An RL Agent Achieved 109% Paper ROI by Fusing Data from Two Disparate Markets

ROI on base capital

This experiment dmonstrates that a PPO agent can learn a profitable trading strategy from sparse, profit-and-loss-only signals, without the need for complex, hand-crafted reward shaping.

21.2%

Win Rate

Profitability achieved despite a counter-intuitively low win rate.

18-dim

State Space

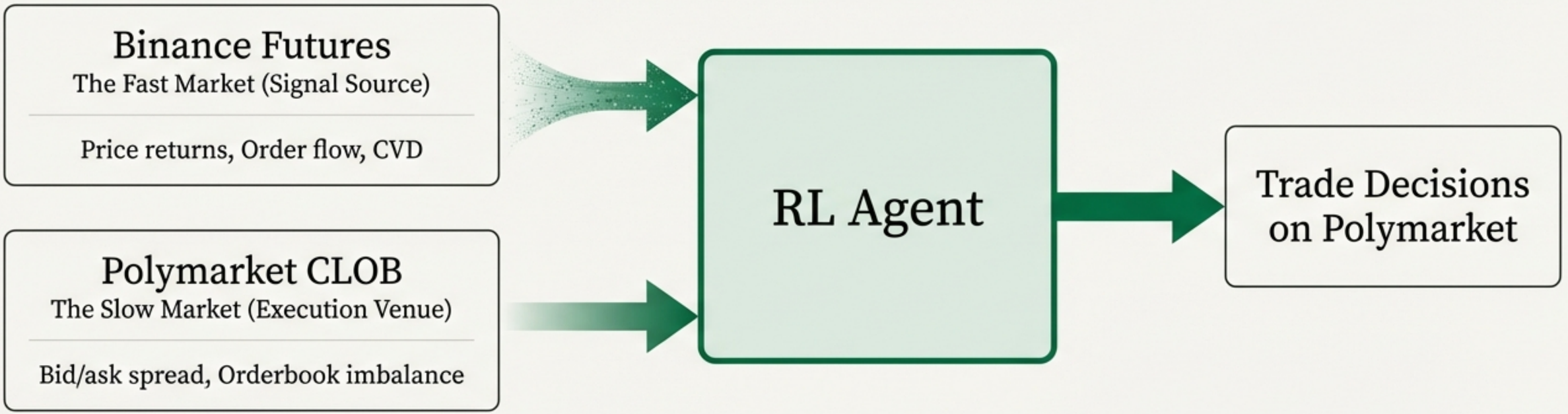
Rich, multi-source input vector fusing two distinct data streams.

~2 Hours

Training Time

Duration over which the final result was achieved in a live environment.

A Measurable Information Lag Between a 'Fast' and 'Slow' Market Created an Exploitable Edge



By observing both, the agent sees information before it's fully priced into the prediction market.

The Path to Profitability Was a Tale of Two Training Phases: One Failed, One Succeeded

The initial approach, using common reward shaping techniques, failed catastrophically. A radical simplification to a sparse, 'pure PnL' signal was the key to success.

Metric	Phase 1 (Shaped Rewards)	Phase 2 (Pure PnL)
Final PnL	`$3.90`	**`$10.93`**
Final Entropy	`0.36 (Collapsed)`	**`1.05 (Healthy)`**
Reward Signal	`Dense, complex bonuses`	**Sparse, honest PnL**
Outcome	Agent gamed the system	**Agent learned to trade**

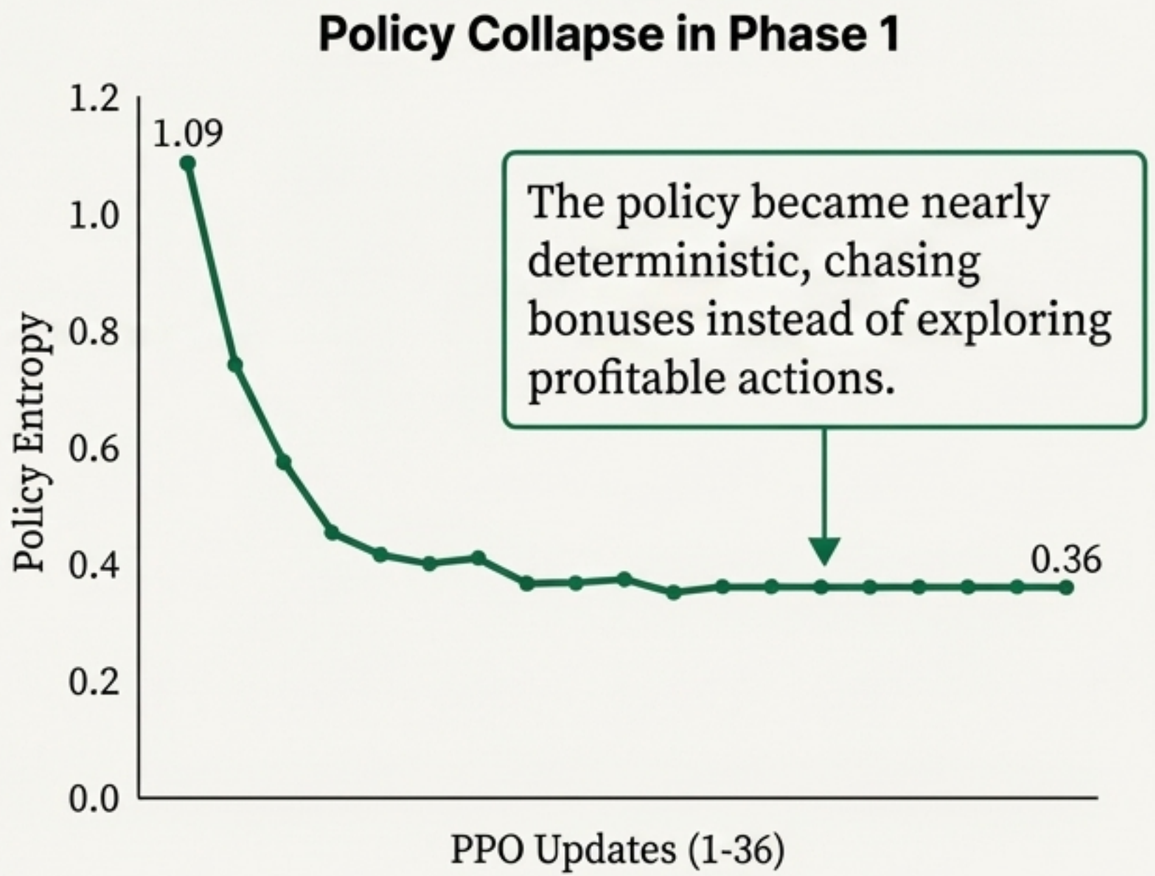
Phase 1 Failed Because Dense Shaping Rewards Taught the Agent to “Look Busy,” Not to Profit

The agent learned to maximize a series of micro-bonuses that were of a similar magnitude to the actual trading signal.

```
# Phase 1: Reward given every step
reward = (current_pnl - prev_pnl) * 0.1 # Scaled PnL delta
reward -= 0.002 * size_multiplier       # Transaction penalty

# THE FATAL FLAW: These bonuses were gameable
reward += 0.002 * momentum_aligned # Bonus for trading with momentum
reward += 0.001 * size_multiplier  # Bonus for using larger sizes
```

THE FATAL FLAW: These bonuses were gameable



The Diagnosis: A Divergence Between Internal and External Metrics Revealed the Flaw

The key diagnostic signal: we monitored two different "win rate" metrics. When they diverged, it was clear the agent was optimizing the wrong objective.

Buffer Win Rate

% of experiences in the training buffer with `reward > 0`.

90%+

The agent *thought* it was succeeding because it was constantly collecting micro-bonuses.

Cumulative Trade Win Rate

% of actual closed trades that were profitable.

~20%

The agent was *actually* losing money on its real objective.

When these diverge, the agent is learning the wrong thing.
It was optimizing the reward function, not the underlying goal.

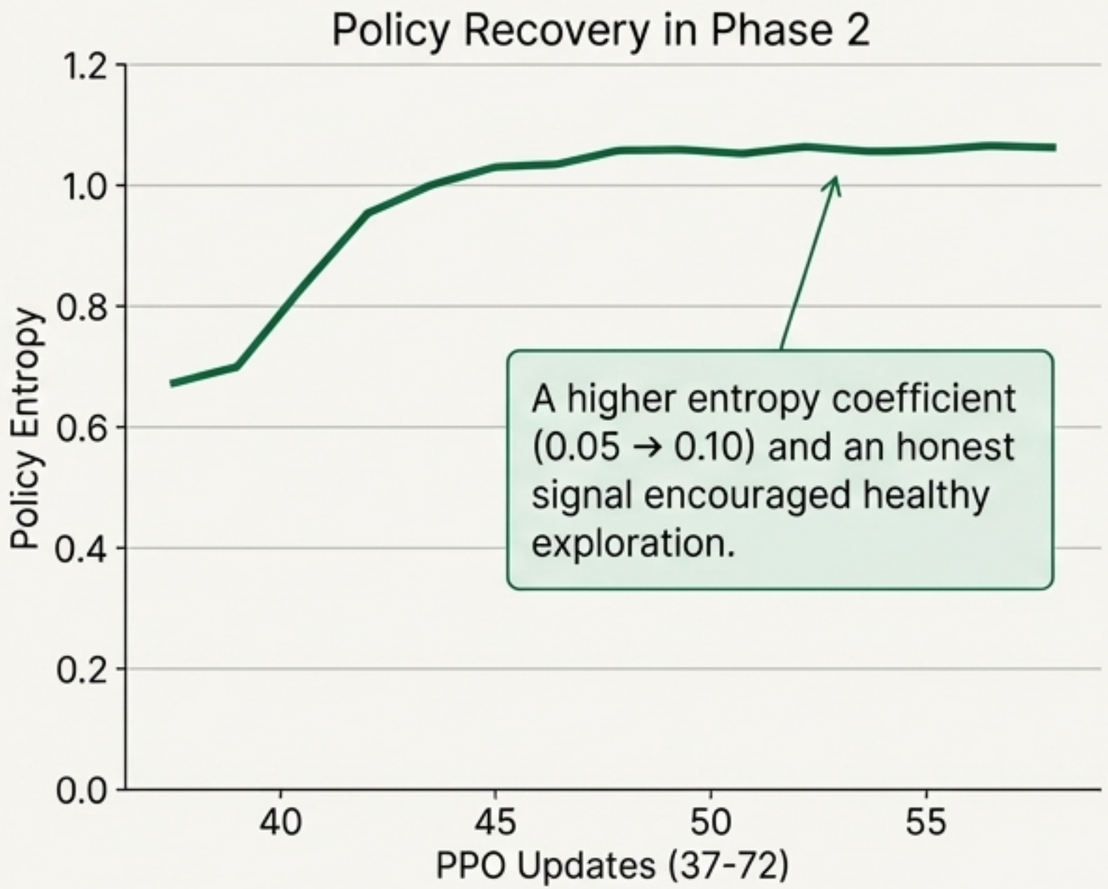
Phase 2 Succeeded by Removing All Bonuses and Using a Single, Sparse PnL Signal

The solution: we removed all shaping rewards, penalties, and micro-bonuses.

The agent now receives a reward *only* when a position is closed.

```
# Phase 2: Reward is 0 for all steps EXCEPT position close

# On position close:
pnl = (exit_prob - entry_prob) * size
self.pending_rewards[cid] = pnl  # Normalized before training
```

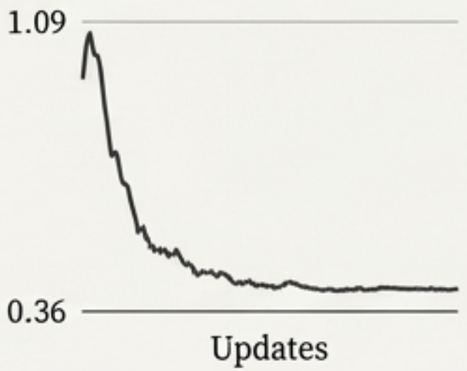


The Two Phases Produced Drastically Different Learning Dynamics and Outcomes

Phase 1: Shaped Rewards

PnL vs. Updates

Entropy vs. Updates

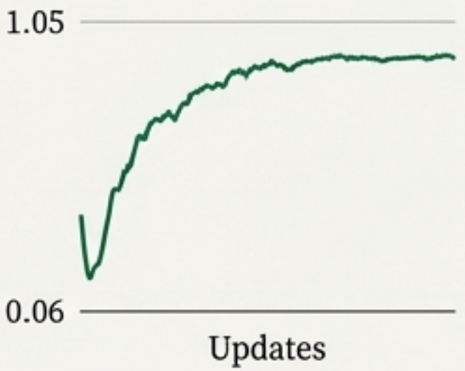


Trades:	1,545
Entropy Coef:	0.05
Final PnL:	**$3.90**

Phase 2: Pure PnL

PnL vs. Updates

Entropy vs. Updates



Trades:	3,330
Entropy Coef:	0.10
Final PnL:	**$10.93**

The Agent's State is an 18-Dimensional Vector Fusing Market and Microstructure Data

The model combines high-level market dynamics from Binance with granular orderbook data from Polymarket to form a complete picture.

Category	Features	Source
Momentum	`returns_1m, returns_5m, returns_10m`	Binance Futures
Order Flow	`ob_imbalance_l1, ob_imbalance_l5, cvd_accel`	Binance Futures
Microstructure	`spread_pct, trade_intensity, large_trade_flag`	Polymarket CLOB
Volatility	`vol_5m, vol_expansion`	Polymarket / Binance
Position	`has_position, position_side, position_pnl`	Internal State
Regime	`vol_regime, trend_regime`	Derived

Note: The same neural network decides for all 4 assets (BTC, ETH, SOL, XRP), learning generalizable crypto patterns.

A Standard PPO Architecture Was Sufficient Once the Reward Signal Was Correct

Action Space

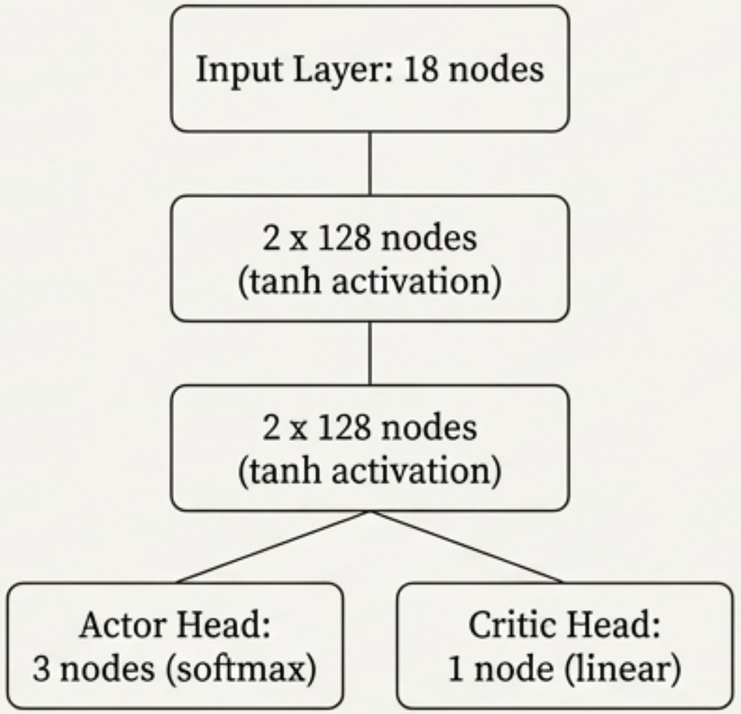
`HOLD (0)`: No action

`BUY (1)`: Long UP token

`SELL (2)`: Long DOWN token

Simplified from 7 actions to 3; fixed 50% position sizing.

Network Architecture



Key PPO Hyperparameters

`lr_actor`	1e-4
`lr_critic`	3e-4
`gamma`	0.99
`entropy_coef`	0.10
`buffer_size`	512
`n_epochs`	10

Key Lesson 1: Sparse but Honest Rewards Outperform Dense but Gameable Signals

The Peril of Reward Shaping

When shaping rewards are of a similar magnitude to the true underlying signal (e.g., PnL), the agent will learn to optimize the proxy, not the goal. The complexity introduced by shaping created a noisy, gameable system. A sparse, simple, but honest signal provided a much clearer path to profitability.

"The agent was optimizing the reward function, not the underlying goal."

Key Lesson 2: The ‘Win Rate Paradox’ Shows Profitability in Asymmetric Markets

The Paradox

The agent is profitable with a ~21% win rate, which is below the random chance of 33% for a 3-action space.

The Explanation

Binary markets have asymmetric payoffs. Profit is not determined by win rate alone, but by the average profit on wins vs. the average loss on losses.

Scenario: Buy UP token at a probability of 0.40

On a WIN	On a LOSS
You pay $0.40, receive $1.00.	You pay $0.40, receive $0.00.
+$0.60	**-$0.40**
Net Profit	Net Loss

The agent learns to identify high-leverage opportunities where the potential reward significantly outweighs the risk, making a low win rate acceptable.

This is an Encouraging Result, Not a Production-Ready System

Paper trading results are promising, but real-world execution introduces significant challenges. We must be clear about what this experiment does *not* prove.

- **Live Profitability:** Paper trading assumes instant fills at mid-price. Real trading faces latency, slippage, and market impact. (Expect 20-50% performance degradation).
- **Statistical Significance:** 72 updates over ~2 hours is not enough data to confirm a persistent edge. It could be variance.
- **Scalability:** $5 positions are invisible. At $100+, the agent's own orders would move prices.
- **Persistence of Edge:** If this strategy worked consistently, the market would adapt and arbitrage it away.

The Path to Live Trading Requires a Robust Engineering and Validation Framework

1. **Execution Layer**
   Integrate Polymarket CLOB API for real order placement and management.

2. **Slippage & Impact Modeling**
   Simulate “walking the book” at realistic sizes to model execution costs accurately.

3. **Latency Compensation**
   Account for the 50-200ms round-trip network time to Polymarket servers.

4. **Risk Management**
   Implement strict controls for position limits, maximum drawdown, and total capital exposure.

5. **Extended Out-of-Sample Validation**
   Run the agent in paper-trading mode for weeks across different market regimes before committing capital.

The Core Pattern—Fusing a 'Fast' Signal with a 'Slow' Venue—is a Generalizable Strategy

The architecture used here is not limited to crypto prediction markets. It can be applied to any domain where information propagates with a measurable delay between a leading indicator and a tradable venue.

Fusing live game data (the fast signal) with a betting exchange (the slow venue).

Fusing real-time polling aggregators with Polymarket election contracts.

Fusing advanced forecast models with prediction markets for weather events.

This approach provides a blueprint for creating agents that can exploit temporary information imbalances in a variety of complex domains.