Category 1: Data Integrity Issues (Look-Ahead Bias)

Problem Statement:

The model achieved unrealistic performance (Sharpe Ratio: 210, Total Return: 8,145%) due to multiple layers of look-ahead bias where future information leaked into the prediction process.

Technical Details:

1. **Feature Engineering Leakage**: Original features had 0.12-0.53 correlation with future returns, indicating they contained forward-looking information
2. **Environment Implementation Bug**: The observation buffer at step `t` included features from the current timestep before making predictions, allowing the model to "see" the bar it was supposed to predict
    - Bug location: `transformer_a2c_env.py:147`
    - Code: `new_obs = self.data[self._current_step]` should be `self.data[self._current_step - 1]`

Impact:

Model learned to exploit data leakage rather than genuine market patterns, rendering all validation metrics meaningless.

Category 2: Reward Calculation Bugs (Metric Reliability)

Problem Statement:

The reward calculation system had implementation bugs that masked actual performance and prevented proper gradient signals.

Technical Details:

1. **Sharpe Ratio Clipping**: Hardcoded `np.clip(sharpe, -10, 10)` suppressed extreme values that should have signaled bugs
2. **Sortino Edge Case**: Returned arbitrary value (10.0) when insufficient downside samples existed, creating false positive signals
3. **Missing Validation Logging**: No granular metrics during validation made it impossible to diagnose issues

Impact:

Impossible to distinguish between genuine performance and numerical artifacts, delayed bug detection by multiple training runs.

Category 3: Exploration-Exploitation Failure (Policy Collapse)

Problem Statement:

After fixing all data leakage, the model converged to a degenerate policy (100% FLAT positions) with zero variance in actions, preventing any learning.

Technical Details:

1. **Entropy Coefficient Too Low**: `entropy_coef=0.01` insufficient to maintain exploration in a sparse reward environment
2. **Bootstrap Failure**: Model converges to safe FLAT policy before experiencing meaningful rewards from LONG/SHORT positions
3. **Feedback Loop**: FLAT → reward=0 → near-zero gradients → policy stays FLAT → critic learns V(s)≈0 everywhere

Current Metrics:

- Validation Sharpe: 0.0000
- Returns mean: 0.000000
- Returns std: 0.000000
- All 79,049 validation samples have identical FLAT action

Impact:

Model learns a trivial non-trading policy that is locally optimal but globally useless.

Summary for Expert Review:

We encountered a **cascading failure pattern**:

1. Initial data leakage bugs created false confidence (8,145% returns)
2. After fixing leakage, reward calculation bugs masked the underlying exploration problem
3. With clean data and metrics, revealed fundamental issue: insufficient entropy for sparse reward RL problem

Key Question for Expert:

In financial RL with sparse rewards (only profitable when correctly timing positions), what is the recommended approach to prevent entropy collapse while maintaining sample efficiency? Should we prioritize:

- Higher entropy coefficient (0.1 vs 0.01)?
- Curriculum learning (train on synthetic profitable episodes first)?
- Different algorithm entirely (PPO, SAC, or off-policy methods)?