REINFORCEMENT LEARNING

Deep Reinforcement Learning Trading Agent

Training an AI Agent to Trade SPY

Diego Sanchez-Carapia | December 5th 2025 | CSCI4170

01

Problem Statement

THE CHALLENGE

Rule-based trading systems cannot adapt to changing market conditions, leading to poor performance during volatility or trend shifts.

OUR SOLUTION

Train a Deep Q-Network agent that learns optimal buy, sell, and hold decisions directly from market data.

Why is trading difficult?

→ Markets are noisy and unpredictable

→ Timing decisions have delayed consequences

→ Transaction costs eat into profits

Why Reinforcement Learning?

→ Learns from experience, not fixed rules

→ Optimizes for long-term profit

→ Adapts to changing patterns

02

Data & Features

DATA SOURCE

Yahoo Finance

2015 - 2024

ASSET USED

- **SPY** - S&P 500 ETF

TRAIN / TEST

80 / 20

18 NORMALIZED INPUT FEATURES

PRICE DATA

Open, High, Low, Close, Volume

MOMENTUM

RSI (14-day), OBV, Returns

TREND

MACD, Signal, Histogram, SMA 20/50

VOLATILITY

Bollinger Bands, ATR, VIX

EXTERNAL MACRO SIGNALS

VIX (market fear/volatility) and TNX (interest rates) provide broader economic context beyond just price action.

Trading Environment

ACTION SPACE

0	1	2
HOLD	BUY	SELL

STATE VECTOR (21 DIMENSIONS)

18 Technical Features + Position + Cash + Holdings

REWARD FUNCTION

reward = (Δ portfolio value) × 100

Scaled percentage change encourages profit-seeking behavior

TRADING RULES

Transaction cost: **0.1%**

Invalid action penalty: **-0.1**

Hold cost: **-0.0001**

ACTION MASKING

Invalid actions are blocked: can only BUY when flat, SELL when holding.

What is a DQN?

Deep Q-Network (DQN) combines Q-Learning (a classic RL algorithm) with deep neural networks. The network learns to predict the **expected future reward** for each possible action, then picks the action with the highest predicted value.

1

OBSERVE

Agent sees current market state (prices, indicators, position)

2

DECIDE

Neural network outputs Q-values for Hold, Buy, Sell

3

ACT

Execute action with highest Q-value (or explore randomly)

4

LEARN

Update network weights based on reward received

Q-VALUE

Q(state, action) = Expected total future reward if we take this action now

BELLMAN EQUATION

$Q = \text{reward} + \gamma \times \max(\text{future } Q)$..... learns to look ahead, not just immediate reward

05

Network Architecture

NEURAL NETWORK STRUCTURE

21 → 128 → 128 → 3

INPUT | ReLU | ReLU | Q-VALUES

TARGET NETWORK

Separate network updated every 1000 steps prevents oscillation

EXPERIENCE REPLAY

100K buffer stores past experiences for random sampling

HYPERPARAMETERS

Learning Rate	0.001
Discount (γ)	0.99
Batch Size	64
ε Decay	1.0 → 0.01

EXPLORATION

ε-greedy: starts random (explore), gradually shifts to learned policy (exploit)

STABILITY

Gradient clipping (max 1.0) prevents exploding gradients

06

Training Comparison

METRIC	100 eps	250 eps	500 eps ★	1000 eps
Agent Return	8.85%	8.25%	**38.46%**	0.85%
Final Value	$10,885	$10,825	**$13,845.77**	$10,085
Max Drawdown	-1.70%	-2.33%	-8.24%	-5.51%
Sharpe Ratio	1.205	1.084	**1.442**	0.115
Buy & Hold Return	51.77%	51.77%	**51.77%**	51.77%
Total Trades	12	48	151	112

Key Finding: 500 episodes is optimal. Training beyond this causes **overfitting**. The model memorizes training data and performs worse on new data.

07

Best Model Results

AGENT RETURN (500 EPS)

38.46%

BUY & HOLD

51.77%

FINAL VALUE

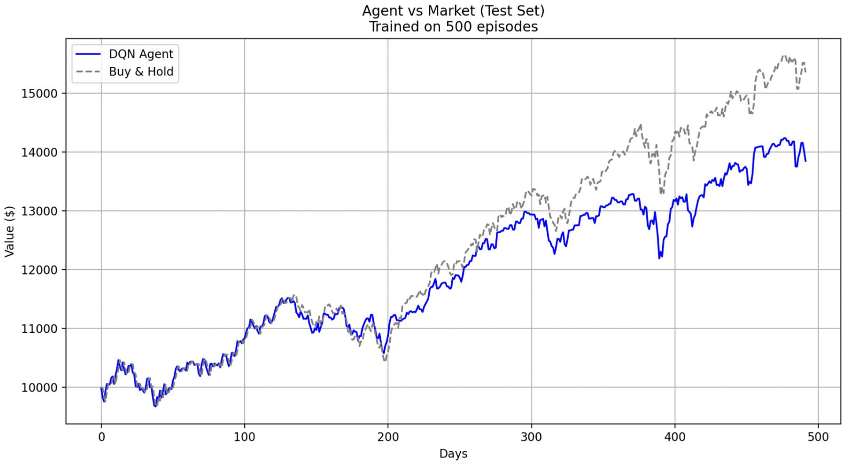
$13845.76

Total Trades Executed **151**

Important Context

Raw returns underperform buy-and-hold, but this comparison misses the risk story. The agent trades actively while buy-and-hold takes full market exposure.

PORTFOLIO VALUE OVER TIME



Thank You!