**AI Financial Advisor: A Deep Reinforcement Learning Approach to Portfolio Management**

**Author:** Andrew Branum

**Date:** June 9, 2025

**Version:** 1.0 (Draft)

**Abstract**

This report details the design, implementation, and evaluation of an automated financial portfolio management system powered by Deep Reinforcement Learning (RL). The objective is to develop an autonomous agent capable of making dynamic asset allocation decisions to maximize risk-adjusted returns. The system features a modular architecture, including a robust data pipeline for collecting and processing market data, a custom-built simulation environment that models realistic market conditions like transaction costs, and an advanced RL agent trained using Proximal Policy Optimization (PPO). We employ a rigorous hyperparameter optimization process using Optuna and evaluate the final agent against standard benchmarks (Equal-Weight and SPY Buy-and-Hold) on unseen data, measuring performance with a comprehensive suite of financial metrics including the Sharpe Ratio, Max Drawdown, and Calmar Ratio. This document outlines the complete methodology and presents the framework for analyzing the agent's performance and strategic behavior.

**1. Introduction**

The task of continuously managing an investment portfolio is a complex, dynamic optimization problem. A successful strategy must balance the pursuit of high returns with the management of risk, adapting to ever-changing market conditions. While traditional portfolio management theories provide a static framework, they often fall short in capturing the non-linear, path-dependent nature of financial markets.

Deep Reinforcement Learning offers a compelling alternative. By framing portfolio management as a sequential decision-making process, an RL agent can learn a sophisticated, state-dependent policy directly from market data. The agent learns to map market observations to trading decisions (actions) in order to maximize a cumulative reward signal over time. This data-driven approach allows the agent to uncover complex patterns and strategies that may not be apparent to human traders or static models.

This project implements an end-to-end system to train and evaluate such an RL agent. The goal is to create a robust and practical "AI Financial Advisor" that can autonomously manage a portfolio of assets.

**2. System Architecture & Methodology**

The project is structured into three core components: a data pipeline, a realistic simulation environment, and a training and evaluation framework.

**2.1. Data Pipeline**

The foundation of any financial machine learning system is high-quality data. The data\_collector.py script establishes an automated pipeline for this purpose.

* **Data Source:** Historical market data is sourced from the yfinance library, providing daily Open, High, Low, Close, and Volume (OHLCV) data.
* **Feature Engineering:** Raw price data is enriched with a comprehensive set of technical indicators using the ta library. These features provide the agent with a richer representation of the market state. The calculated features include:
  + **Trend Indicators:** SMA (10, 50), MACD
  + **Momentum Indicators:** RSI (14), Momentum (10)
  + **Volatility Indicators:** Bollinger Bands (High, Low, Mid), Average True Range (ATR), Historical Volatility (5, 20)
* **Data Storage:** All processed data is stored locally in a SQLite database (market\_data.db). This creates a persistent, queryable data warehouse, preventing the need to re-download and re-process data for every experiment. The schema is designed with a UNIQUE(ticker, date) constraint to ensure data integrity and prevent duplicate entries.

**2.2. Simulation Environment (PortfolioEnv)**

The rl/portfolio\_env.py script defines a custom portfolio management environment that adheres to the Gymnasium API standard. This environment is the simulated world where the agent learns to trade.

* **Observation Space:** The agent's perception of the market is a windowed, multi-feature tensor. At each step, the agent observes the last window\_size (e.g., 30 days) of data for all features across all assets in the portfolio. This temporal structure allows the agent to recognize trends and patterns. A sophisticated normalization scheme is applied, where price-based features are normalized relative to the start of the window and oscillators like RSI are scaled to a 0-1 range.
* **Action Space:** The agent's action is a continuous vector representing the target weights for each asset in the portfolio. The actions are automatically normalized to sum to 1, representing a fully invested portfolio.
* **Market Dynamics:** The environment simulates the core mechanics of trading, including:
  + **Transaction Costs:** To penalize excessive trading and encourage more efficient strategies, a configurable percentage-based transaction cost is applied every time the agent rebalances its portfolio.
  + **Portfolio Rebalancing:** The environment calculates daily portfolio returns based on the agent's chosen weights and the price movements of the underlying assets.

**2.3. Reward Shaping**

The design of the reward function is critical to guiding the agent's learning process. A simple profit-based reward can lead to unacceptably volatile strategies. Therefore, our environment employs an advanced reward shaping mechanism to explicitly encourage risk-aware behavior. The final reward at each step is calculated as:

Final Reward = (Raw Portfolio Return \* Loss Aversion Factor) - (Volatility Penalty)

* **Loss Aversion:** Negative daily returns are multiplied by a loss\_aversion\_factor (e.g., 1.5), making the agent more sensitive to losses than it is to equivalent gains.
* **Volatility Penalty:** The reward is penalized by the standard deviation of the portfolio's returns over a rolling window (e.g., 20 days), scaled by a volatility\_penalty\_weight. This directly incentivizes the agent to generate a smoother equity curve.

**2.4. Agent and Training Algorithm**

The project utilizes the **Proximal Policy Optimization (PPO)** algorithm, implemented in the stable-baselines3 library. PPO is a state-of-the-art, on-policy algorithm known for its stability, reliability, and excellent performance across a wide range of tasks, making it a strong choice for this financial application.

**3. Experimental Design**

A rigorous experimental setup is crucial for developing a model that generalizes to new, unseen data. Our workflow is divided into two distinct phases: hyperparameter optimization and final evaluation.

**3.1. Hyperparameter Optimization**

The rl/train\_baseline\_optuna.py script orchestrates a systematic search for the best model and environment hyperparameters using the **Optuna** framework.

* **Objective:** The goal is to find the combination of parameters that maximizes the annualized Sharpe Ratio.
* **Process:** For each trial, Optuna suggests a new set of hyperparameters. A PPO agent is trained using these parameters on a dedicated training dataset (e.g., 2010-2018).
* **Validation:** The performance of the trained agent is then measured on a separate, out-of-sample validation set (e.g., 2019-2020). The Sharpe Ratio on this validation set is returned to Optuna as the objective score. This strict separation of training and validation data prevents overfitting and ensures that the selected hyperparameters are likely to perform well on future data.

**3.2. Final Model Evaluation**

The rl/evaluate.py script provides a comprehensive framework for assessing the performance of the final, trained agent on a completely unseen hold-out test set (e.g., 2023-Present).

* **Benchmarks:** To provide context for the agent's performance, it is evaluated against two standard baseline strategies:
  1. **Equal-Weight Portfolio:** A portfolio that maintains an equal allocation to all assets.
  2. **Market Index Buy-and-Hold:** The performance of the SPY ETF over the same period.
* **Key Performance Indicators (KPIs):** A wide array of financial metrics are calculated and reported for both the agent and the benchmarks:
  1. **Risk-Adjusted Return:** Annualized Sharpe Ratio, Calmar Ratio.
  2. **Return:** Cumulative Return, Final Portfolio Value.
  3. **Risk:** Max Drawdown, Annualized Volatility.
  4. **Trading Behavior:** Average Daily Turnover, Total Transaction Costs.
* **Visualization:** The script automatically generates a suite of plots to facilitate qualitative analysis, including portfolio value comparisons, agent weight allocation over time, and distributions of daily returns.

**4. Preliminary Results & Discussion**

*(This section is a placeholder to be filled in after the Optuna study and final evaluation are complete.)*

The hyperparameter optimization study, run for [Number] trials, identified a set of optimal parameters that yielded a validation Sharpe Ratio of [Best Sharpe]. The most influential parameters were observed to be [Parameter 1] and [Parameter 2], suggesting...

The final agent, trained on the full dataset using these optimized parameters, was evaluated on the hold-out test period from [Start Date] to [End Date]. The agent achieved a final portfolio value of [$Value], outperforming the Equal-Weight benchmark but underperforming the SPY Buy-and-Hold strategy.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **RL Agent** | **Equal-Weight** | **SPY Buy-and-Hold** |
| Cumulative Return (%) | [Value] | [Value] | [Value] |
| Annualized Sharpe Ratio | [Value] | [Value] | [Value] |
| Max Drawdown (%) | [Value] | [Value] | [Value] |
| Calmar Ratio | [Value] | [Value] | [Value] |

The weight allocation plot reveals that the agent learned a [Describe Strategy, e.g., momentum-following, mean-reversion] strategy. It tended to [Describe Behavior, e.g., increase allocation to assets with rising RSIs while reducing exposure to assets with high recent volatility].

**5. Conclusion & Future Work**

This project successfully demonstrates a complete, end-to-end framework for applying Deep Reinforcement Learning to the complex challenge of financial portfolio management. We have built a robust data pipeline, a realistic simulation environment with advanced reward shaping, and a rigorous training and evaluation methodology.

The current system provides a strong foundation for future research and development. Several avenues for future work include:

* **Expanding the Asset Universe:** Incorporating a wider and more diverse set of assets, including different sectors, international equities, and other asset classes like bonds and commodities.
* **Exploring Alternative RL Algorithms:** While PPO is a strong baseline, evaluating other modern algorithms like Soft Actor-Critic (SAC) could yield different and potentially superior trading strategies.
* **Integrating Macroeconomic Data:** Enhancing the agent's observation space with macroeconomic indicators (e.g., interest rates, inflation data) could allow it to learn strategies that adapt to broader economic regimes.
* **Live Trading Integration:** Developing the architecture to connect the trained agent to a brokerage API for paper or live trading, which would introduce new challenges such as latency and API reliability.