# **Workflow & Overview**

Code:

[implementation](file:///Users/sumit/Library/Application%20Support/JetBrains/PyCharm2024.3/scratches/Reinforcement%20Learning%20Finance/DRL_for_portfolio_trading_chen.py)

# Overview

A screenshot of a portfolio optimization tool

Description automatically generated

A screenshot of a computer

Description automatically generated

***V(s\_t) = E [R\_t + γR\_{t+1} + γ²R\_{t+2} + ... + γ^{T-t} R\_T | s\_t]*** *(1)*

**Where** each **R\_t** is the **differential Sharpe ratio**:

R\_t = δS\_t = (ρ\_t - μ\_t)/σ\_t² - 0.5 × S\_t × (ρ\_t - μ\_t)²/σ\_t³

And **ρ\_t** is the **portfolio return**:

ρ\_t = Σᵢ₌₁¹¹ wᵢ(t-1) × rᵢ(t) + w\_cash(t-1) × 0

**NOTE**: The Value Network predicts: "Given current market state s\_t, what is the expected cumulative differential Sharpe performance from all future portfolio decisions until episode end?"

A screenshot of a computer screen

Description automatically generated

A screenshot of a device

Description automatically generated

A screenshot of a test

Description automatically generated

## **What’s the purpose of Value Network**

# The Value network (critic) serves a crucial role in **variance reduction** during training. Here's what it does:

Without Value network:

*# Gradient uses total future rewards directly*

gradient ∝ log π(a|s) × (R\_t + R\_{t+1} + ... + R\_T) where R\_t is the reward (differential Sharpe) at time t

**Problem**: Future rewards have high variance - sometimes you get lucky/unlucky with market movements, making learning unstable.

**With Value Network (Actor-Critic):**

python

*# Value network predicts expected future rewards*

V(s) = E[R\_t + R\_{t+1} + ... + R\_T | state = s]

*# Advantage = actual rewards - baseline expectation*

A(s, a) = (R\_t + R\_{t+1} + ... + R\_T) **- V(s)**

*# Gradient uses advantage instead*

gradient ∝ log π(a|s) × **A(s, a)**

## **What’s the PPO loss function?**

The single log\_prob number from the policy network tells us: **"How likely was this specific portfolio allocation according to the current policy?"**

**Concrete Example**

Suppose at timestep t:

**State**: Market is volatile, tech sector showing strength

**Action taken**: [0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.50, 0.05, 0.05, 0.05, 0.05] (50% tech, rest equal)

**log\_prob = -1.2. *(this is the log\_prob coming out of the policy network)***

This means: π (allocating 50% to tech | current market state) has probability e^(-1.2) ≈ 0.30

**What We Use It For**

**1. PPO Probability Ratio:**

ratio = torch.exp (new\_log\_probs - old\_log\_probs)

If:

old\_log\_prob = -2.0 (old policy thought this allocation was unlikely)

new\_log\_prob = -1.2 (new policy thinks it's more likely)

ratio = e^(-1.2 - (-2.0)) = e^0.8 ≈ 2.23

**Interpretation**: New policy is 2.23x more likely to take this action than old policy.

**2. Policy Update Direction:**

policy\_loss = -ratio \* advantage

* If advantage > 0 (action was good) and ratio > 1 (new policy more likely to take it) → loss is negative → **encourage this behavior**
* If advantage < 0 (action was bad) and ratio > 1 (new policy more likely to take it) → loss is positive → **discourage this behavior**

**Key Insight**

The single log\_prob (from the policy network) is a **confidence score** for the entire portfolio decision:

* **High log\_prob** (closer to 0): "I'm confident this allocation makes sense"
* **Low log\_prob** (very negative): "This allocation seems unlikely/risky given current conditions"

It's how the agent quantifies its **certainty** about portfolio choices, which drives the learning process.

## **What is Differential Sharpe (Reward)**

1. Running Statistics

At each time step t (after n returns have been observed), we maintain:

* Sample mean return:

* Sample variance (online update form of *Welford’s algorithm*):

where are the observed returns.

These are updated incrementally with the formula:

## 2. Sharpe Ratio Estimate

At each step, once n≥2:

Where

## 3. Differential Sharpe Ratio

The code computes something resembling the **incremental (differential) change in Sharpe ratio** from adding the new return

The returned value is:

## 4. Interpretation

* The first term

​​

represents the **marginal effect of the new return** on the Sharpe ratio (like a gradient term).

* The second term

acts as a **correction term** that penalizes deviations proportional to the squared residual, scaled by the Sharpe ratio.

**5. Summary**

Mathematically, this implements:

* Online computation of mean and variance of returns.
* At each new return ​, it outputs an approximation of the **instantaneous derivative of the Sharpe ratio** with respect to adding that return.

So in compact notation:

where , ​ are the updated sample mean and standard deviation after including ​.

# APPENDIX

1. <https://gatambook.substack.com/p/deep-reinforcement-learning-for-portfolio?utm_source=substack&utm_medium=email>