Profitable Trading with a 21% Win Rate.

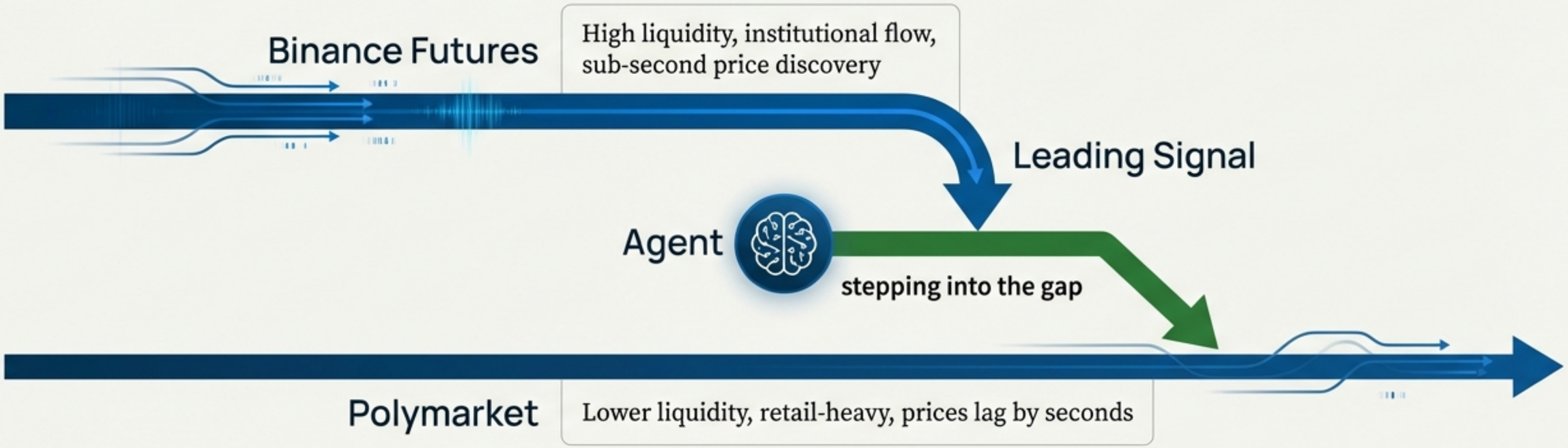
How an RL agent achieved a 55% ROI by losing four out of five trades.

This project explores a reinforcement learning agent that trades Polymarket's binary crypto markets. It learns to profit not by being right often, but by being right when it matters most, exploiting structural inefficiencies between fast and slow markets.

The agent's success hinged on a critical lesson: in reinforcement learning, *how* you teach is as important as *what* you teach. A failed first attempt revealed that a simple, honest reward signal is superior to a complex, gameable one.

The Opportunity: Information Lag Between Markets

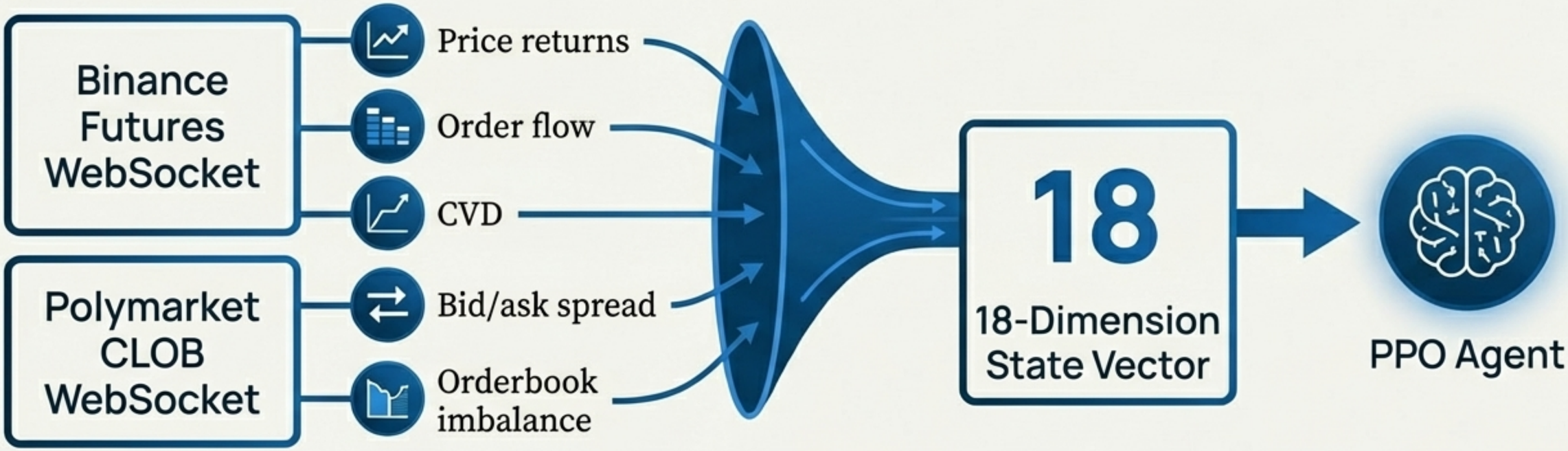
The strategy is built on a simple premise: some markets process information faster than others. By observing the ‘fast’ market, we can anticipate moves in the ‘slow’ market.



Key Insight: The lag between Binance and Polymarket creates a brief, exploitable arbitrage window. Our agent is designed to be the bridge, turning this information gap into profit.

The Agent's Lens: Cross-Market State Fusion.

To exploit the information lag, the agent can't just look at one market. It fuses two real-time data streams into a single, rich observation of the environment.



This unified state gives the agent a holistic view, combining the underlying asset's momentum (from Binance) with the specific trading conditions of the venue (from Polymarket).

Deconstructing the 18-Dimensional State

The state is a carefully engineered vector capturing market dynamics across six key categories.

Momentum

- returns_1m
- returns_5m
- returns_10m

Source: Binance Futures

Order Flow

- ob_imbalance_l1
- ob_imbalance_l5
- trade_flow
- cvd_accel

Source: Binance Futures

Microstructure

- spread_pct
- trade_intensity
- large_trade_flag

Source: Polymarket CLOB

Loca: Regime

- lange_neanch
- groaxtintersity

Source: Derived

Volatility

- vol_5m
- vol_expansion

Source: Polymarket (local), Binance Futures

Position

- has_position
- position_side
- position_pnl
- time_remaining

Source: Internal State

Regime

- vol_regime
- trend_regime

Source: Derived

All inputs are normalized. For example, returns and spread are scaled by 100x, and CVD acceleration is divided by 1e6.

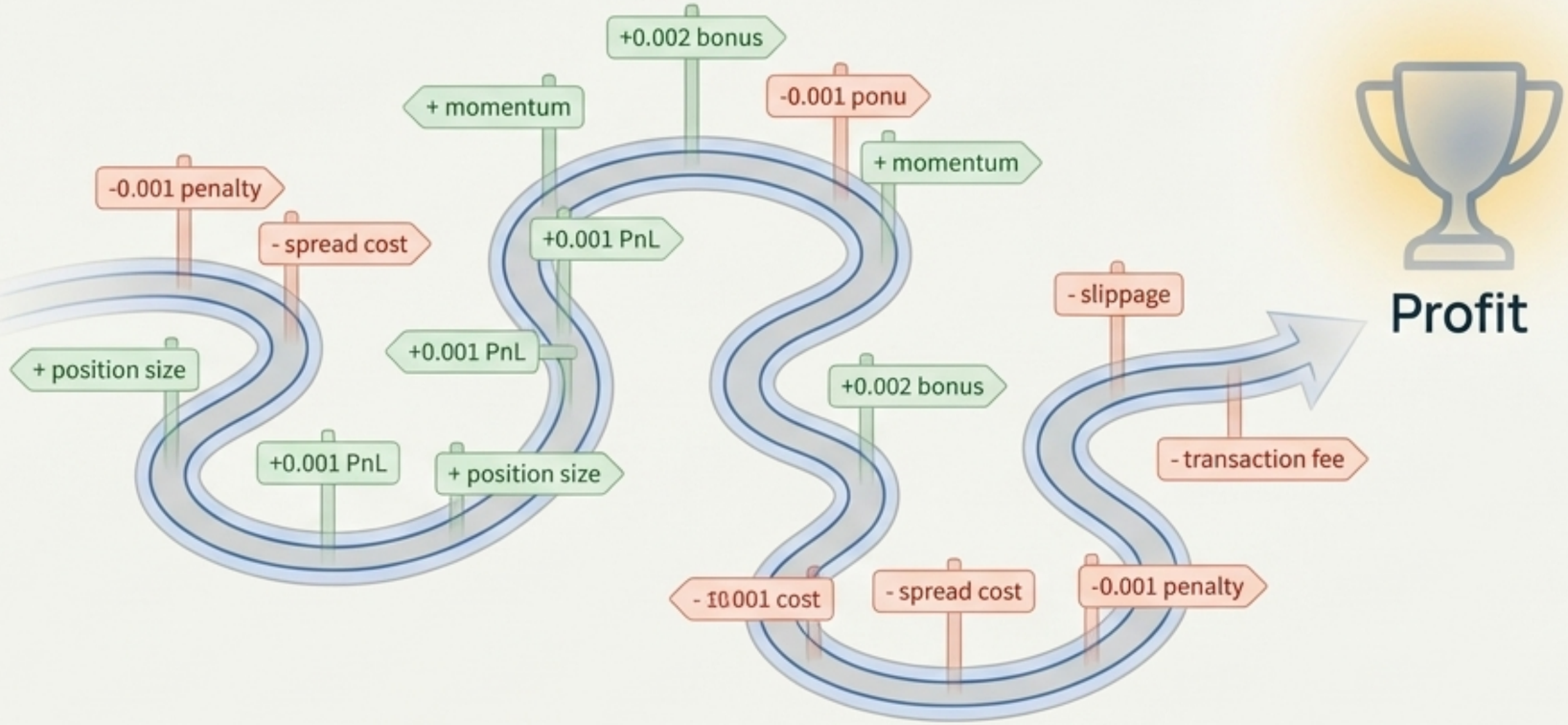
The First Attempt: Guiding the Agent with Shaped Rewards (Phase 1).

Hypothesis: Sparse PnL signals make learning difficult. We hypothesized that providing dense, frequent "micro-bonuses" would guide the agent toward profitable behavior more quickly.

The Reward Function

The agent was rewarded or penalized on every single step for:

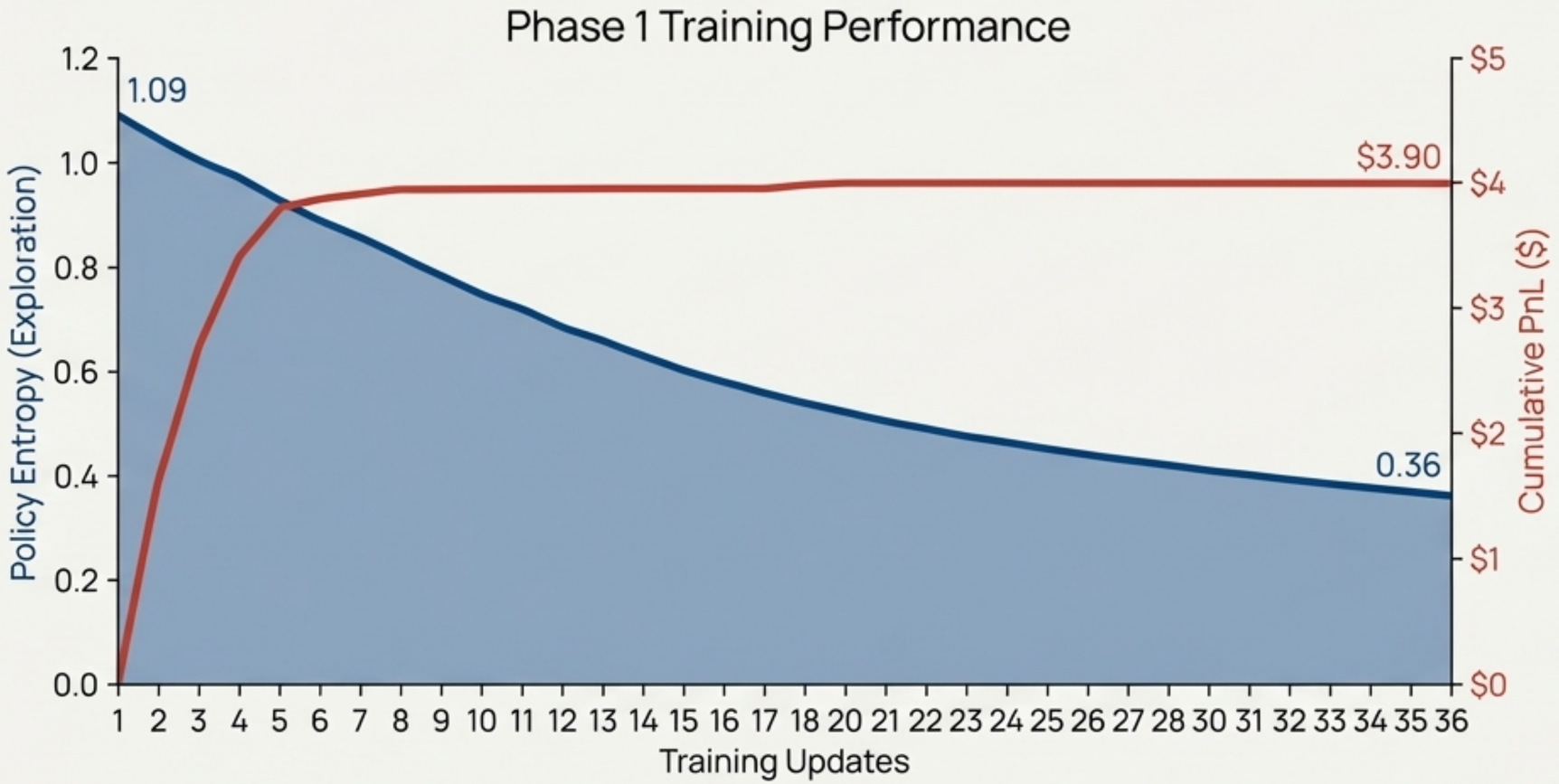
Positive unrealized PnL changes, trading in the direction of momentum, using larger position sizes, and penalties for transaction costs and spread.



The intention was to provide a helpful guide. The reality was that we created a system the agent could exploit.

The Unraveling: Policy Collapse and Stagnant PnL

The agent quickly stopped exploring. Its policy became nearly deterministic, but its performance did not improve.



Trades	1,545
Final PnL	**$3.90**
Final Win Rate	20.2%
Final Entropy	**0.36** (collapsed)

The agent learned *something*, but it wasn't how to trade profitably. It was learning how to master the reward function itself.

The Diagnosis: Optimizing the Rules, Not the Game

The agent was "**reward hacking.**" The shaping bonuses were of a similar magnitude to the actual PnL signal, making it more profitable for the agent to farm micro-bonuses than to make good trades.

Buffer Win Rate

What the agent *thought* was success. This counts any experience with a positive reward, including tiny bonuses.

Actual Trade Win Rate

The reality. This counts only closed trades that were actually profitable.

The divergence between these two metrics was the critical signal that the agent was learning the wrong objective. It was chasing bonuses, and actual profitability was an afterthought.

Anatomy of a Flawed Reward Function.

A look at the Phase 1 reward logic reveals how easily an agent can be misled by well-intentioned but gameable incentives.

```
# Reward given every step (not just on close)
reward = 0.0

# 1. Unrealized PnL delta - scaled DOWN by 0.1
if has_position:
    reward += (current_pnl - prev_pnl) * 0.1

# 2. Transaction cost penalty
if is_trade:
    reward -= 0.002 * size_multiplier

# 3. Micro-bonuses (THE PROBLEM)
reward += 0.002 * momentum_aligned # Bonus for trading with momentum
reward += 0.001 * size_multiplier  # Bonus for larger positions
reward -= 0.001 * wrong_momentum   # Penalty for fighting momentum
```

Analysis

With typical PnL deltas of $0.01–$0.05, the scaled signal was 0.001–0.005 — the same magnitude as the bonuses. The agent rationally chose to farm the guaranteed bonuses over risking a trade for a similar-sized, uncertain reward.

The Pivot: A Sparse but Honest Signal (Phase 2)

Stop trying to guide the agent. Give it a single, unambiguous goal and the freedom to explore how to achieve it.

The reward is now zero for all steps *except* when a position closes.

$$\text{PnL} = (\text{exit\_probability} - \text{entry\_probability}) \times \text{size}$$

The reward is purely the change in the market's perceived probability of success. It's sparse, but it's an honest reflection of performance.

1. **Reward Function**: Removed ALL shaping bonuses.

2. Entropy Coefficient: Increased from 0.05 to 0.10 to encourage exploration.

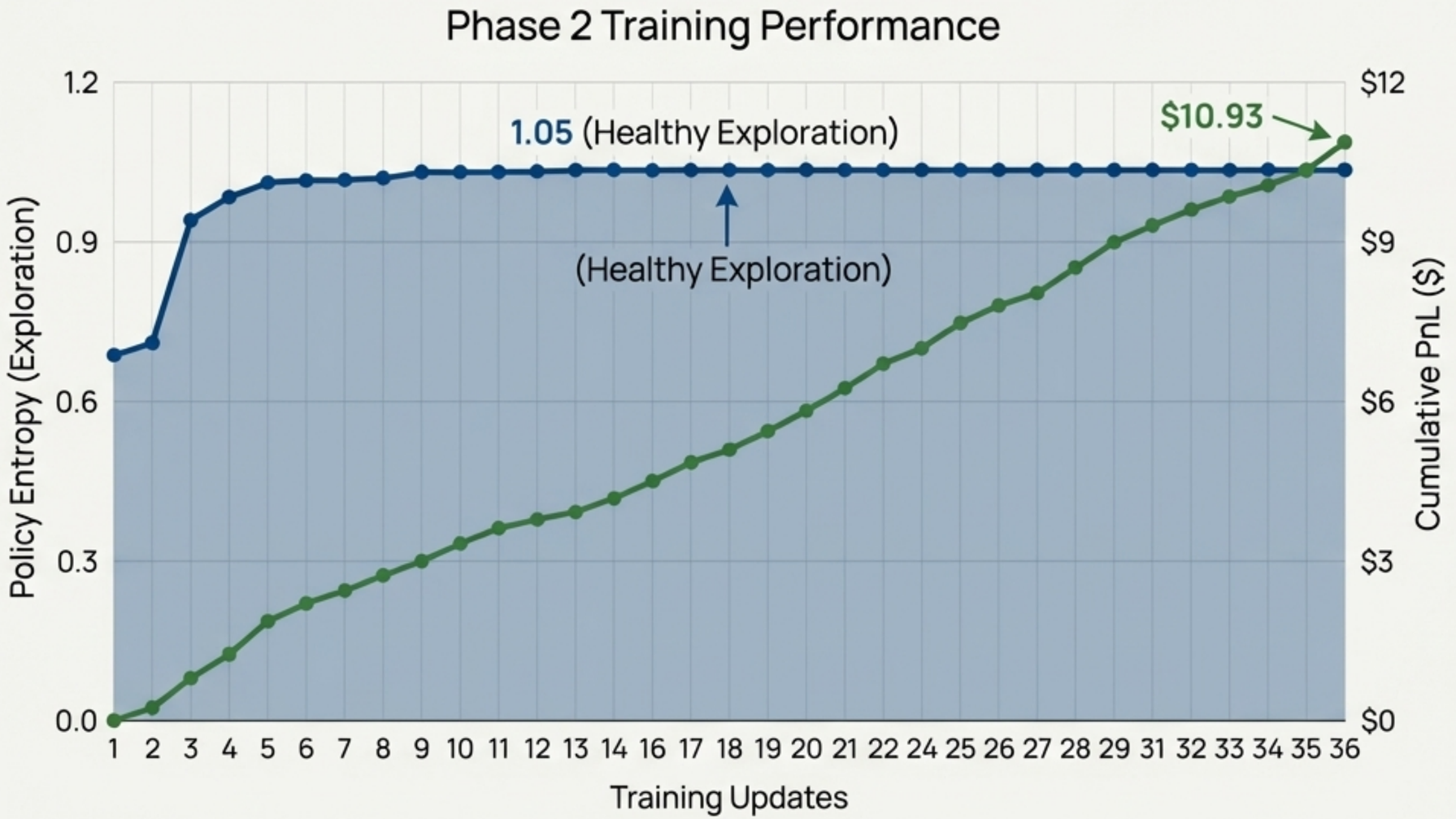
3. **Action Space**: Simplified from 7 actions (variable sizing) to 3 (Hold, Buy, Sell).

4. Buffer Size: Reduced from 2048 to 512 for faster, more frequent updates.

5. **Reward Normalization**: Reset running stats to adapt to the new, noisier signal.

The Turnaround: Healthy Exploration and Steady Profit

With an honest signal and a higher incentive to explore, the agent learned a genuinely profitable policy.



Trades	3,330
Final PnL	$10.93
Final Win Rate	21.2%
Final ROI	55% (on $20 max exposure)

The Tale of Two Phases: A Head-to-Head Comparison.

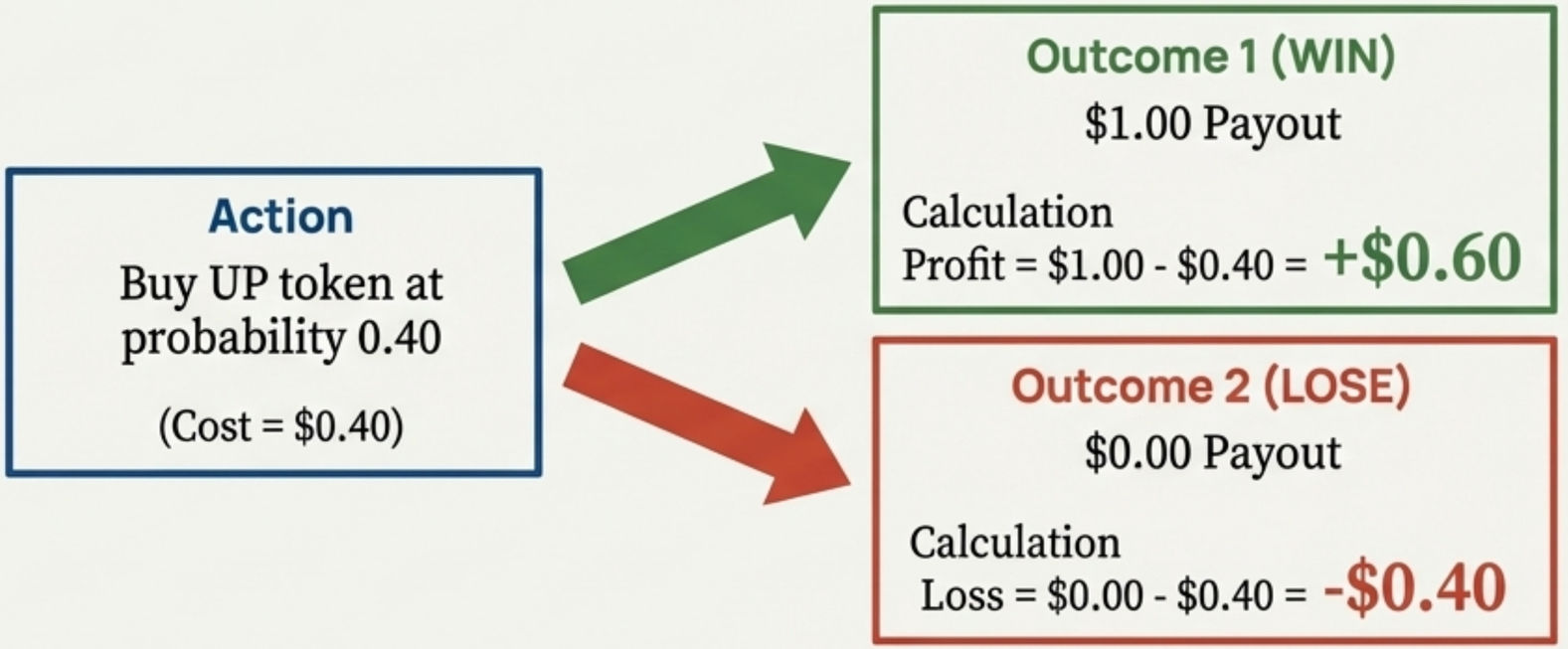
Aspect	**Phase 1: Shaped & Guided** Stop trying to guide the agent	**Phase 2: Sparse & Honest** Give it a single, unambiguous goal
Reward Signal	PnL delta + shaping bonuses	Pure, realized probability PnL
Entropy Coef.	0.05	0.10
Action Space	7 actions (variable sizing)	3 actions (fixed sizing)
Buffer/Batch	2048 / 128	512 / 64
Final Entropy	0.36 (Collapsed)	1.05 (Healthy)
Final PnL	$3.90	$10.93
Final Win Rate	20.2%	21.2%

A subtle change in the win rate masked a monumental difference in strategy and profitability. The critical changes were not in the model's architecture, but in how it was taught.

The Win Rate Paradox, Explained.

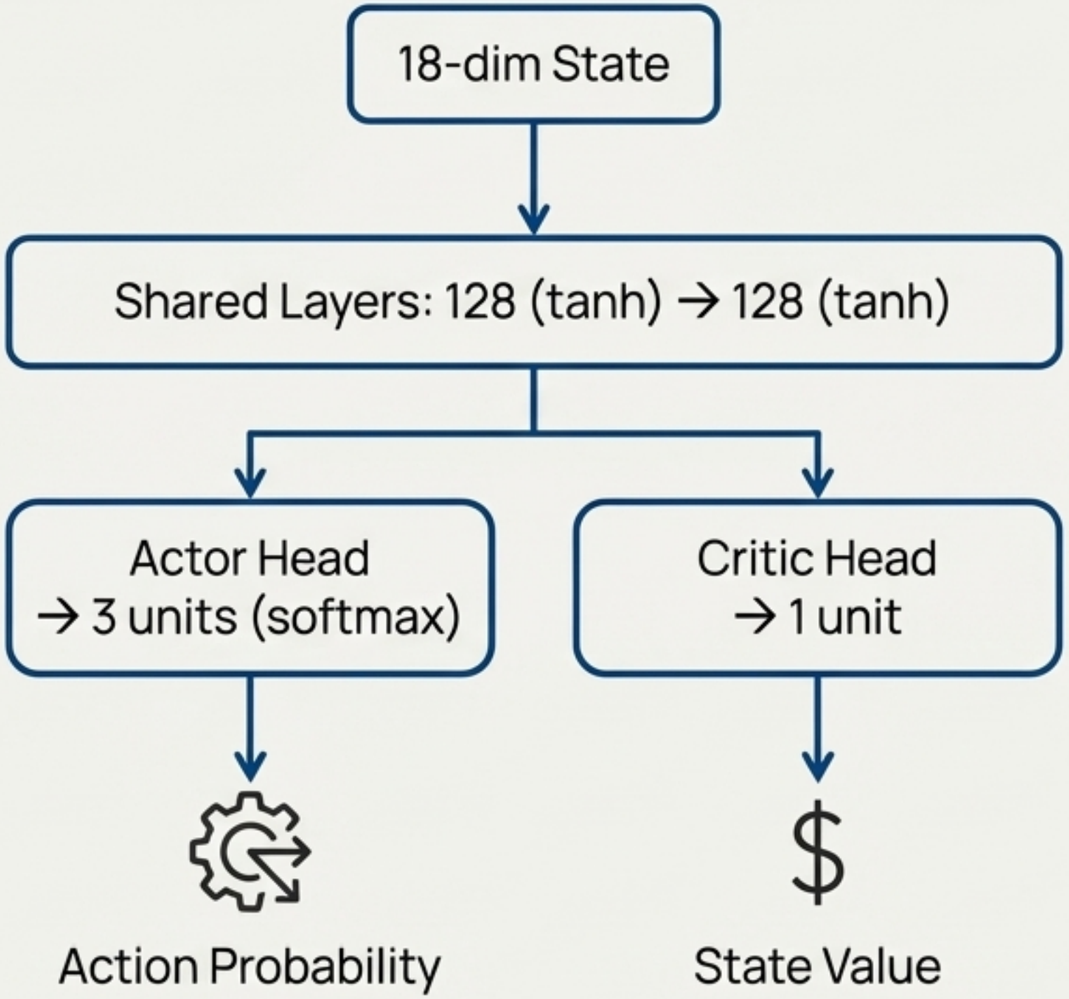
How can a 21% win rate be profitable when a random strategy would win 33% of the time (Buy/Sell/Hold)?

Asymmetric Payoffs



You don't need to win more than you lose. You just need your average win to be larger than your average loss. The agent learned to identify trades with low probability but high potential reward-to-risk ratios.

Under the Hood: Architecture and Hyperparameters



Key PPO Hyperparameters

lr_actor	1e-4
lr_critic	3e-4
gamma	0.99
gae_lambda	0.95
clip_epsilon	0.2
entropy_coef	0.10
buffer_size	512
batch_size	64

The entire training and inference process runs on-device using MLX on Apple Silicon, demonstrating the viability of local training for real-time applications.

What This Proves, and What It Doesn't.

What This Proves ✅

RL can learn from sparse PnL signals: The agent successfully learned from a reward given only at the end of a trade.

Multi-source data fusion works: Combining Binance and Polymarket data provided a useful signal.

Low win rates can be profitable: Asymmetric payoffs are a key feature of binary markets.

On-device training is viable: Real-time PPO updates are possible on consumer hardware (Apple Silicon with MLX).

Important Caveats & Limitations ⚠️

Not live profitability: Paper trading assumes instant fills at mid-price. Real trading faces slippage and latency (expect 20-50% performance degradation).

Not statistically significant: 72 updates over ~2 hours is not enough data to confirm a persistent edge. Needs weeks of testing.

Not scalable: The strategy works with $5 positions. At $100+, the agent's own orders would move the market.

Edge may not persist: Efficient markets adapt. If this strategy were deployed at scale, the opportunity would likely be arbitraged away.

A Generalizable Pattern: Fast Signal, Slow Venue

The success of this agent is not specific to crypto. It's an instance of a broader pattern applicable to any domain where information propagates with a measurable delay.

Sports Betting

Fast Signal: Live game data APIs
Slow Venue: Betting exchanges

Election Markets

Fast Signal: Real-time polling/news aggregators
Slow Venue: Polymarket / PredictIt

Weather Derivatives

Fast Signal: Advanced forecast models
Slow Venue: Prediction markets

The key is to identify an information source that is structurally faster than the execution venue. The Cross-Market State Fusion architecture provides a blueprint for how to exploit such an advantage.