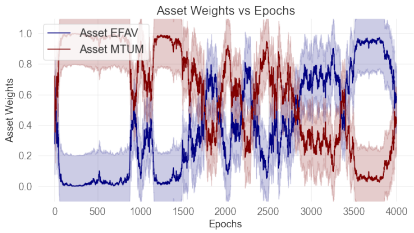

Figure 125: Two Asset REINFORCE with Baseline Asset Weights $\boldsymbol{\lambda = 0.5}$

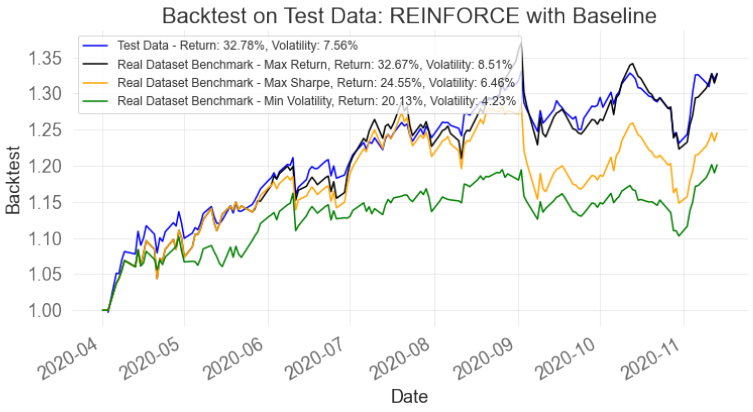

Figure 126: Two Asset REINFORCE with Baseline Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.5}$

Actor-Critic

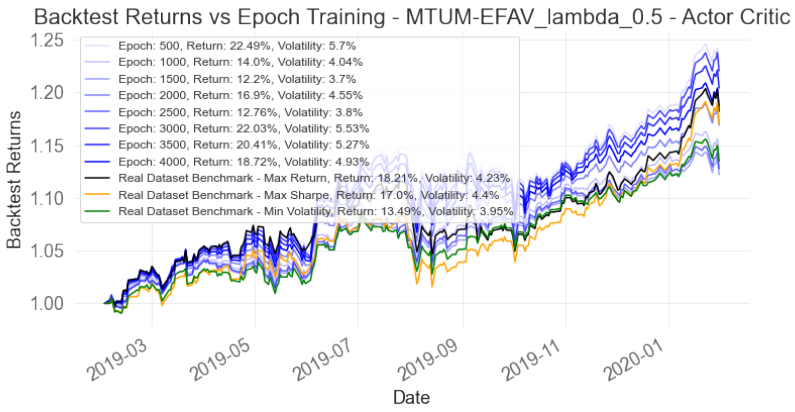

Figure 127: Two Asset Actor-Critic Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.5}$

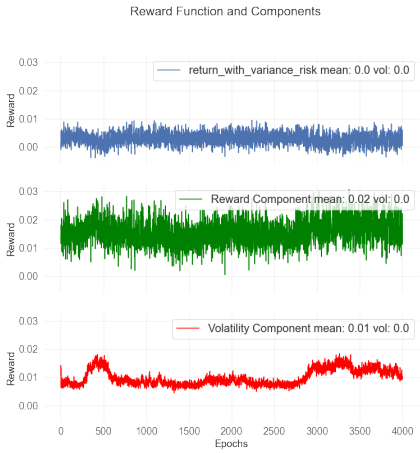

Figure 128: Two Asset Actor-Critic Reward Function $\boldsymbol{\lambda = 0.5}$

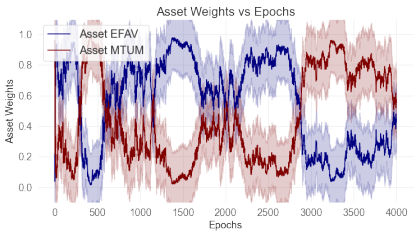

Figure 129: Two Asset Actor-Critic Asset Weights $\boldsymbol{\lambda = 0.5}$

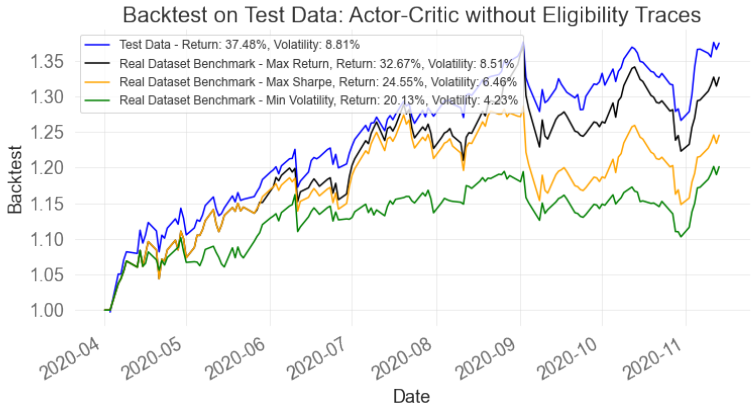

Figure 130: Two Asset Actor-Critic Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.5}$

Actor-Critic with Eligibility Traces

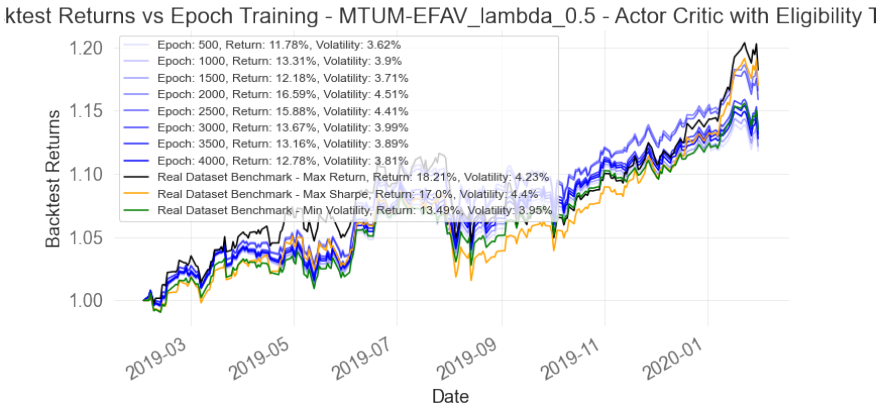

Figure 131: Two Asset Actor-Critic with Eligibility Traces Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.5}$

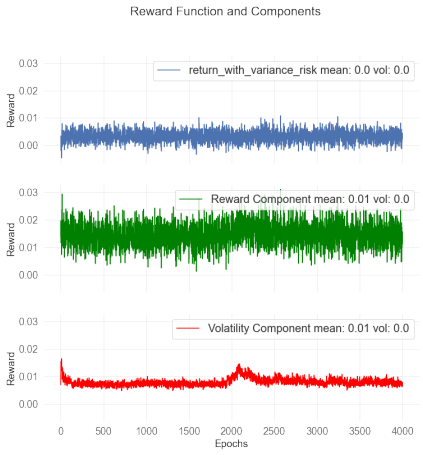

Figure 132: Two Asset Actor-Critic with Eligibility Traces Reward Function $\boldsymbol{\lambda = 0.5}$

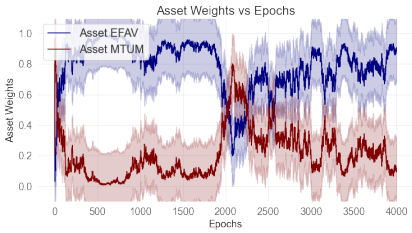

Figure 133: Two Asset Actor-Critic with Eligibility Traces Asset Weights $\boldsymbol{\lambda = 0.5}$

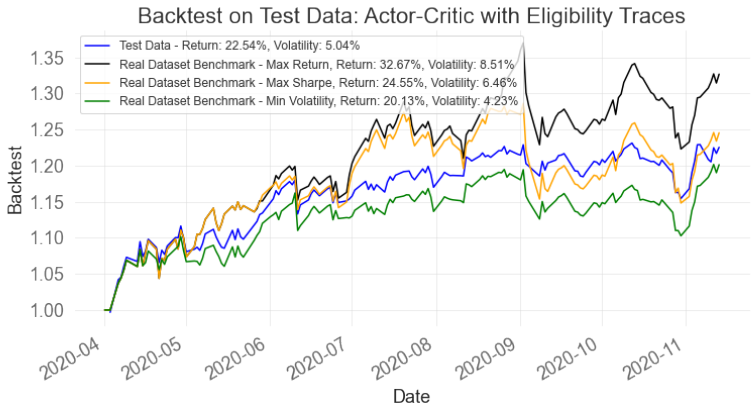

Figure 134: Two Asset Actor-Critic with Eligibility Traces Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.5}$

$\boldsymbol{\lambda = 0.8}$ Case

REINFORCE

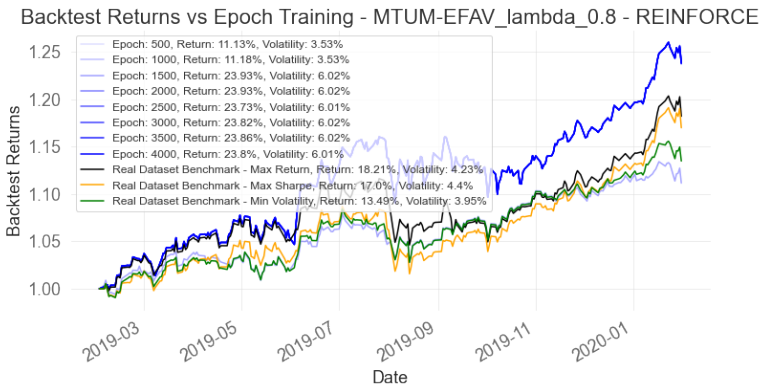

Figure 135: Two Asset REINFORCE Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.8}$

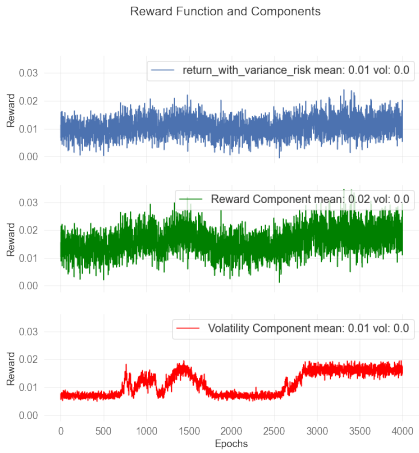

Figure 136: Two Asset REINFORCE Reward Function $\boldsymbol{\lambda = 0.8}$

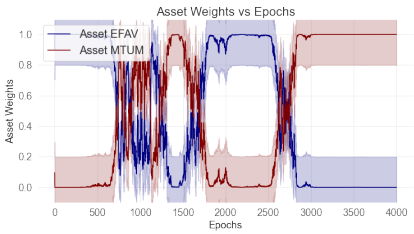

Figure 137: Two Asset REINFORCE Asset Weights $\boldsymbol{\lambda = 0.8}$

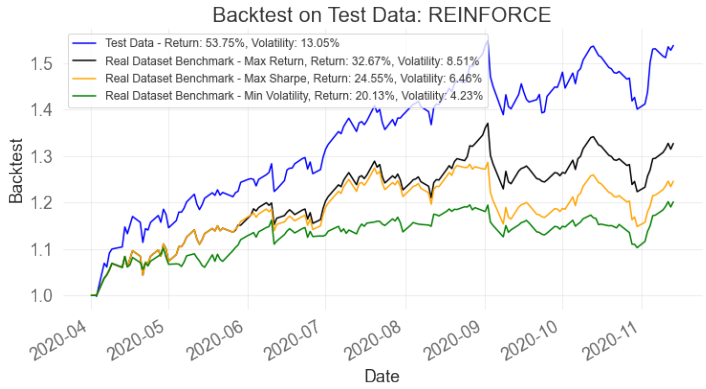

Figure 138: Two Asset REINFORCE Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.8}$

REINFORCE with Baseline

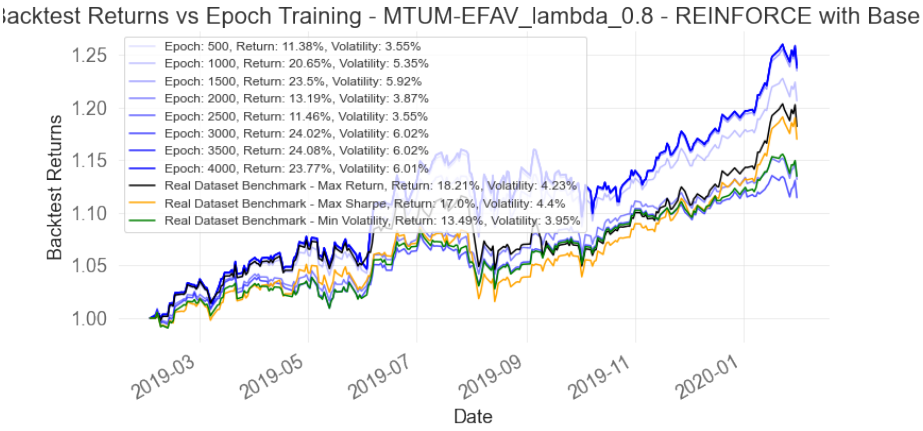

Figure 139: Two Asset REINFORCE Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.8}$

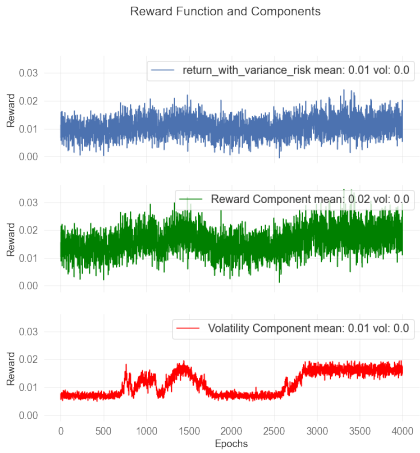

Figure 140: Two Asset REINFORCE with Baseline Reward Function $\boldsymbol{\lambda} = \mathbf{0.8}$

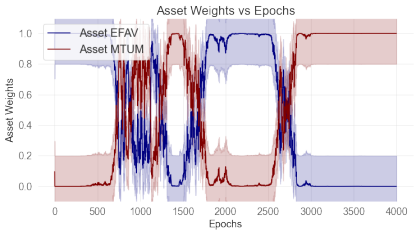

Figure 141: Two Asset REINFORCE with Baseline Asset Weights $\boldsymbol{\lambda} = \mathbf{0.8}$

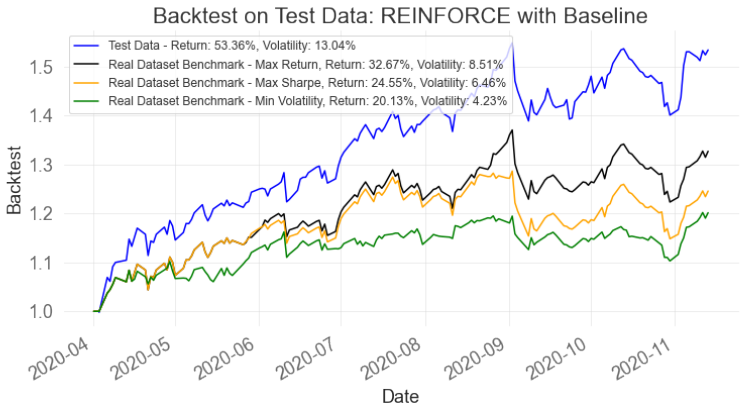

Figure 142: Two Asset REINFORCE with Baseline Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda} = \mathbf{0.8}$

Actor-Critic

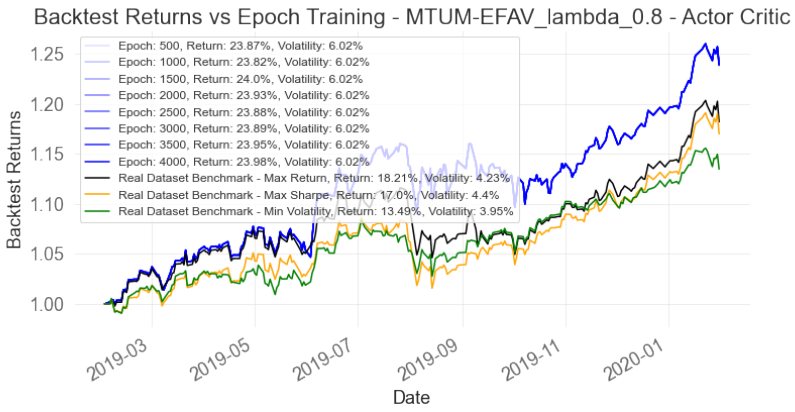

Figure 143: Two Asset Actor-Critic Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.8}$

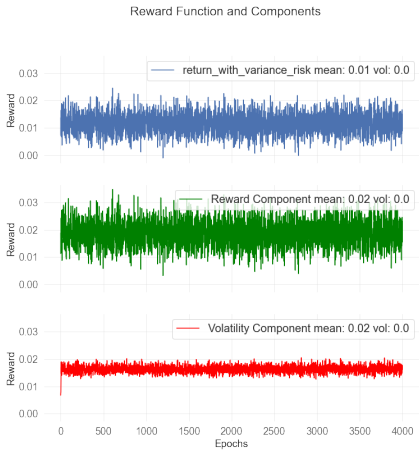

Figure 144: Two Asset Actor-Critic Reward Function $\boldsymbol{\lambda = 0.8}$

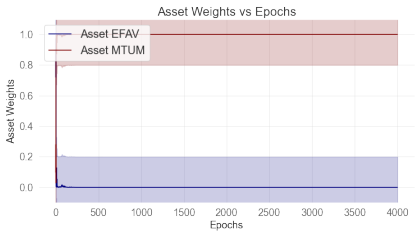

Figure 145: Two Asset Actor-Critic Asset Weights $\boldsymbol{\lambda = 0.8}$

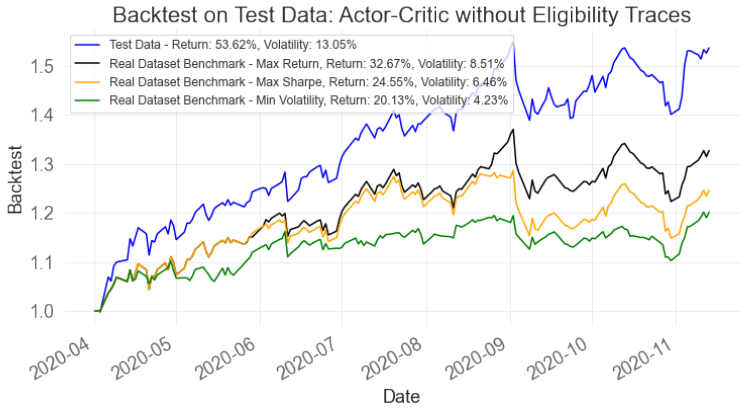

Figure 146: Two Asset Actor-Critic Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.8}$

Actor-Critic with Eligibility Traces

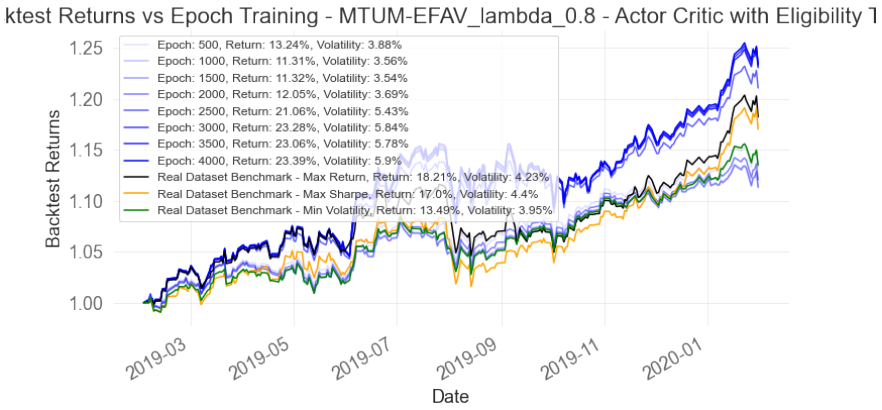

Figure 147: Two Asset Actor-Critic with Eligibility Traces Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.8}$

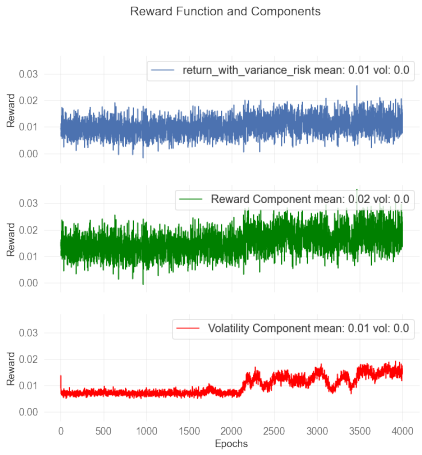

Figure 148: Two Asset Actor-Critic with Eligibility Traces Reward Function $\boldsymbol{\lambda = 0.8}$

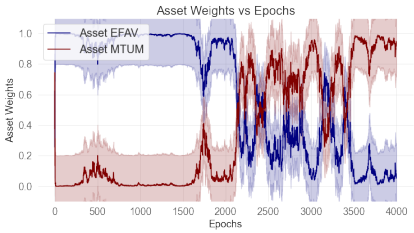

Figure 149: Two Asset Actor-Critic with Eligibility Traces Asset Weights $\boldsymbol{\lambda = 0.8}$

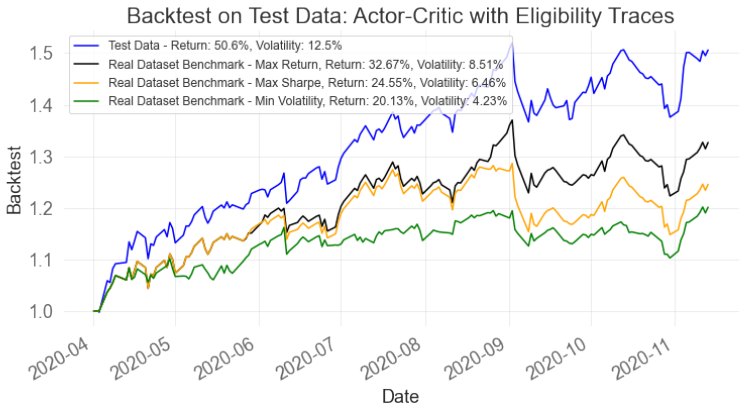

Figure 150: Two Asset Actor-Critic with Eligibility Traces Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.8}$

$\boldsymbol{\lambda = 1}$ Case (Maximum Return)

REINFORCE with Baseline

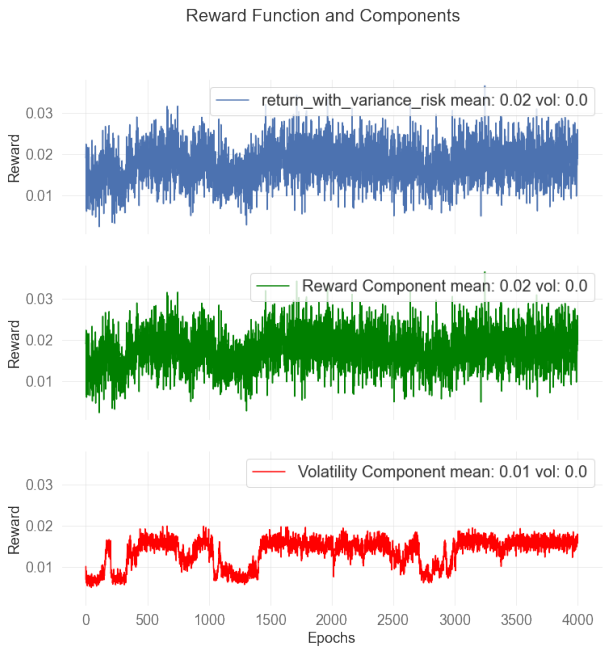

Figure 151: Two Asset REINFORCE with Baseline Reward Function $\boldsymbol{\lambda = 1}$

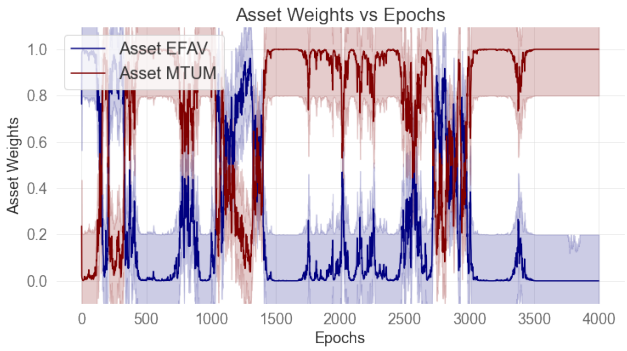

Figure 152: Two Asset REINFORCE with Baseline Asset Weights $\boldsymbol{\lambda = 1}$

Actor-Critic

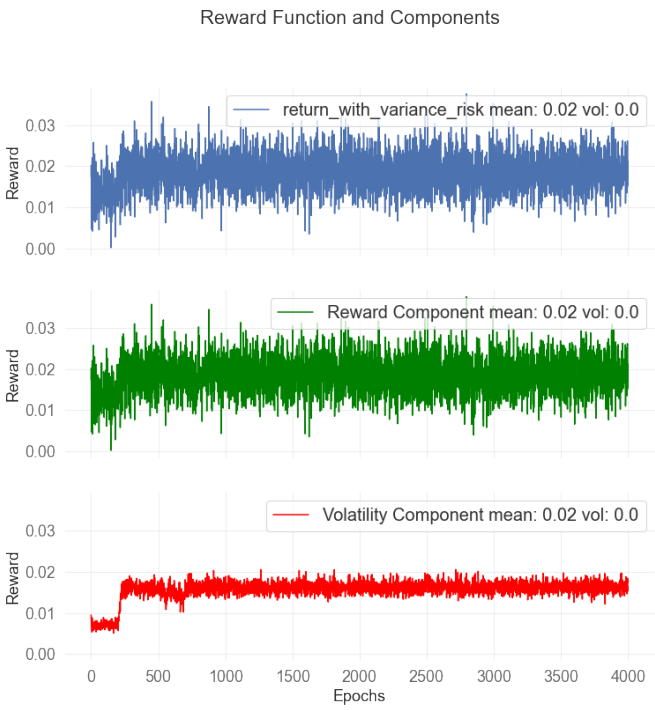

Figure 153: Two Asset Actor-Critic Reward Function $\boldsymbol{\lambda = 1}$

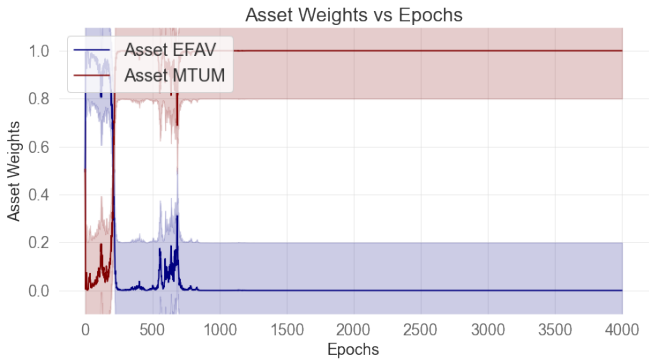

Figure 154: Two Asset Actor-Critic Asset Weights $\boldsymbol{\lambda = 1}$

Actor-Critic with Eligibility Traces

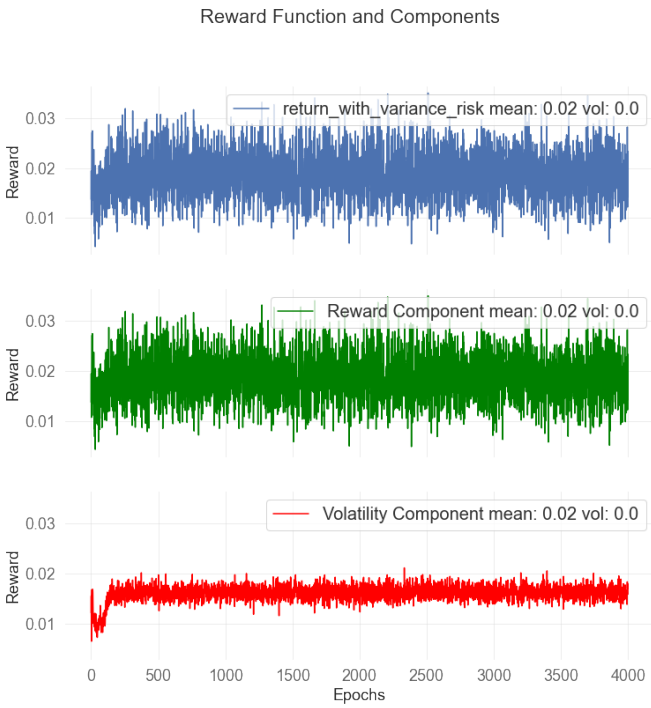

Figure 155: Two Asset Actor-Critic with Eligibility Traces Reward Function $\boldsymbol{\lambda = 1}$

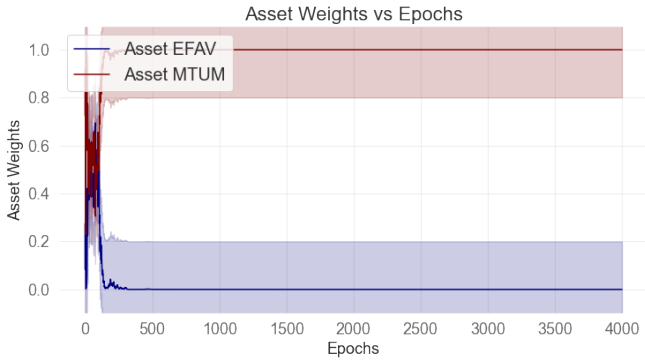

Figure 156: Two Asset Actor-Critic with Eligibility Traces Asset Weights $\boldsymbol{\lambda = 1}$

8.3 Full Portfolio Results

$\boldsymbol{\lambda = 0}$ Case (Minimum Volatility)

REINFORCE with Baseline

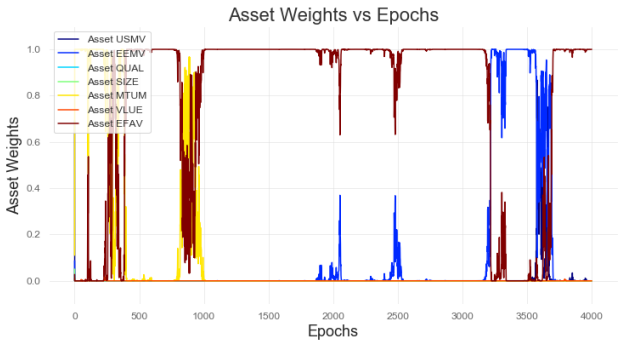

Figure 157: Full Portfolio REINFORCE with Baseline Asset Weights $\boldsymbol{\lambda = 0}$

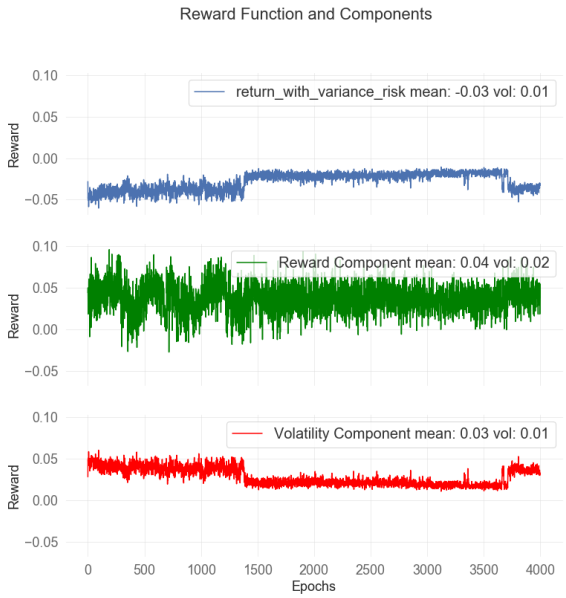

Figure 158: Full Portfolio REINFORCE with Baseline Reward Function $\boldsymbol{\lambda = 0}$

Actor-Critic

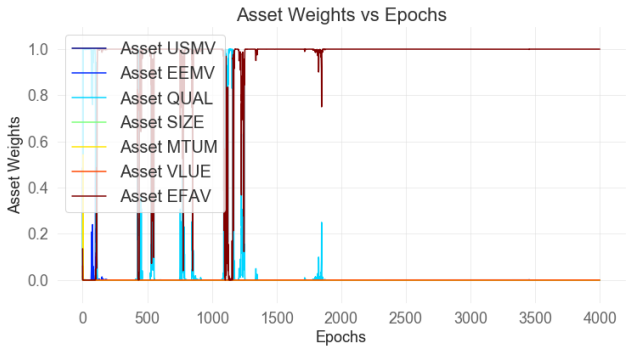

Figure 159: Full Portfolio Actor-Critic Asset Weights $\boldsymbol{\lambda = 0}$

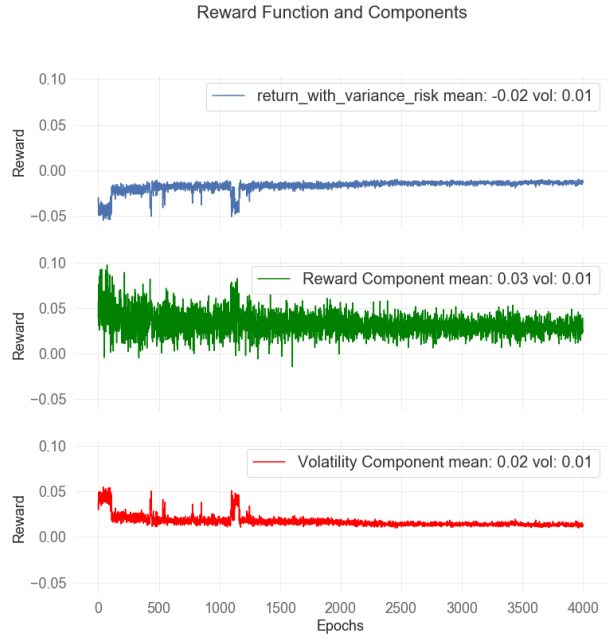

Figure 160: Full Portfolio Actor-Critic Reward Function $\boldsymbol{\lambda = 0}$

Actor-Critic with Eligibility Traces

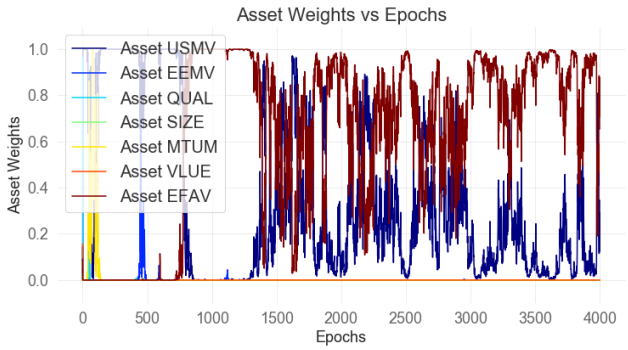

Figure 161: Full Portfolio Actor-Critic with Eligibility Traces Asset Weights $\boldsymbol{\lambda = 0}$

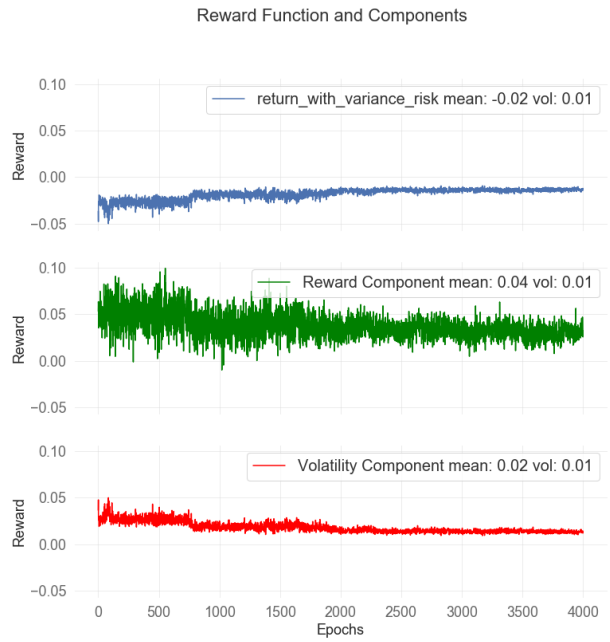

Figure 162: Full Portfolio Actor-Critic with Eligibility Traces Reward Function $\boldsymbol{\lambda = 0}$

$\boldsymbol{\lambda = 0.2}$ Case

REINFORCE

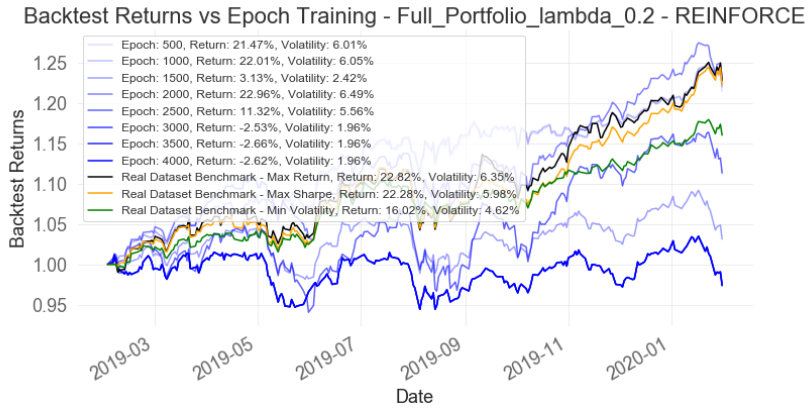

Figure 163: Full Portfolio REINFORCE Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.2}$

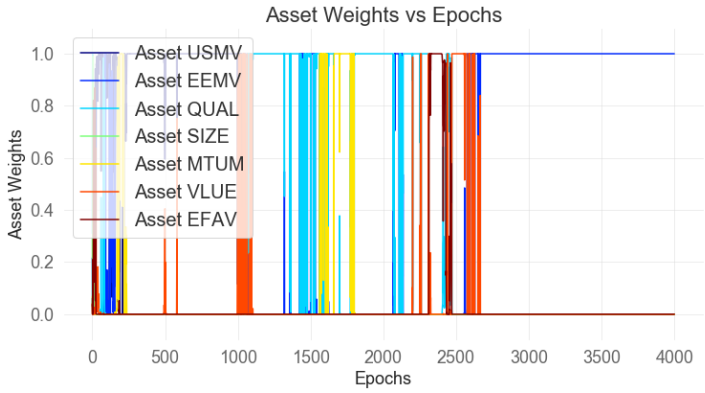

Figure 164: Full Portfolio REINFORCE Asset Weights $\boldsymbol{\lambda = 0.2}$

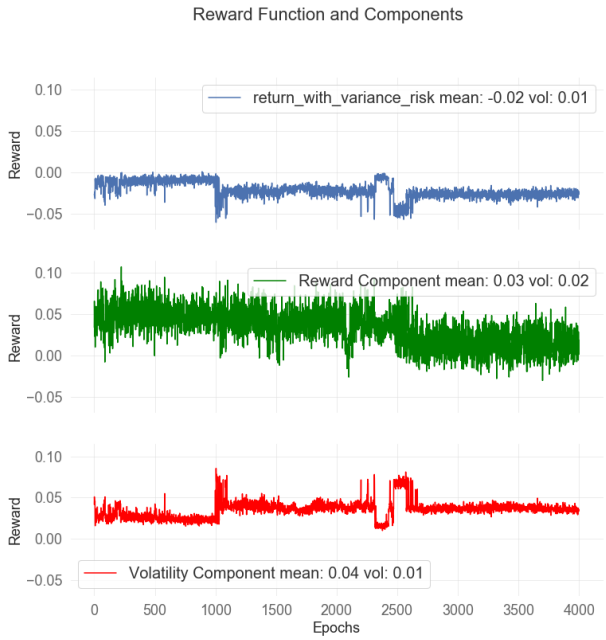

Figure 165: Full Portfolio REINFORCE Reward Function $\boldsymbol{\lambda = 0.2}$

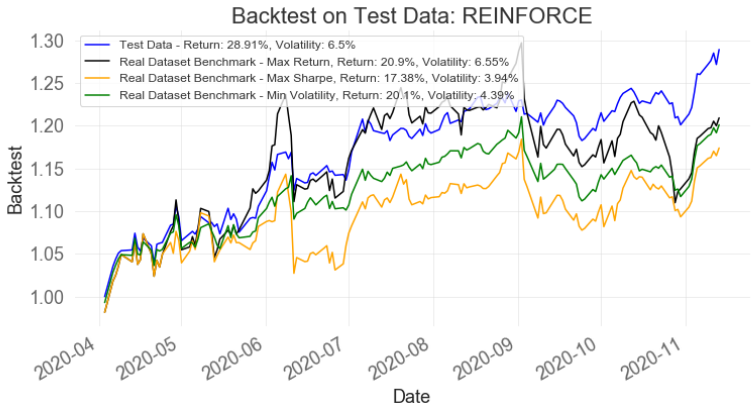

Figure 166: Full Portfolio REINFORCE Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.2}$

REINFORCE with Baseline

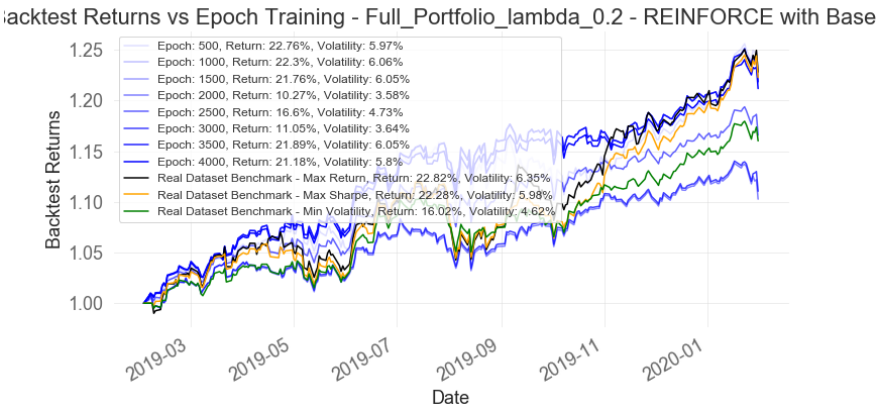

Figure 167: Full Portfolio REINFORCE with Baseline Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.2}$

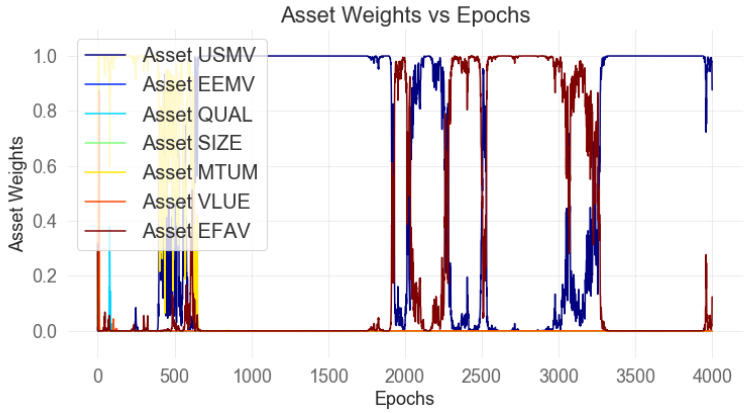

Figure 168: Full Portfolio REINFORCE with Baseline Asset Weights $\boldsymbol{\lambda = 0.2}$

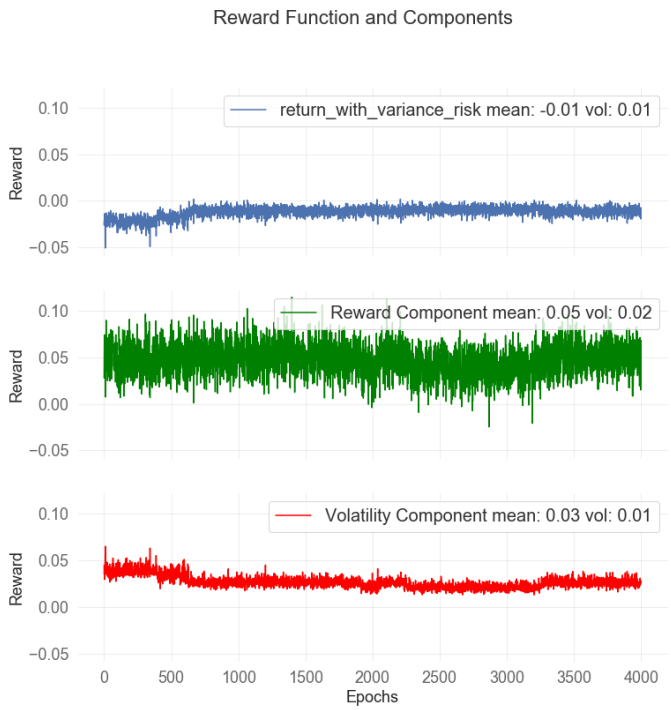

Figure 169: Full Portfolio REINFORCE with Baseline Reward Function $\boldsymbol{\lambda} = \mathbf{0.2}$

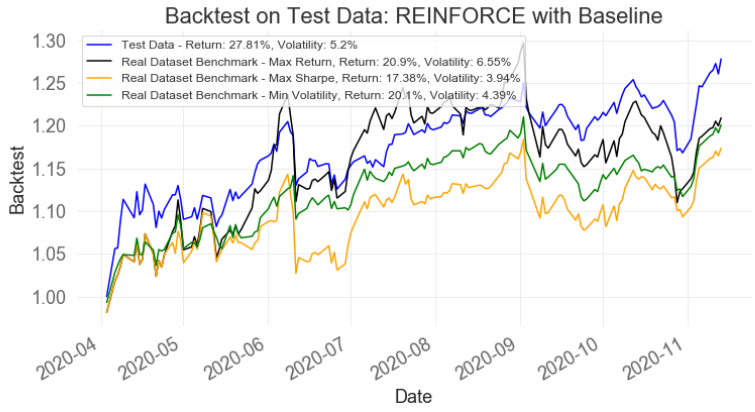

Figure 170: Full Portfolio REINFORCE with Baseline Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda} = \mathbf{0.2}$

Actor-Critic

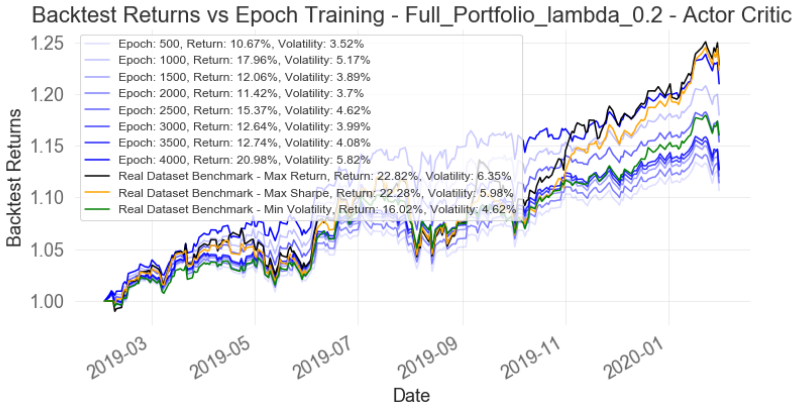

Figure 171: Full Portfolio Actor-Critic Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.2}$

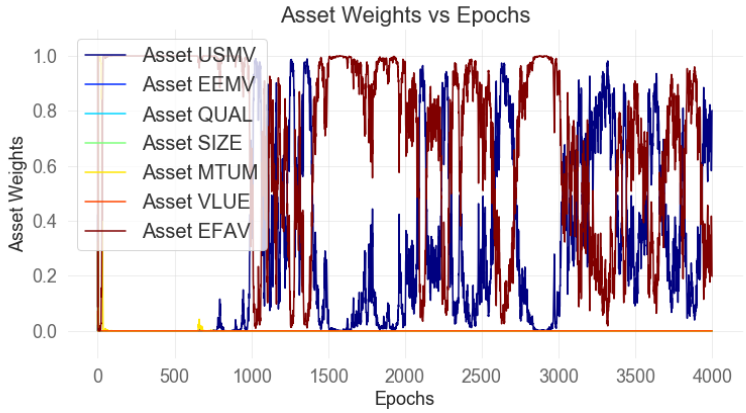

Figure 172: Full Portfolio Actor-Critic Asset Weights $\boldsymbol{\lambda = 0.2}$

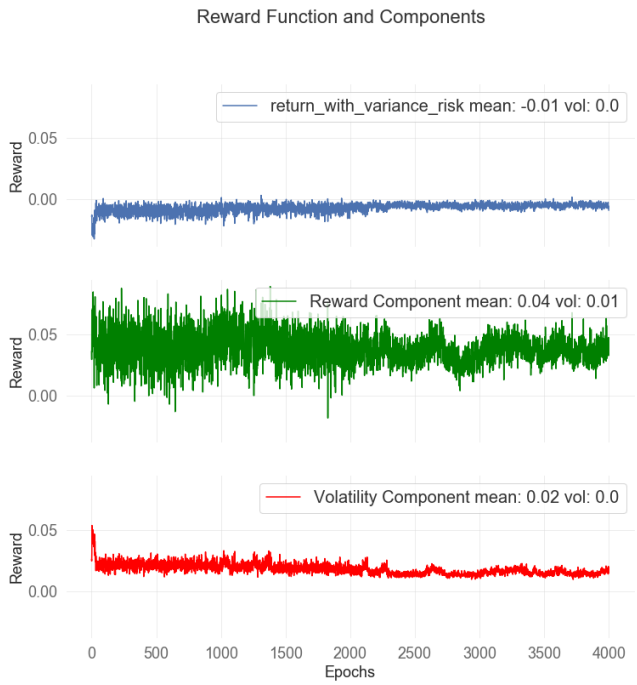

Figure 173: Full Portfolio Actor-Critic Reward Function $\boldsymbol{\lambda = 0.2}$

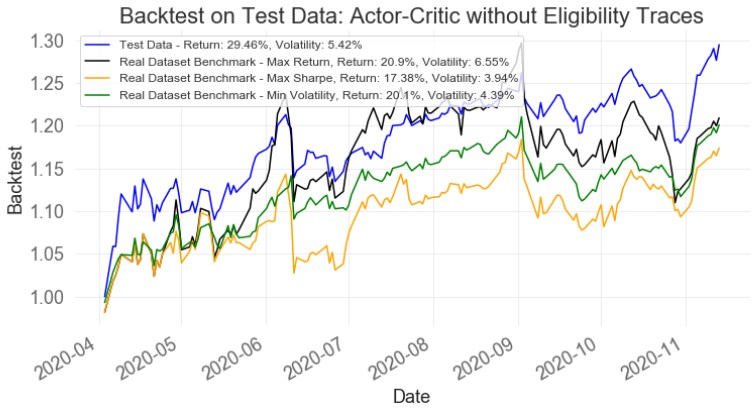

Figure 174: Full Portfolio Actor-Critic Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.2}$

Actor-Critic with Eligibility Traces

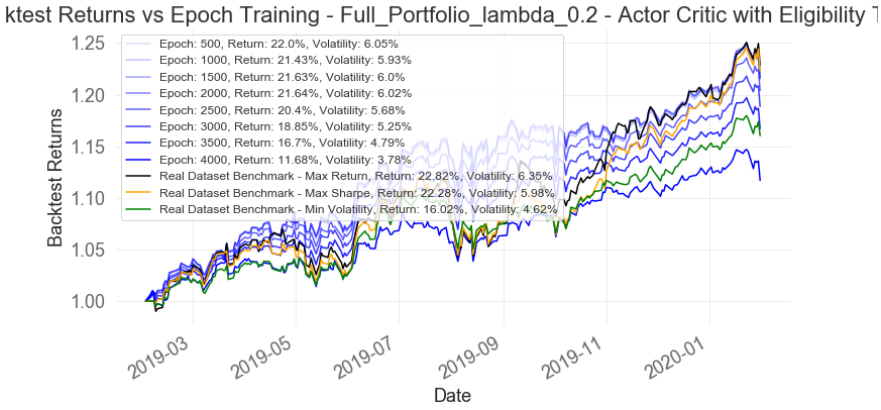

Figure 175: Full Portfolio Actor-Critic with Eligibility Traces Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.2}$

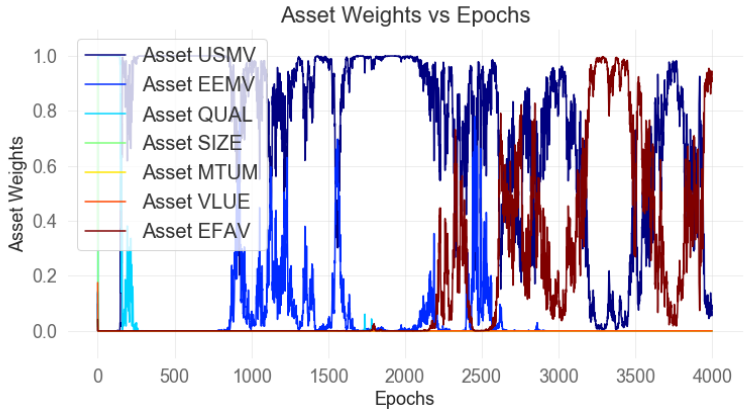

Figure 176: Full Portfolio Actor-Critic with Eligibility Traces Asset Weights $\boldsymbol{\lambda = 0.2}$

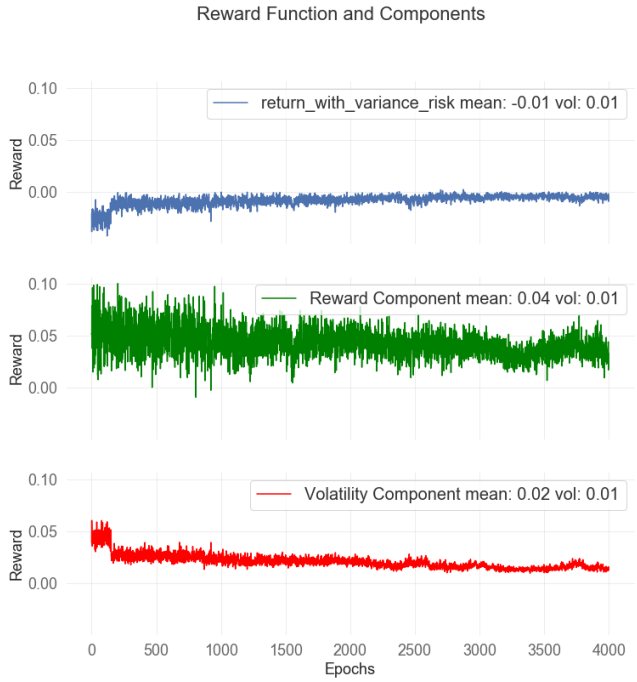

Figure 177: Full Portfolio Actor-Critic with Eligibility Traces Reward Function $\boldsymbol{\lambda = 0.2}$

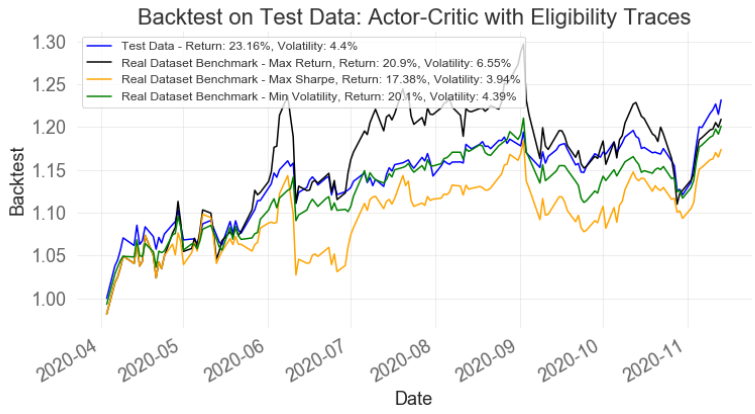

Figure 178: Full Portfolio Actor-Critic with Eligibility Traces Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.2}$

$\boldsymbol{\lambda = 0.5}$ Case

REINFORCE with Baseline

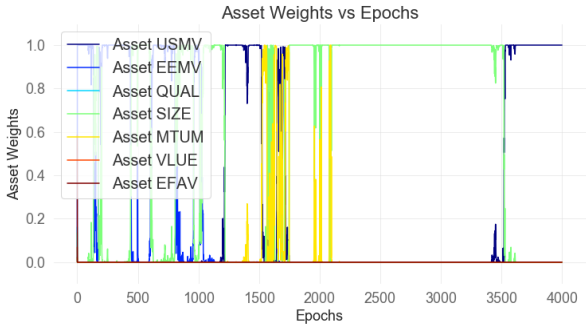

Figure 179: Full Portfolio REINFORCE with Baseline Asset Weights $\boldsymbol{\lambda = 0.5}$

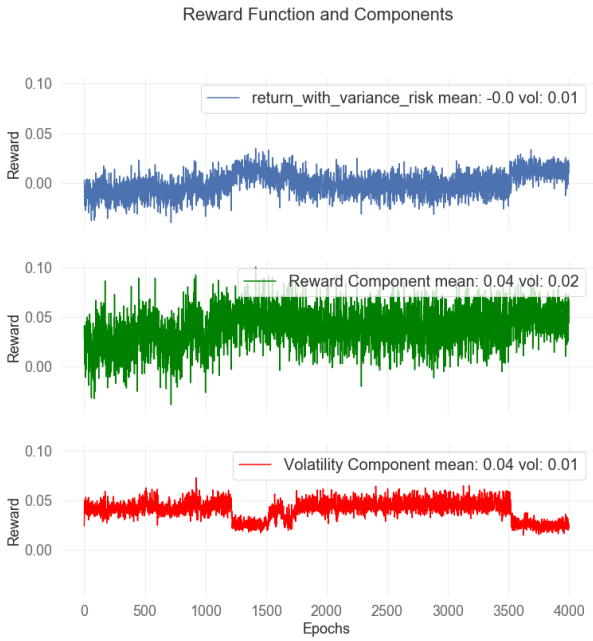

Figure 180: Full Portfolio REINFORCE with Baseline Reward Function $\boldsymbol{\lambda = 0.5}$

Actor-Critic

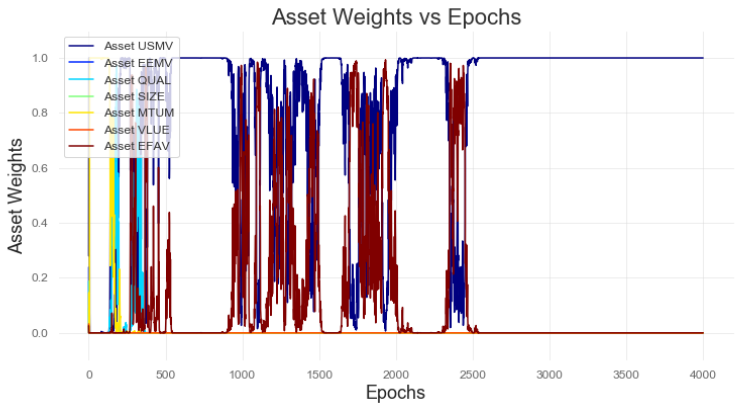

Figure 181: Full Portfolio Actor-Critic Asset Weights $\boldsymbol{\lambda = 0.5}$

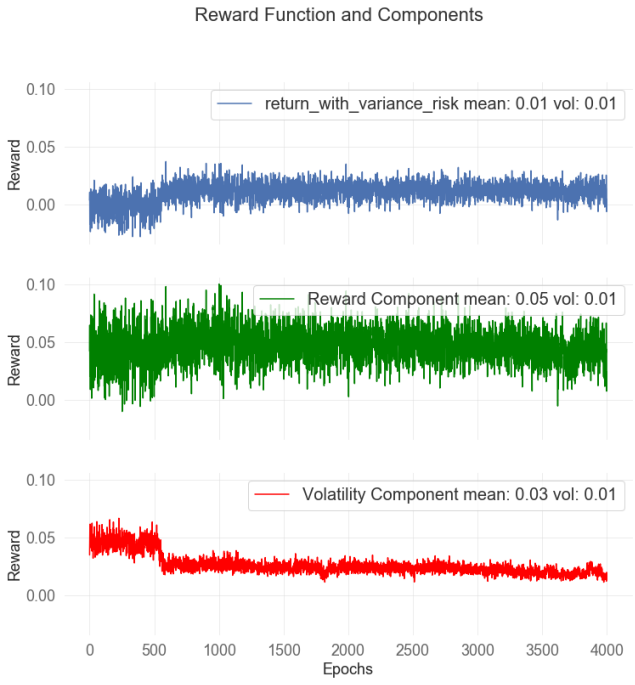

Figure 182: Full Portfolio Actor-Critic Reward Function $\boldsymbol{\lambda = 0.5}$

Actor-Critic with Eligibility Traces

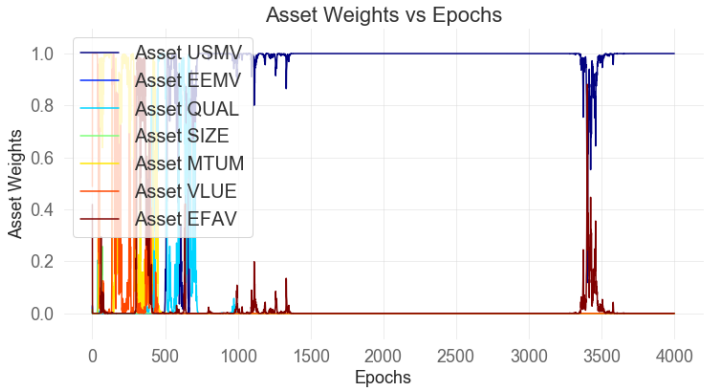

Figure 183: Full Portfolio Actor-Critic with Eligibility Traces Asset Weights $\boldsymbol{\lambda = 0.5}$

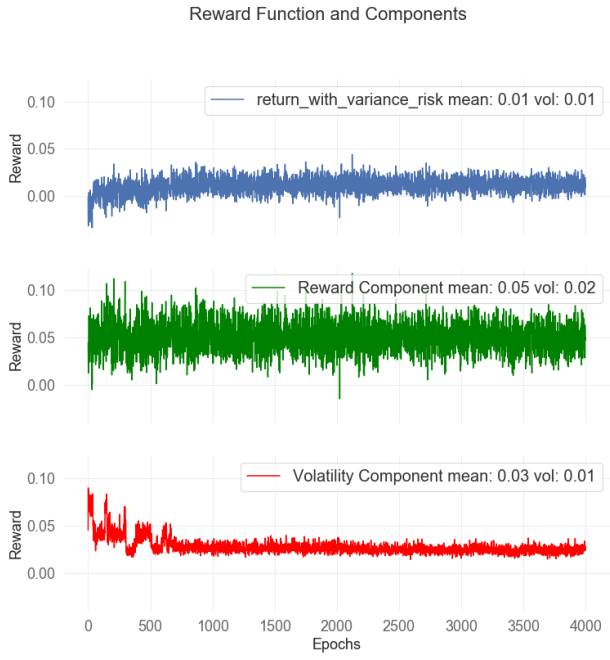

Figure 184: Full Portfolio Actor-Critic with Eligibility Traces Reward Function $\boldsymbol{\lambda = 0.5}$

$\boldsymbol{\lambda = 0.8}$ Case

REINFORCE

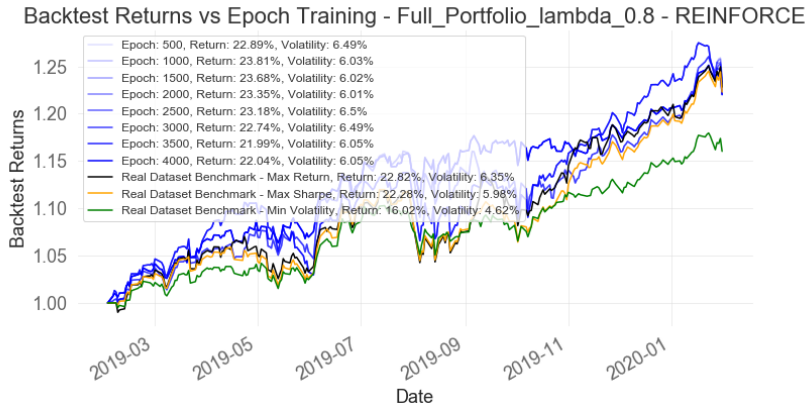

Figure 185: Full Portfolio REINFORCE Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.8}$

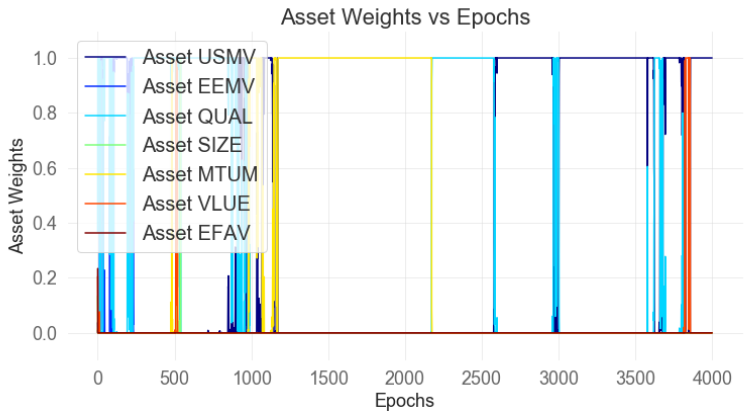

Figure 186: Full Portfolio REINFORCE Asset Weights $\boldsymbol{\lambda = 0.8}$

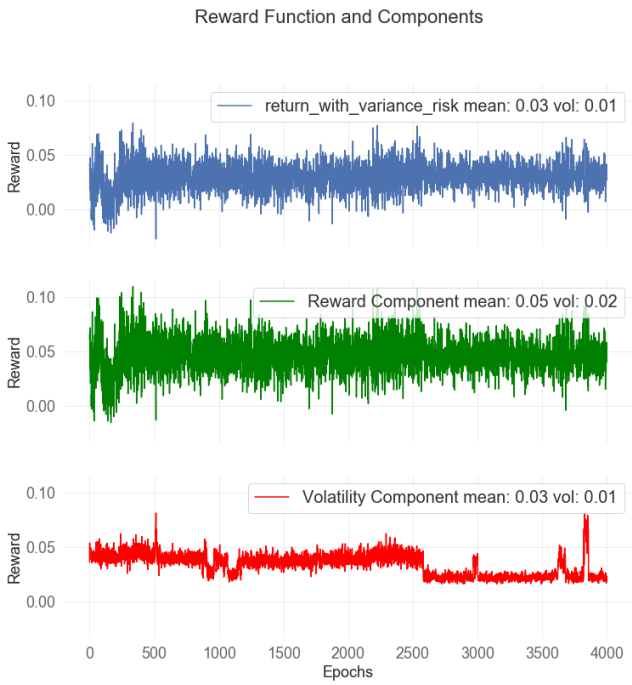

Figure 187: Full Portfolio REINFORCE Reward Function $\boldsymbol{\lambda = 0.8}$

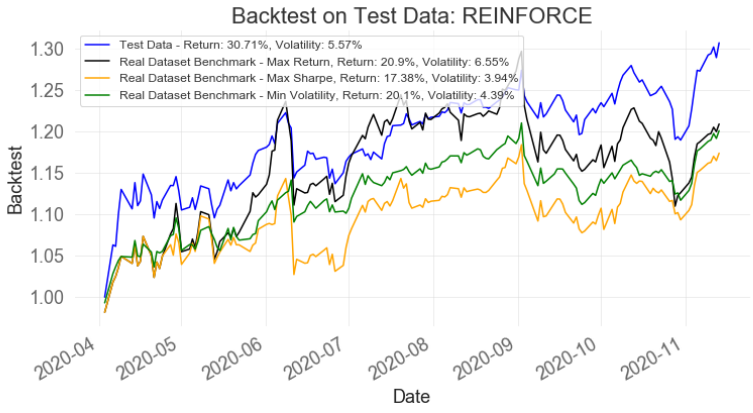

Figure 188: Full Portfolio REINFORCE Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.8}$

REINFORCE with Baseline

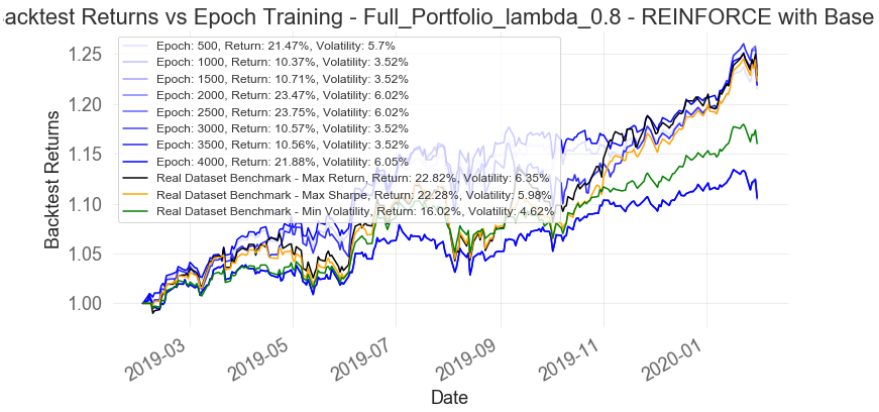

Figure 189: Full Portfolio REINFORCE with Baseline Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.8}$

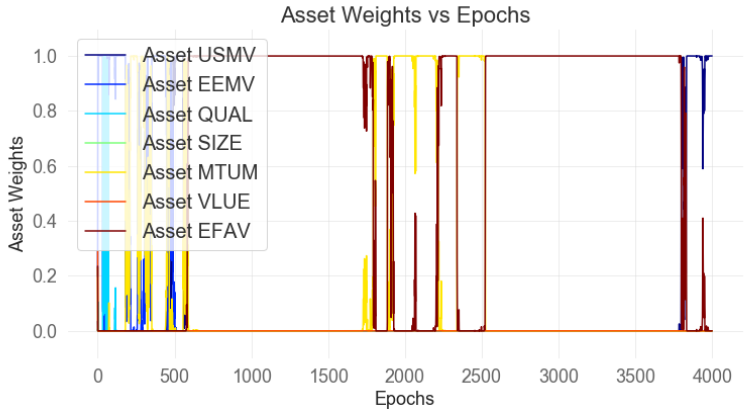

Figure 190: Full Portfolio REINFORCE with Baseline Asset Weights $\boldsymbol{\lambda = 0.8}$

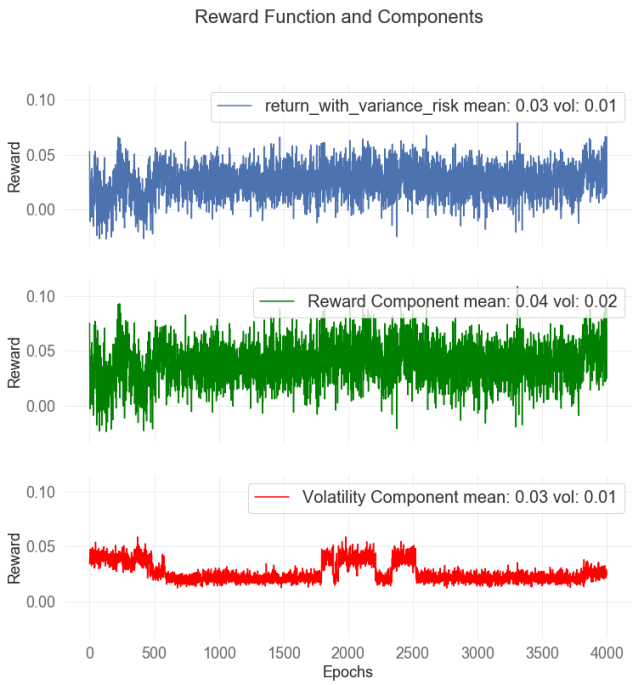

Figure 191: Full Portfolio REINFORCE with Baseline Reward Function $\boldsymbol{\lambda = 0.8}$

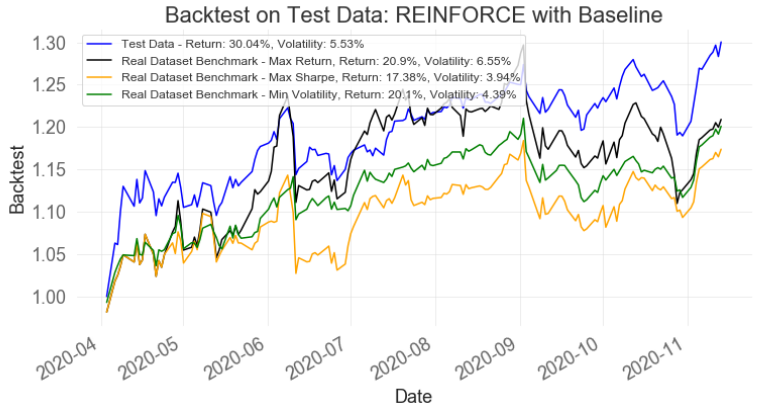

Figure 192: Full Portfolio REINFORCE with Baseline Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.8}$

Actor-Critic

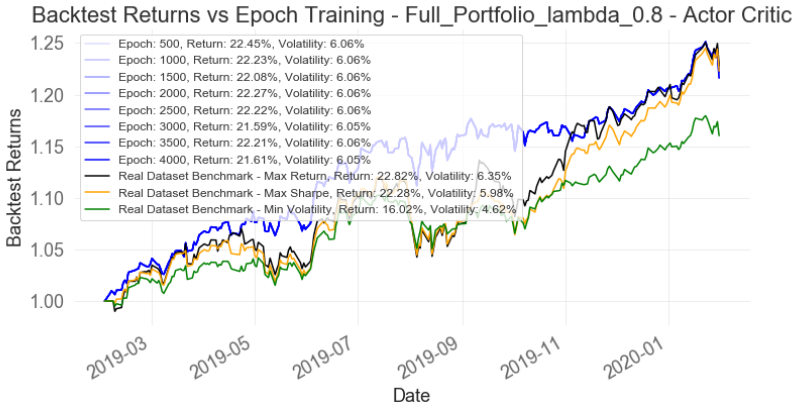

Figure 193: Full Portfolio Actor-Critic Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.8}$

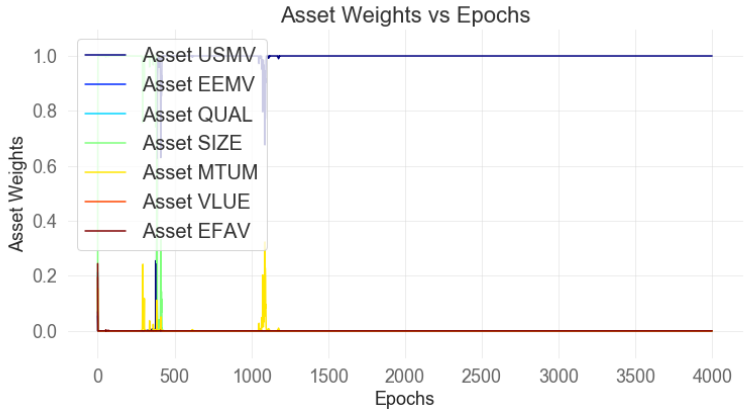

Figure 194: Full Portfolio Actor-Critic Asset Weights $\boldsymbol{\lambda = 0.8}$

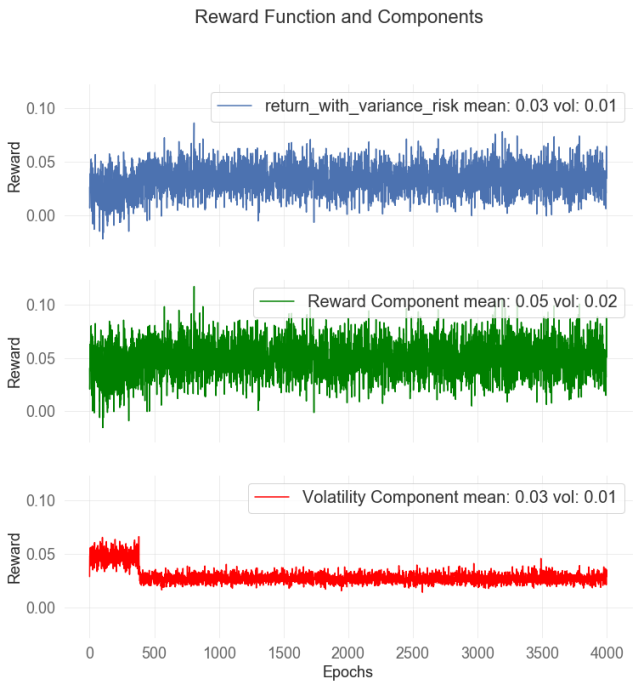

Figure 195: Full Portfolio Actor-Critic Reward Function $\boldsymbol{\lambda = 0.8}$

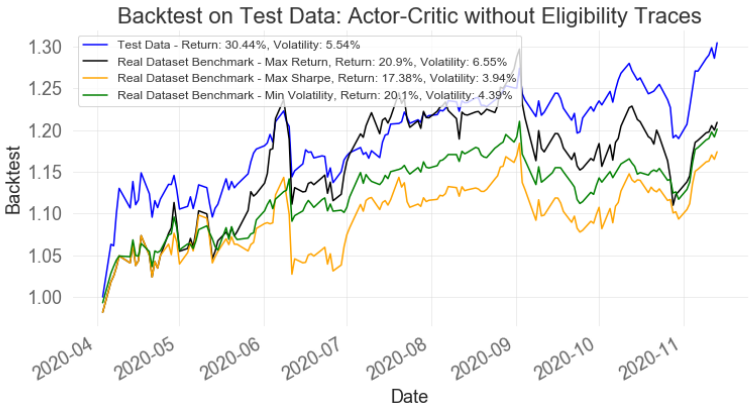

Figure 196: Full Portfolio Actor-Critic Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.8}$

Actor-Critic with Eligibility Traces

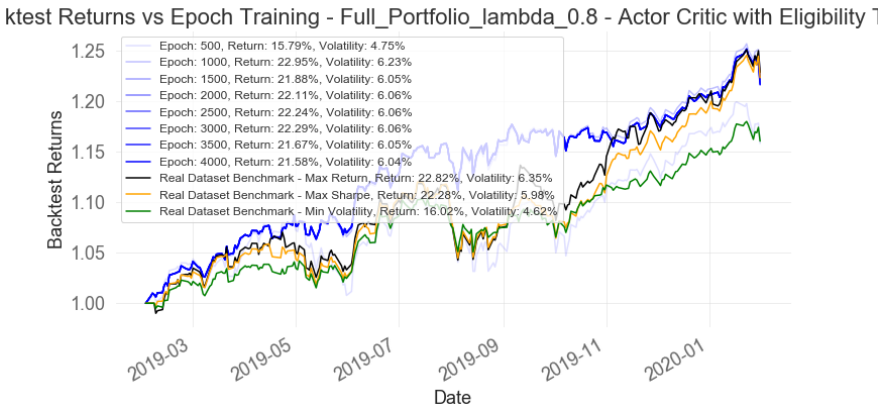

Figure 197: Full Portfolio Actor-Critic with Eligibility Traces Training Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.8}$

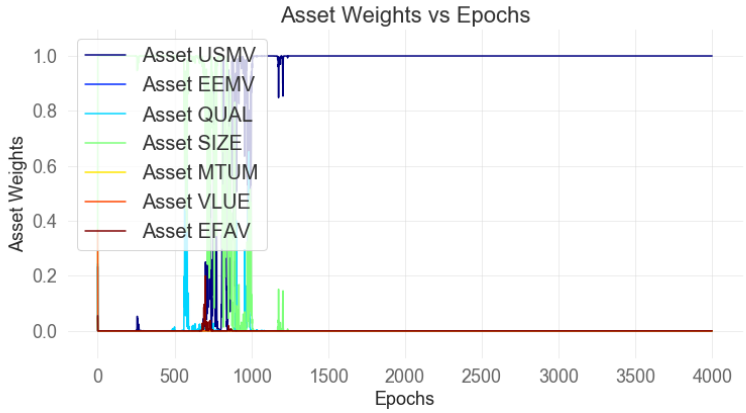

Figure 198: Full Portfolio Actor-Critic with Eligibility Traces Asset Weights $\boldsymbol{\lambda = 0.8}$

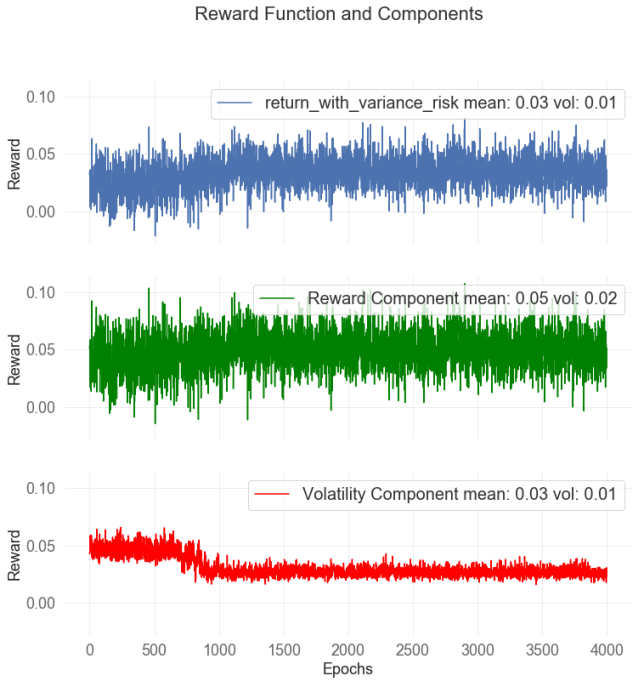

Figure 199: Full Portfolio Actor-Critic with Eligibility Traces Reward Function $\boldsymbol{\lambda = 0.8}$

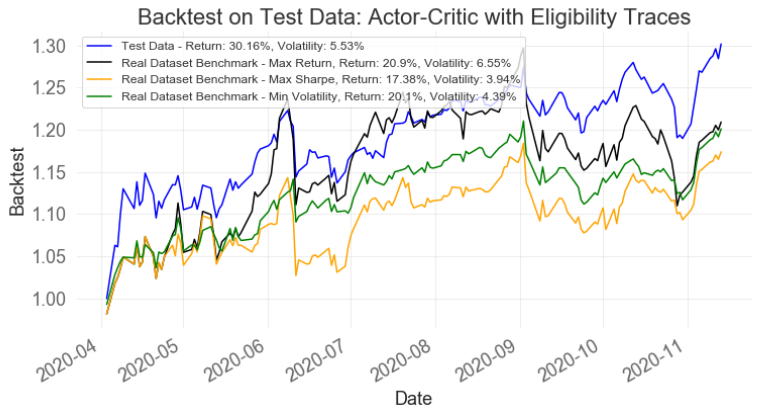

Figure 200: Full Portfolio Actor-Critic with Eligibility Traces Test Data Backtest Return vs. Benchmark Returns, $\boldsymbol{\lambda = 0.8}$

$\boldsymbol{\lambda = 1}$ Case (Maximum Return)

REINFORCE with Baseline

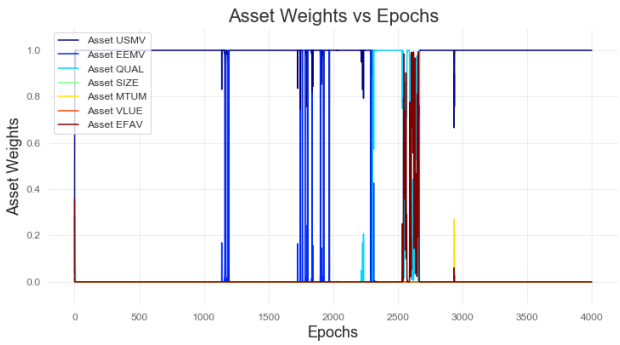

Figure 201: Full Portfolio REINFORCE with Baseline Asset Weights $\boldsymbol{\lambda = 1}$

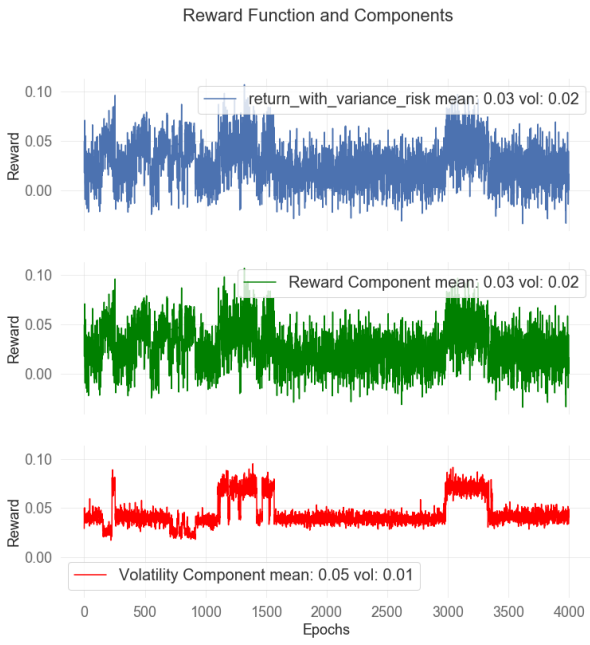

Figure 202: Full Portfolio REINFORCE with Baseline Reward Function $\boldsymbol{\lambda = 1}$

Actor-Critic

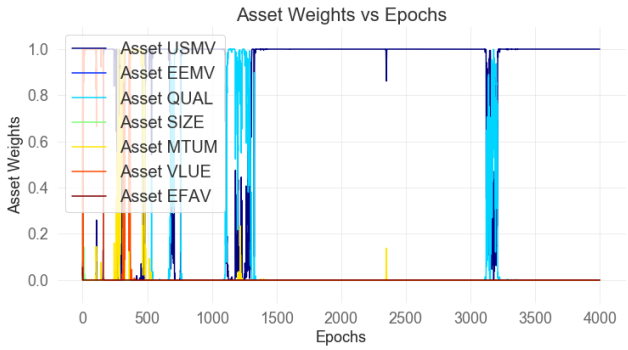

Figure 203: Full Portfolio Actor-Critic Asset Weights $\boldsymbol{\lambda = 1}$

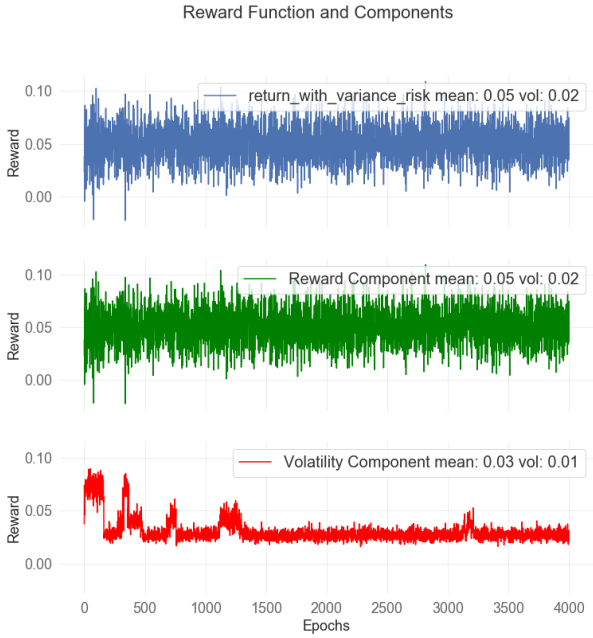

Figure 204: Full Portfolio Actor-Critic Reward Function $\boldsymbol{\lambda = 1}$

Actor-Critic with Eligibility Traces

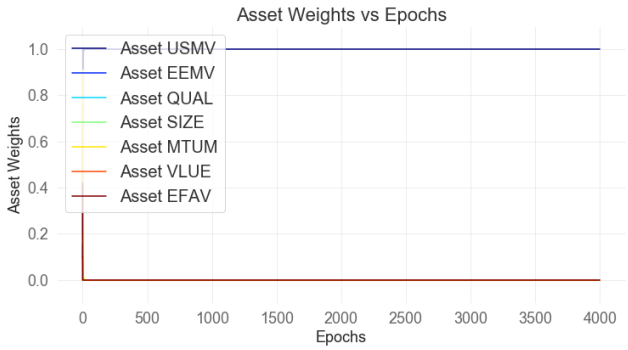

Figure 205: Full Portfolio Actor-Critic with Eligibility Traces Asset Weights $\boldsymbol{\lambda = 1}$

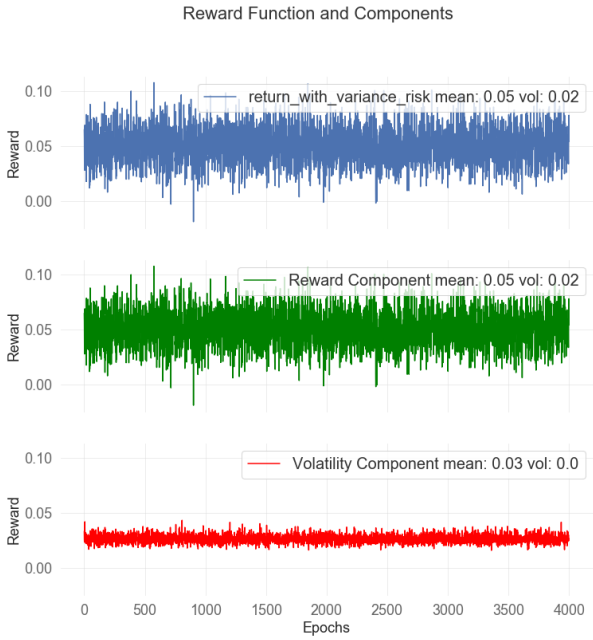

Figure 206: Full Portfolio Actor-Critic with Eligibility Traces Reward Function $\boldsymbol{\lambda = 1}$

8.4 Gradients in Model Training

In this section, we plot the evolution of the gradients of our features over the course of model training for a Full ETF portfolio with $\boldsymbol{\lambda} = \mathbf{0}$ and $\boldsymbol{\lambda} = \mathbf{1}$ using the REINFORCE algorithm. The gradients of the other algorithms behave in a similar manner. This illustrates how our Policy Gradient Methods learn from our features.

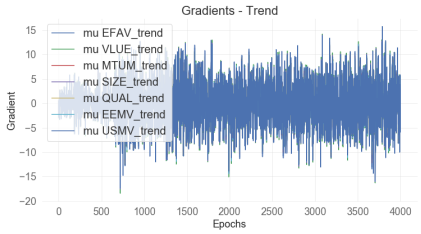

Figure 207: Full Portfolio REINFORCE Gradient: Trend, $\boldsymbol{\lambda} = \mathbf{0}$

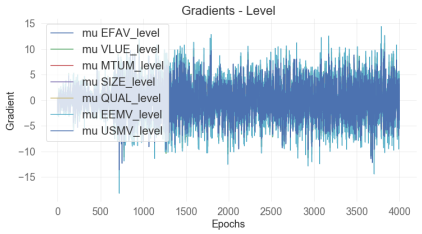

Figure 208: Full Portfolio REINFORCE Gradient: Level, $\boldsymbol{\lambda} = \mathbf{0}$

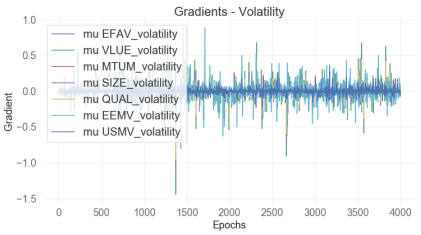

Figure 209: Full Portfolio REINFORCE Gradient: Volatility, $\boldsymbol{\lambda} = \mathbf{0}$

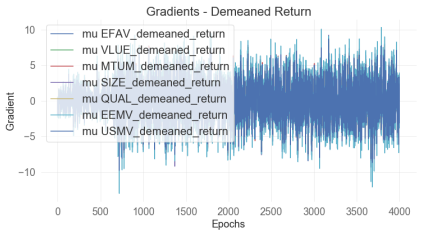

Figure 210: Full Portfolio REINFORCE Gradient: Demeaned Returns, $\boldsymbol{\lambda} = \mathbf{0}$

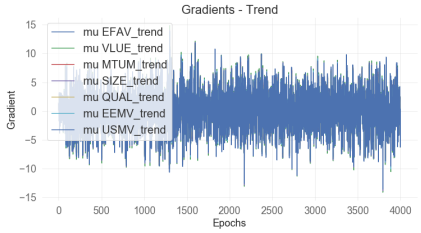

Figure 211: Full Portfolio REINFORCE Gradient: Trend, $\boldsymbol{\lambda = 1}$

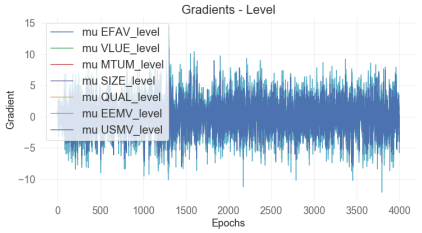

Figure 212: Full Portfolio REINFORCE Gradient: Level, $\boldsymbol{\lambda = 1}$

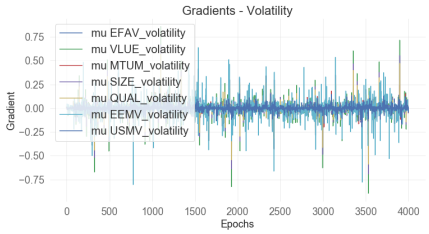

Figure 213: Full Portfolio REINFORCE Gradient: Volatility, $\boldsymbol{\lambda = 1}$

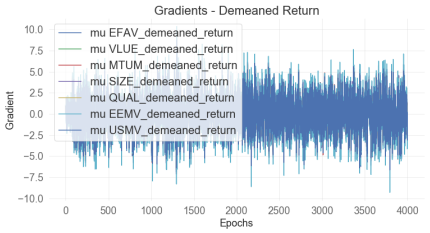

Figure 214: Full Portfolio REINFORCE Gradient: Demeaned Returns, $\boldsymbol{\lambda = 1}$

9 References

Cumming, James. An investigation into the use of reinforcement learning techniques within the algorithmic trading domain. Master's thesis, Imperial College London, United Kingdom, 2015. http://www.doc.ic.ac.uk/teaching/distinguished-projects/2015/j.cumming.pdf

Dempster, M.A.H. and Leemans, V. An automated fx trading system using adaptive reinforcement learning. Expert Systems with Applications, 30(3):543 – 552, 2006. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2005.10.012. http://www.sciencedirect.com/science/article/pii/S0957417405003015. Intelligent Information Systems for Financial Engineering.

Deng, Y., Bao, F. Kong, Y, Ren, Z. and Dai, Q Deep direct reinforcement learning for financial signal representation and trading. IEEE transactions on neural networks and learning systems, 28(3):653–664, 2017.

Honchar, Alexandr. "AI for portfolio management: from Markowitz to Reinforcement Learning." Medium: The Startup. 28 Sep 2019. https://medium.com/swlh/ai-for-portfolio-management-from-markowitz-to-reinforcement-learning-cffedcbba566

Forseth, Erik and Tricker, Ed. "Equal Risk Contribution Portfolios," Graham Capital Management, March 2019.

Lapan, Maxim. Deep Reinforcement Learning Hands-on: Apply Modern RL Methods, with Deep Q-Networks, Value Iteration, Policy Gradients, TRPO, AlphaGo Zero and More. Packt Publishing, 2018.

Market data APIs. Barchart.com — Commodity, Stock, and Currency Quotes, Charts, News Analysis. https://www.barchart.com/ondemand

Moody, J. and Saffell, M. Learning to trade via direct reinforcement. IEEE Transactions on Neural Networks, 12(4):875–889, Jul 2001. ISSN 1045-9227. doi: 10.1109/72.935097.

Ng, Yibin. "Forecasting Stock Prices using Exponential Smoothing." Towards Data Science, 29 Sept. 2019. https://towardsdatascience.com/forecasting-stock-prices-using-exponential-smoothing-b37dfe54e8e9

Shriber, Todd. "Survey Confirms Smart Beta Growth Trajectory." Investopedia, Investopedia, 29 Jan. 2020. www.investopedia.com/news/survey-confirms-smart-beta-growth-trajectory

Sutton, Richard S., and Andrew G. Barto. Reinforcement Learning: an Introduction. The MIT Press, 2018. http://incompleteideas.net/book/the-book.html