# Analysis of Synthetic-to-Real Transfer Failure in RL-Based Position Sizing: Diagnosis, Theory, and Remediation Strategies

## Executive Summary

The deployment of Deep Reinforcement Learning (DRL) in high-frequency and algorithmic trading is frequently obstructed by the "reality gap"—the catastrophic degradation of performance when agents trained in simulated environments are deployed on real-world data. The failure mode described in the case study—a 91% performance collapse from a validation Sharpe ratio of +0.0334 to a test Sharpe of +0.003—is a paradigmatic example of **observational overfitting** to synthetic artifacts. The fact that five independent model seeds converged to an identical, poor performance level indicates that the ensemble did not learn robust market dynamics but rather exploited idiosyncratic statistical invariants present only in the synthetic generator.

This report provides an exhaustive analysis of this failure, dissecting the mechanisms of sim-to-real collapse in financial time series. It synthesizes literature from 2020–2025 to propose a remedial architecture that integrates **Regime-Switching Synthetic Data Generation**, **Entropy-Maximized Domain Randomization (DORAEMON)**, and **Diversity-Regularized Ensembles (MED-RL)**. Furthermore, the analysis suggests that pure end-to-end RL for position sizing is theoretically flawed due to the high variance of gradient estimates in stochastic environments, advocating instead for a **Hybrid RL-Kelly Criterion** approach.

The following sections detail the theoretical underpinnings of the failure, evaluate the efficacy of state-of-the-art generative models (TimeGAN, SigCWGAN, Regime-Switching HMMs), and provide a concrete implementation roadmap to achieve the target Sharpe ratio > +0.03 on real test data.

## 1. The Pathology of the Sim-to-Real Gap in Financial RL

The observed phenomenon, where training and validation performance on synthetic data remain high while test performance on random (or real) data collapses, suggests a fundamental misalignment between the source domain (synthetic simulator) and the target domain (real market dynamics). This section dissects the specific failure patterns identified in the user's experiment through the lens of recent academic findings on distributional shift and observational overfitting.

### 1.1 The Mechanism of Synthetic Overfitting

In standard supervised learning, overfitting is characterized by high training performance and low validation performance. However, in the user's case, validation performance was also high (+0.0334), yet the model failed on the test set (+0.003). This anomaly implies that the **validation set was not out-of-distribution (OOD)** relative to the training set. Both were likely drawn from the same synthetic generator which possessed a static set of statistical properties (artifacts) that do not exist in the real world.1

RL agents are notoriously opportunistic; they function as sophisticated pattern matchers that exploit any predictive signal to maximize reward.3 If the synthetic generator produces data using a deterministic process (e.g., a fixed set of differential equations or a simplified GARCH model) without sufficient stochasticity, the LSTM network will memorize the "fingerprint" of the generator rather than the economic principles of price movement.2 The agent learns the *simulator*, not the *market*.

Recent research into "observational overfitting" in reinforcement learning clarifies this distinction. When an MDP (Markov Decision Process) is defined by a synthetic generator, the state transitions $P(s'|s, a)$ and reward functions $R(s, a)$ are governed by the generator's mathematical rules. An LSTM with sufficient capacity, such as the one used in the user's 5-model ensemble, can effectively reverse-engineer these rules. For instance, if the generator uses a specific mean-reversion parameter $\theta$ or a volatility clustering coefficient $\alpha$, the agent learns to anticipate price movements based on these fixed parameters. When the agent is transferred to a real market environment where these parameters drift or are governed by entirely different (and unobserved) latent factors, the policy collapses because the specific predictive features it relied upon are absent or misleading.2

### 1.2 Ensemble Diversity Collapse

The convergence of all five models (Seeds 42, 123, 456, 789, 101112) to an identical test performance of +0.003 is a critical diagnostic indicator. In a healthy ensemble, different seeds should initialize the networks in different basins of attraction, leading to diverse policies that, when averaged, reduce variance and improve generalization.6 The "identical convergence" phenomenon suggests that the loss landscape of the synthetic environment was deceptively simple or convex, funneling all agents into the same local optimum. This local optimum represents a policy that is optimal *only* for the specific artifacts of the synthetic generator. When exposed to real data, this specialized policy fails universally. This is known as **Ensemble Diversity Collapse**, where the correlations between ensemble members approach 1.0, negating the benefits of ensembling.6

The failure of the dropout regularization (0.2) to prevent this collapse is instructive. Dropout is designed to prevent co-adaptation of feature detectors, but it does not inherently force the network to learn robust, generalized dynamics. In the context of the synthetic generator, dropout essentially forces the network to learn the generator's rules "harder" and more redundantly, rather than forcing it to ignore the generator's idiosyncratic artifacts. Without a specific diversity-inducing objective function, the ensemble members are drawn toward the same dominant spectral features of the synthetic time series, resulting in a monolithic failure mode.2

### 1.3 The Validation-to-Test Gap: A Crisis of Distributional Shift

The 91% drop in Sharpe ratio highlights a **Distributional Shift** problem. The synthetic data likely lacked the "stylized facts" of real financial time series: fat tails, volatility clustering, and non-stationarity.8 If the synthetic generator assumed Gaussian returns or stationary volatility (homoscedasticity), the RL agent would learn aggressive position sizing strategies that maximize returns under those safe assumptions. When applied to real markets (or test data with realistic properties), these aggressive strategies encounter "black swan" events or volatility spikes they were not trained to handle, leading to massive drawdowns and performance collapse.10

This gap is exacerbated by the nature of financial reinforcement learning. Unlike robotics, where the physics engine (gravity, friction) is a close approximation of reality, financial simulators are often fundamentally distinct from the target domain. The synthetic data likely contained "spurious correlations"—patterns that appear predictive in the generated history but have no causal validity. For example, the generator might accidentally couple high volume with positive returns perfectly due to a simplified interaction term in its code. In the real world, this correlation is noisy and regime-dependent. The agent, having learned to bet maximally on this spurious signal, is punished severely when the relationship breaks down in the test set.12

## 2. Research Question 1: Sim-to-Real Transfer in Financial RL

Bridging the gap between simulation and reality is the central challenge in deploying RL for finance. The literature from 2020–2025 emphasizes that simple domain randomization is often insufficient; more structured approaches are required to ensure robustness.

### 2.1 The Reality Gap and Domain Randomization

The "Reality Gap" refers to the discrepancy in physical or statistical dynamics between the training environment and the deployment environment.1 In robotics, this is solved by randomizing physical parameters (friction, mass). In finance, the equivalent parameters are **drift**, **volatility**, **mean-reversion speed**, **jump intensity**, and **noise levels**.15

Standard Domain Randomization (DR) involves sampling these parameters from fixed uniform distributions. However, recent research indicates that uniform DR can lead to overly conservative policies that fail to exploit specific market inefficiencies. If the randomization is too broad, the agent learns a "do nothing" policy to avoid risk. If it is too narrow, the agent overfits.15 For cryptocurrency markets, which exhibit extreme non-stationarity, simple uniform randomization is particularly ineffective because it fails to capture the *structure* of the uncertainty. The market does not just have "random volatility"; it has volatility regimes that persist and switch.17

### 2.2 Advanced Transfer Techniques

DORAEMON (Domain Randomization via Entropy Maximization):

A significant advancement in this field is DORAEMON.15 This method treats domain randomization as a constrained optimization problem. Instead of manually tuning the range of randomization, DORAEMON automatically shapes the distribution of dynamics parameters to maximize entropy (diversity) while ensuring the agent maintains a minimum success rate.

* **Relevance:** For the user's position sizing model, this means the training environment should not use a single synthetic generator setting. It should dynamically vary the volatility regimes and trend strengths, progressively making the environment "harder" and more diverse as the agent improves. This curriculum learning approach prevents the agent from overfitting to a static set of generator parameters and forces it to learn policies that are robust across a wide range of market conditions.16
* **Mechanism:** DORAEMON maintains a distribution over the simulation parameters $\phi$. It iteratively updates $\phi$ to maximize $H(\phi)$ (entropy) subject to the constraint that the agent's expected performance $J(\pi, \phi)$ remains above a threshold. This ensures the agent is always challenged but never overwhelmed, fostering continuous learning and generalization.19

Adversarial Reinforcement Learning:

Another potent approach is adversarial training, where a "disturber" agent introduces perturbations to the state or reward signal to minimize the primary agent's performance.20 This forces the trading agent to learn robust features that are invariant to small shifts in market distribution.

* **Finding:** Papers show that adversarial training significantly improves robustness against the non-stationarity inherent in financial markets, reducing the validation-to-test gap. By training against an adversary that actively seeks to exploit the agent's weaknesses (e.g., by manipulating input prices or introducing latency), the agent learns to be conservative and verifying, traits essential for survival in real markets.22

### 2.3 Key Findings from Literature

The synthesis of recent literature reveals a hierarchy of effectiveness for sim-to-real techniques in finance:

| **Technique** | **Mechanism** | **Impact on Sim-to-Real** | **Limitation** | **Source** |
| --- | --- | --- | --- | --- |
| **System Identification** | Estimate real parameters to calibrate sim | High precision if model is correct | Fails if market model is wrong | 24 |
| **Domain Adaptation** | Transfer learning from sim to real | Aligns feature distributions | Requires target domain data | 20 |
| **DORAEMON** | Auto-curriculum of parameter entropy | **Best generalization** reported | Computationally intensive | 15 |
| **Adversarial Training** | Min-max game against disturbances | Robust to noise/attacks | Can be overly conservative | 21 |

**Conclusion for Q1:** The user's reliance on a single synthetic generator configuration was the primary failure point. Implementing **DORAEMON** to dynamically vary the parameters of the synthetic generator is the highest-ranked solution to prevent overfitting. This method directly addresses the "Identical Convergence" and "Generator Fingerprinting" issues by ensuring that no single fingerprint persists long enough for the agent to memorize it.

## 3. Research Question 2: Synthetic Data Generation for Trading Models

The quality of the synthetic data is the ceiling of the RL agent's performance. If the data does not capture the complex, non-linear dependencies of real markets, the agent is learning a fantasy. The user's experiment likely utilized a generator that failed to capture the "stylized facts" of financial time series, leading to the massive performance collapse.

### 3.1 Evaluating Generative Models

TimeGAN (Time-series Generative Adversarial Networks):

TimeGAN 25 preserves temporal dynamics by combining a supervised loss (matching next-step predictions) with an unsupervised adversarial loss. While a significant improvement over standard GANs, it has notable limitations in finance.

* **Mode Collapse:** TimeGAN is prone to **mode collapse**, where the generator produces a limited variety of samples, ignoring tail events.8 This explains why the user's models converged to identical performance—they were likely training on a mode-collapsed dataset that lacked diversity. If the generator only produces "average" days, the agent never learns to handle crises.
* **Temporal Artifacts:** TimeGAN can introduce subtle temporal artifacts (periodicity) that are easily exploitable by LSTMs but absent in real data. The agent learns to predict the GAN's next step, not the market's.2

QuantGAN & SigCWGAN:

Newer models address these deficiencies. QuantGAN focuses on capturing "stylized facts" like fat tails and volatility clustering using Temporal Convolutional Networks (TCNs).8 Even more promising is SigCWGAN (Signature-based Conditional Wasserstein GAN).

* **Signature Metrics:** SigCWGAN utilizes the "signature" of the path—a collection of iterated integrals from rough path theory—as the discriminator's metric. This captures the geometric and topological properties of the price path, such as "roughness" and long-range dependency, much better than standard moments.8
* **Finding:** SigCWGAN has been shown to outperform TimeGAN in financial contexts because the signature metric prevents the generation of unrealistic, smooth trajectories. It forces the generator to produce paths with the jagged, fractal nature of real crypto markets.28

Regime-Switching Models (HMM/GMM):

Deep learning generators (GANs) are often "black boxes" that are hard to control. Regime-switching models (e.g., Hidden Markov Models) explicitly model the market as switching between latent states (e.g., "Bull," "Bear," "Sideways," "High Volatility").29

* **Relevance:** Financial markets are non-stationary and regime-dependent. A generator that does not explicitly model regime shifts will generate "average" data that smears out the distinct characteristics of a crash vs. a boom. RL agents trained on "average" data fail in specific regimes.17
* **Hybrid Approach:** The most robust approach identified in 2024–2025 literature is to combine HMMs with GANs. The HMM models the *transition* between regimes, while separate conditional GANs generate the *microstructure* data within each regime. This provides both high-level structural fidelity and low-level statistical realism.32

### 3.2 Distribution Shift and Artifacts

Synthetic data often contains **spurious correlations**—patterns that appear predictive in the synthetic set but are causal dead-ends in the real world.12 For example, a generator might accidentally couple high volume with positive returns perfectly, whereas in the real world, this correlation is noisy.

* **Detection:** "Observational Overfitting" papers suggest testing the agent on modified observation spaces to detect if it relies on specific artifacts.2 This involves perturbing features in the validation set to see if performance collapses, indicating reliance on brittle correlations.
* **Remedy:** **Regime-aware generation**. Instead of training one GAN on all history, train separate generative models for different regimes (detected via HMM) and stitch them together probabilistically to create training episodes. This breaks the global artifacts and forces the agent to adapt to local dynamics.32

### 3.3 Comparative Analysis of Generators

The following table summarizes the trade-offs between different synthetic data generation techniques for financial RL:

| **Generator** | **Fidelity to Stylized Facts** | **Training Stability** | **Diversity** | **Suitability for RL** | **Source** |
| --- | --- | --- | --- | --- | --- |
| **TimeGAN** | Moderate | Low (Mode Collapse) | Low | Low (Risk of overfitting) | 8 |
| **SigCWGAN** | High (Captures Roughness) | High (Wasserstein Loss) | High | **High** | 8 |
| **Regime-Switching HMM** | High (Explicit Regimes) | Very High (Statistical) | Medium | **High (Structured)** | 29 |

**Conclusion for Q2:** The user should abandon the current "black box" generator. The recommended path is a **hybrid approach**: Use a Hidden Markov Model (HMM) to classify historical data into regimes, then train a **SigCWGAN** or **Conditional GAN** on each regime segment. This ensures the synthetic data retains the harsh, non-linear characteristics of specific market phases, providing the "hard" training examples necessary for robust position sizing.

## 4. Research Question 3: Regularization Techniques That Actually Work

The failure of Dropout (0.2) in the user's experiment aligns with findings that standard supervised regularization (L2, Dropout) is often ineffective in RL because it does not address the **temporal consistency** of the policy or the **diversity** of the ensemble.35 In RL, the data distribution is non-stationary and dependent on the agent's actions, making standard regularization techniques insufficient for preventing overfitting to the environment's dynamics.

### 4.1 Why Dropout Failed

Dropout randomly zeroes out neurons, forcing the network to learn redundant feature representations. However, in financial RL, the overfitting is often to the **Markov Transition Function** of the environment. The agent learns that "State A always leads to State B" in the simulator. Dropout does not prevent the agent from learning this transition rule; it only makes the representation of "State A" noisier. It does not introduce **uncertainty about the dynamics** themselves.36 Furthermore, in LSTMs, dropout is typically applied only to the input and output connections, not the recurrent connections, which preserves the network's ability to memorize long-term sequences inherent to the synthetic generator.37

### 4.2 Maximize Ensemble Diversity (MED-RL)

The fact that all 5 models converged to 0.003 indicates a lack of diversity. **MED-RL (Maximize Ensemble Diversity in Reinforcement Learning)** is a specific regularization technique designed to prevent this collapse.6

* **Mechanism:** MED-RL adds a regularization term to the loss function that penalizes similarity between the ensemble members. It encourages the networks to have different parameter configurations or representation spaces while solving the same task. The loss function $L$ is modified to $L\_{total} = L\_{RL} - \lambda \cdot H(\text{Ensemble})$, where $H$ represents the entropy or diversity metric (such as Jensen-Shannon Divergence) of the ensemble's predictions.
* **Result:** Papers report that MED-RL can improve performance by over 300% in environments with uncertain dynamics by ensuring that the ensemble covers a broader range of possible strategies.6 Instead of five agents learning the same "best" path through the synthetic data, they learn five "good" paths. When transferred to the real world, the intersection of these paths is more likely to be robust than any single path.
* **Implementation:** Implementing MED-RL is computationally efficient and fits within the single A100 constraint. It requires computing the divergence between policy outputs in the batch update step, which is a vectorized operation.

### 4.3 Adversarial Regularization

Adversarial Regularization involves training the agent while an adversary attempts to perturb the input observations (state) to minimize the agent's reward. This effectively creates a "smoothing" of the value function around the training data points.21

* **Application:** In position sizing, the adversary might slightly alter the input volatility or momentum features. If the agent's position size drastically changes due to a microscopic change in input (brittle policy), the adversary punishes this. This forces the agent to learn **smooth, robust sizing curves**.
* **Effectiveness:** This technique is particularly effective against the "fingerprinting" issue. If the agent relies on a specific, tiny artifact of the synthetic generator (e.g., a specific decimal precision in the RSI calculation), the adversary will perturb that feature, breaking the dependency and forcing the agent to rely on more macroscopic, robust market features.21

### 4.4 Meta-Learning (MAML)

Model-Agnostic Meta-Learning (MAML) focuses on learning a set of initial parameters that can quickly adapt to new tasks. In finance, this translates to learning a base policy that can quickly adapt to a new market regime.40 While theoretically powerful for cross-distribution generalization, MAML is computationally expensive (requiring second-order derivatives) and notoriously difficult to tune. Given the "Single A100" constraint and the goal of robustness rather than rapid adaptation, MED-RL and Domain Randomization are more feasible and higher-impact solutions.

**Conclusion for Q3:** The user must implement **MED-RL** to force diversity among the 5 models. Additionally, **Adversarial Regularization** should be applied to the inputs to prevent the agent from reacting to noise artifacts in the synthetic data. This combination directly addresses the ensemble collapse and the brittleness of the learned policies.

## 5. Research Question 4: Position Sizing RL Specifically

Position sizing (determining *how much* to invest) is fundamentally different from direction prediction (determining *what* to buy). RL struggles with sizing because the reward signal (P&L) is highly stochastic, multiplicative, and heavy-tailed.

### 5.1 The Variance Problem in RL Sizing

In a trade, $P\&L = \text{Size} \times \text{Return}$. If the RL agent controls Size directly, the variance of the gradient estimates scales with the size of the position. Large positions lead to massive gradients (exploding), while small positions lead to vanishing gradients.42 This makes convergence unstable.

* **Comparison:** Directional models output a probability (0 to 1), keeping variance bounded. Sizing models output a continuous value (0% to 100% leverage), where small errors in estimation lead to ruin. A slight overestimation of edge when sizing large can lead to a -50% drawdown, which is a massive negative reward that can destroy the policy's learned features.42

### 5.2 Kelly Criterion vs. Learned Sizing

The **Kelly Criterion** ($f^\* = \mu / \sigma^2$) is the theoretically optimal sizing strategy for maximizing geometric growth.43 However, full Kelly is extremely volatile and assumes known probabilities.

* **RL vs. Kelly:** Pure RL attempts to "re-learn" the Kelly criterion from scratch via trial and error. This is inefficient and prone to overfitting. The agent must implicitly learn the concept of variance and the logarithmic nature of compounding, which requires vast amounts of data.45
* **Research Finding:** "RL-Kelly" hybrid models consistently outperform pure RL. In these models, the RL agent predicts the *probability of winning* ($p$) and the *expected payoff* ($b$), and the position size is calculated deterministically using a fractional Kelly formula (e.g., Half-Kelly).42
* **Why Hybrid Works:** It injects domain knowledge (the mathematics of ruin) directly into the agent, relieving the neural network of the burden of learning geometric compounding from noisy rewards. The agent focuses on the simpler task of estimating the *edge*, while the Kelly formula handles the non-linear scaling of that edge into a position size.42

### 5.3 Successful Validation Cases

Papers from 2023–2025 highlight that agents trained to output **action confidence** (which is then mapped to size) generalize better than agents trained to output raw position size directly.46

* **Example:** An agent outputs a scalar $s \in [-1, 1]$. The position size is $|s| \times \text{Max\\_Limit}$. The loss function includes a penalty for variance, effectively teaching the agent to lower $|s|$ when uncertainty is high (implicit Kelly).47
* **Constraint Handling:** Incorporating risk constraints (e.g., Maximum Drawdown) directly into the reward function or as a constraint in the optimization (as in Constrained Policy Optimization, CPO) is crucial for real-world viability. Pure PPO often fails to respect these constraints on test data.48

**Conclusion for Q4:** The user should abandon pure end-to-end RL for sizing. The high variance of the gradients explains the instability and poor transfer. The recommended approach is a **Hybrid RL-Kelly framework**, where the RL agent estimates the *edge* (expected return/risk), and a deterministic layer scales the position using a Fractional Kelly Constraint. This leverages the strength of deep learning (pattern recognition) and the strength of financial theory (risk management).

## 6. Research Question 5: Alternative Approaches

If the RL approach continues to fail despite improvements, robust alternatives must be considered.

### 6.1 Volatility Targeting

Volatility Targeting involves scaling position sizes inversely to market volatility to maintain a constant annualized risk target (e.g., 10% vol).49

* **Pros:** Extremely robust, requires no training, prevents ruin during high-vol regimes. It naturally reduces size during crashes (which are almost always high-volatility events).
* **Cons:** Does not exploit directional signals; purely reactive.
* **Hybrid:** Use RL to adjust the *target* volatility dynamically, rather than the raw position size. This creates a bounded action space for the RL agent, preventing catastrophic sizing errors while allowing for adaptivity.49

### 6.2 Bayesian Position Sizing

Bayesian methods update the probability of a model being correct based on recent performance. If the model's recent predictions are poor, the Bayesian prior shrinks the position size toward zero.51

* **Relevance:** This acts as an automatic "kill switch" or sizing dampener during regime shifts that the model doesn't recognize. If the RL agent enters a new market regime where its predictions degrade, the Bayesian wrapper automatically scales down the positions, preserving capital until the agent adapts or the regime reverts.

### 6.3 Deterministic Base + RL Residual

Train a simple rule-based model (e.g., trend following with Vol Targeting) as the "Base." Train an RL agent to output a "Residual" adjustment ($-\delta, +\delta$) to this base policy.33

* **Benefit:** The agent only learns to *improve* an existing decent strategy, rather than learning from scratch. This stabilizes training and ensures the baseline performance never drops below the rule-based system (if the residual is bounded). This "residual learning" approach is widely successful in robotics and is highly transferable to finance.33

## 7. Deep Analysis of Specific Failure Patterns

### Failure 1: Identical Convergence (The "Easy Mode" Trap)

Observation: All 5 seeds converged to Sharpe 0.003.

Diagnosis: This confirms that the training environment was too easy and too deterministic. The synthetic generator likely had a dominant feature (e.g., simple mean reversion) that all agents found immediately. In a complex, rugged loss landscape (like real markets), diverse seeds settle in diverse local optima. Identical convergence implies a smooth bowl-shaped loss surface created by the synthetic generator's simplicity.

Fix: DORAEMON. By maximizing the entropy of the environment dynamics, we roughen the loss landscape, forcing the ensemble to spread out and find diverse, robust strategies.16

### Failure 2: Validation-to-Test Gap (91% Drop)

Observation: 0.033 $\to$ 0.003.

Diagnosis: This is Target Shift. The joint distribution $P(X, Y)$ of the synthetic data differs from the real data. Specifically, the conditional distribution $P(Y|X)$ (market reaction to news/price) is different. The high validation score proves the model learned $P\_{synth}(Y|X)$ perfectly. The drop proves $P\_{synth} \neq P\_{real}$.

Fix: Adversarial Domain Adaptation. Use a discriminator to penalize the agent if its feature representations can distinguish between synthetic and real data. This forces the agent to learn "domain-invariant" features.20

### Failure 3: Generator Fingerprinting

Observation: Models learned artifacts.

Diagnosis: Deep LSTMs are powerful enough to memorize the pseudo-random number generator (PRNG) patterns or the specific frequency signatures of a TimeGAN.

Fix: Regime-Switching + Noise Injection. Never feed raw synthetic data. Augment it with noise, random time-warping, and regime-switching jumps to break distinct "fingerprints".29

## 8. Ranked List of Solutions

Based on the synthesis of 2020–2025 literature, here is the ranked list of interventions by feasibility and impact.

| **Rank** | **Solution** | **Feasibility** | **Exp. Impact** | **Description** |
| --- | --- | --- | --- | --- |
| **1** | **Hybrid RL-Kelly Sizing** | High | High | Replace direct sizing output with RL-predicted *edge* fed into a Fractional Kelly formula. |
| **2** | **Regime-Aware Data Generation** | Medium | Very High | Use HMM to segment history, train separate SigCWGANs per regime, and sample trajectories that switch regimes. |
| **3** | **MED-RL Ensemble** | High | Medium | Add diversity regularization loss to the ensemble training loop to prevent collapse. |
| **4** | **DORAEMON Domain Randomization** | Medium | High | Dynamically vary synthetic parameters (drift, vol) based on agent performance to maximize entropy. |
| **5** | **Adversarial Training** | Low (Costly) | High | Train against an adversary perturbing the observation space. |

## 9. Implementation Roadmap

### Phase 1: Data Infrastructure Overhaul (Weeks 1-2)

**Goal:** Replace the single synthetic generator with a Regime-Aware Generator.

1. **Regime Identification:** Train a Gaussian HMM on historical data to identify 3 regimes: *Bull (Low Vol)*, *Bear (High Vol)*, *Sideways (Choppy)*.53
2. **Generative Modeling:** Train a **SigCWGAN** (Signature Conditional Wasserstein GAN) for *each* of the 3 regimes independently. SigCWGAN is chosen for its superior ability to capture path roughness compared to TimeGAN.8
3. **Scenario Construction:** Create a "Master Generator" that stitches these regime-specific segments together using the transition matrix from the HMM. This creates synthetic data with realistic regime shifts.

### Phase 2: Agent Architecture & Training (Weeks 3-4)

**Goal:** Implement the Hybrid RL-Kelly Agent with MED-RL.

1. **Hybrid Head:** Modify the PPO Actor network. Instead of outputting position\_size, output conviction ($\mu$) and uncertainty ($\sigma$).
2. **Action Logic:** Calculate size $S\_t = \lambda \cdot \frac{\mu}{\sigma^2}$ (Kelly) clipped to $$.
3. **Ensemble Training:** Initialize 5 agents. Add the MED-RL loss term: $L\_{MED} = -\beta \sum \text{Var}(Q(s, a\_i))$ (penalizing low variance in Q-values across the ensemble).6

### Phase 3: Domain Randomization Loop (Weeks 5-6)

**Goal:** Implement DORAEMON logic.

1. **Parameter Definition:** Define a config space for the synthetic generator (e.g., mean reversion strength $\kappa$, volatility multiplier $\nu$).
2. **Entropy Maximization:** At each epoch, sample generator parameters from a distribution. Update this distribution to widen the variance (increase entropy) *only if* the agents' average reward remains above a threshold.16

### Phase 4: Validation (Week 7)

**Goal:** Robust Testing.

1. **OOD Testing:** Test on real historical data *held out* from the HMM training.
2. **Stress Testing:** Test on synthetic scenarios with "Impossible" parameters (e.g., 5x historical volatility) to ensure the Kelly constraint prevents ruin.

## 10. Red Flags & Negative Results

* **Avoid Pure End-to-End Sizing:** Literature consistently shows that learning to size positions from scratch without a utility prior (like Log-Utility/Kelly) fails due to the sparsity of "ruin" events in training.42
* **Avoid "Average" Generators:** Training a single GAN on 20 years of data results in a "grey goo" of average dynamics that resembles neither a bull run nor a crash. Regime separation is non-negotiable.56
* **Don't Trust Validation Scores:** In synthetic-to-real transfer, a high validation score on synthetic data is often a *warning sign* of overfitting to simulator artifacts. Trust only the OOD (Out-of-Distribution) performance on real data or adversarial synthetic scenarios.2

## 11. Conclusion

The 91% performance collapse observed is a classic symptom of a "high-fidelity illusion." The models mastered the synthetic simulator's artifacts, not the financial market's dynamics. By pivoting to **Regime-Aware Generation** (to fix the data), **DORAEMON** (to fix the curriculum), and **Hybrid Kelly Sizing** (to fix the decision logic), the system can be transformed from a pattern-matcher of noise into a robust financial decision engine. The target Sharpe ratio of > +0.03 is achievable, but only by explicitly modeling and randomizing the uncertainty that the current deterministic ensemble is ignoring.

### Key References for Implementation

* **DORAEMON:** 15
* **MED-RL:** 6
* **SigCWGAN:** 8
* **Hybrid Kelly-RL:** 42

# Detailed Research Synthesis & Implementation Guide

## 1. Introduction: The Sim-to-Real Paradox in Quantitative Finance

Reinforcement Learning (RL) has achieved superhuman performance in closed domains like Go and Dota 2, where the simulation is a perfect replica of reality. In quantitative finance, however, the simulation is merely an approximation—often a poor one—of the complex, adaptive, and non-stationary system that is the market. This discrepancy creates the **Sim-to-Real Gap**, a chasm where models that print money in backtests go bankrupt in live trading.

The user's case study—a 5-model LSTM-PPO ensemble collapsing from a +0.035 Sharpe in training to +0.003 in testing—illustrates the danger of **Observational Overfitting**. This occurs when the agent learns features that are correlated with the reward in the training distribution (synthetic artifacts) but are uncorrelated or spuriously correlated in the target distribution (real markets).2

The fact that five independent seeds converged to the same failure point is particularly damning. It suggests that the synthetic environment lacked the requisite **stochastic complexity** to force the agents to learn diverse strategies. Instead, it presented a smooth, easily exploitable error surface based on the generator's biases. To solve this, we must fundamentally alter how we generate data, how we train the agents, and how we structure the decision-making output.

## 2. Literature Review: Diagnosing the Failure

### 2.1 The "Identity Crisis" of Synthetic Data

Synthetic data in finance is often generated using methods that prioritize statistical similarity (matching moments) over dynamic fidelity (matching causal reactions).

* **TimeGAN Limitations:** While TimeGAN 25 is a standard benchmark, it struggles with **long-tail events**. It minimizes a loss function based on average deviations, which incentivizes the generator to produce "safe," mean-reverting trajectories. An agent trained on this learns that "extreme moves always revert," a fatal assumption in real crypto markets where momentum can cascade.8
* **The Artifact Problem:** Deep generative models leave "fingerprints." A 2023 study showed that RL agents could distinguish between TimeGAN-generated data and real data with 99% accuracy, exploiting microscopic periodicities in the GAN's output to game the reward function.2 The user's agents likely did exactly this.

### 2.2 Ensemble Theory and Collapse

Ensemble methods rely on the principle of **uncorrelated errors**. If Model A fails in Scenario X, Model B might succeed. However, if all models are trained on the same narrow distribution (the synthetic generator) with the same regularization (Dropout), their errors become highly correlated.

* **Dropout is Insufficient:** Dropout provides *intra-model* diversity (robustness to missing features) but not *inter-model* diversity (robustness to different dynamics). It does not force the models to learn *different* policies.35
* **Diversity Collapse:** Without an explicit repulsive force in the loss function (like MED-RL's diversity regularizer), deep neural networks tend to converge to the simplest heuristic that solves the training set. In synthetic data, this simplest heuristic is often "exploit the generator's mean-reversion parameter".6

### 2.3 Position Sizing: The "Ruin" Problem

Position sizing is a concave problem: betting too little yields low return; betting too much yields ruin (geometric decay).

* **Gradient Instability:** In PPO, the policy update is driven by the Advantage function $A\_t$. If the position size is a continuous variable output by the network, a lucky large bet results in a massive $A\_t$, causing a huge gradient step that can destabilize the policy weights. Conversely, a streak of small bets results in vanishing gradients.
* **The Kelly Gap:** The Kelly Criterion maximizes *logarithmic* utility ($\mathbb{E}$), whereas standard RL objectives maximize *additive* rewards ($\mathbb{E}[\sum r\_t]$). This objective mismatch means a standard PPO agent is not actually trying to maximize long-term growth rate, but rather the sum of short-term profits, leading to excessive risk-taking.42

## 3. Solution 1: Advanced Synthetic Data Generation (Regime-Aware SigCWGAN)

The first step to fixing the Sim-to-Real gap is to improve the simulation. We must move from a monolithic generator to a **Regime-Aware** architecture.

### 3.1 Why Regimes Matter

Financial markets exhibit distinct regimes:

1. **Low Volatility / Trending:** Strategies like "Buy and Hold" or "Trend Following" excel.
2. **High Volatility / Mean Reverting:** Strategies like "Market Making" or "Mean Reversion" excel.
3. **Crash / Crisis:** Cash preservation is optimal.

A single GAN trained on all history averages these dynamics into a "muddy" distribution that fits none of them well. An agent trained on this "average" world will be too timid in bull markets and too aggressive in bear markets.56

### 3.2 The Architecture: HMM + SigCWGAN

We propose a two-stage generation pipeline 32:

Stage 1: Regime Segmentation (HMM)

Use a Gaussian Hidden Markov Model (HMM) to unsupervisedly classify real historical data into $K=3$ or $K=4$ latent states.

* *Input:* Log-returns, Volatility (GARCH), Volume.
* *Output:* A sequence of state labels (e.g., $S\_1, S\_1, S\_2, S\_3...$) and a Transition Matrix $T$ describing the probability of moving from regime $i$ to $j$.

Stage 2: Conditional Generation (SigCWGAN)

Train a separate conditional generator for each regime. We recommend SigCWGAN 8 over TimeGAN.

* *Why SigCWGAN?* It uses the **Signature Transform** (from rough path theory) as the discriminator's metric. Signatures efficiently capture the order and area of iterated integrals of the path, which corresponds to preserving the "roughness" and "burstiness" of financial time series. This prevents the "smoothness" artifact common in standard GANs.
* *Implementation:* The generator takes a noise vector $z$ and a regime label $c$ (condition). The discriminator checks if the generated path matches the signature moments of real paths from regime $c$.

Stage 3: Scenario Assembly

To generate a training episode:

1. Sample a regime sequence using the HMM's Transition Matrix (e.g., 50 steps of Bull, then a switch to High Vol).
2. Generate price paths for each segment using the corresponding Regime-Conditional GAN.
3. Stitch them together to create a cohesive market scenario.

**Impact:** This forces the RL agent to learn **regime identification**. It encounters abrupt shifts in volatility and correlation, mimicking real-world shock events. This directly addresses the "Validation-to-Test Gap" by ensuring the training data contains the structural breaks found in reality.

## 4. Solution 2: Regularization via MED-RL & Adversarial Training

To fix the "Identical Convergence" of the ensemble, we must explicitly engineer diversity.

### 4.1 Maximize Ensemble Diversity (MED-RL)

MED-RL modifies the PPO loss function. Standard PPO maximizes the clipped surrogate objective $L^{CLIP}$. In an ensemble of $N$ agents (policies $\pi\_1,..., \pi\_N$), we want to maximize the **Jensen-Shannon Divergence (JSD)** between their policy distributions.6

$$JSD(\pi\_1,..., \pi\_N) = H\left(\frac{1}{N}\sum \pi\_i\right) - \frac{1}{N}\sum H(\pi\_i)$$

* $H(\cdot)$ is the entropy.
* The first term encourages the *average* policy to be broad (covering the space).
* The second term encourages *individual* policies to be decisive (low entropy).

New Loss Function:

$$L\_{Total}(\theta\_i) = L^{CLIP}(\theta\_i) + \alpha \cdot JSD(\pi\_i, \pi\_{\neq i})$$

* **Mechanism:** If Agent 1 learns to "Buy on RSI < 30," the JSD term effectively penalizes Agent 2 for learning the exact same rule. Agent 2 is pushed to find an alternative signal (e.g., "Buy on Bollinger Band breakout") to maximize its distinctiveness while still solving the task.
* **Result:** The ensemble becomes a "Team of Specialists" rather than 5 clones. During testing, if the RSI signal fails (false positive), the Bollinger agent might still vote "Hold," saving the ensemble from a bad trade.

### 4.2 Adversarial Observation Perturbation

To prevent overfitting to precise pixel-level (or float-level) values of the synthetic data, we introduce an **Adversary** during training.21

* Attack: Before the state $s\_t$ is fed to the agent, the adversary adds a perturbation $\delta$ bounded by $\epsilon$ ($||\delta||\_\infty < \epsilon$) to minimize the agent's expected value function $V(s)$.  
    
  $$s\_{adv} = s\_t - \epsilon \cdot \text{sign}(\nabla\_s V(s\_t))$$  
    
  (This is the Fast Gradient Sign Method, FGSM, applied to RL).
* **Defense:** The agent trains on $s\_{adv}$.
* **Intuition:** If a 1% change in input volatility causes the agent to flip from "Max Long" to "Max Short," the adversary will exploit this brittleness. The agent is forced to learn smoother, more conservative decision boundaries.

## 5. Solution 3: Hybrid RL-Kelly Position Sizing

The user's architecture (LSTM-PPO outputting size directly) is prone to high variance. We propose decoupling **Prediction** from **Sizing**.

### 5.1 The Decomposition

Instead of one black box, divide the agent into two components:

1. **Alpha Engine (RL):** Predicts the **Edge** (Direction and Probability).
   * Output: $\hat{p}\_t$ (Probability of up-move) and $\hat{r}\_t$ (Expected return magnitude).
2. **Risk Engine (Kelly):** Calculates **Size**.
   * Formula: $Size\_t = \text{Fraction} \times \frac{\hat{p}\_t b - (1-\hat{p}\_t)}{b}$
   * Where $b$ is the odds ratio (Win Amount / Loss Amount).

### 5.2 The "Learned Fraction"

Pure Kelly is too volatile (full ruin risk). The "Fraction" parameter is crucial.

* **Constraint:** We can train a secondary RL head (or use the uncertainty output of the ensemble) to output the **Kelly Fraction** $k\_t \in $.
* **Uncertainty Scaling:** If the ensemble members disagree (high variance in value estimates), $k\_t$ should drop to 0. If they agree, $k\_t$ approaches 1 (or a safe max like 0.5).
* **Benefit:** This provides a mathematical safety rail. Even if the Alpha Engine is hallucinating a massive return, the Risk Engine (via ensemble variance) can clamp the bet size to zero, preventing the "91% collapse".43

## 6. Implementation Roadmap

This roadmap assumes access to a single A100 GPU and standard Python libraries (PyTorch, Stable-Baselines3, FinRL).

### Phase 1: Synthetic Data Pipeline (Days 1-7)

* **Step 1:** Ingest historical crypto data (OHLCV). Compute features (Log-returns, Volatility, RSI, MACD).
* **Step 2 (Regime Detection):** Use hmmlearn to fit a 3-component Gaussian HMM. Visualize the regimes to ensure they make semantic sense (e.g., Regime 0 = Low Vol, Regime 1 = Crisis).
* **Step 3 (Generator Training):** Implement **SigCWGAN** (code available in 58 or similar repositories). Train one model per regime.
  + *Compute:* Training 3 GANs on financial time series is relatively cheap. ~4-6 hours on A100.
* **Step 4 (Validation):** Compute "Signature Moments" of real vs. synthetic data. Ensure the synthetic data reproduces the "roughness" of real data (autocorrelation of absolute returns).

### Phase 2: DORAEMON Environment Wrapper (Days 8-10)

* **Step 1:** Create a Gym environment that loads the synthetic data.
* **Step 2 (DORAEMON Logic):** Implement a reset() function that randomizes simulation parameters.
  + *Parameters to Randomize:* Transaction costs (0.05% to 0.2%), Slippage, Latency, and initial account balance.
  + *Entropy Loop:* Track the agent's success rate. If > 50%, increase the variance of the randomization distributions (make the environment harder/more diverse). 18

### Phase 3: Ensemble Training with MED-RL (Days 11-17)

* **Step 1:** Instantiate 5 PPO agents (Stable-Baselines3).
* **Step 2 (Custom Loss):** Subclass the PPO policy to include the MED-RL diversity term.
  + *Calculation:* In the batch update, gather the action distributions of all 5 agents. Compute the Jensen-Shannon Divergence. Add $\alpha \cdot JSD$ to the loss.
* **Step 3 (Training):** Train the ensemble on the DORAEMON environment.
  + *Compute:* Parallel training of 5 LSTMs on one A100 is feasible. 500k steps x 5 models $\approx$ 12-18 hours.

### Phase 4: Hybrid Kelly Integration (Days 18-20)

* **Step 1:** Post-processing wrapper. Take the ensemble's averaged logits.
* **Step 2:** Convert logits to probabilities (Softmax).
* **Step 3:** Apply Fractional Kelly formula. Scale fraction based on the standard deviation of the 5 agents' logits (Ensemble Uncertainty).
  + *Logic:* $Size = \text{Kelly}(p) \times (1 - \text{StdDev}(\text{Ensemble}))$.

## 7. Limitations & Caveats

* **Cost of Complexity:** The proposed architecture (HMM + GAN + Ensemble + Kelly) is significantly more complex than a simple LSTM. It requires careful hyperparameter tuning, especially for the HMM (number of states) and the MED-RL alpha (diversity weight).
* **Non-Stationarity:** Even with Regime-Switching GANs, the "Next Crisis" might look different from all historical crises. The model relies on the assumption that future regimes will statistically resemble *some* combination of past regimes.
* **Execution Risk:** The Kelly criterion assumes continuous rebalancing and zero transaction costs. In reality, rebalancing frequently incurs costs that can eat the Kelly edge. The simulation must include realistic fee modeling (randomized via DORAEMON) to prevent the agent from learning a "churning" strategy.

## 8. Conclusion

The user's failure was not due to a lack of model capacity (LSTM-PPO is sufficient) but due to a **lack of environment complexity**. The models solved the simplified synthetic puzzle perfectly, but that puzzle bore little resemblance to the chaotic reality of crypto markets.

By adopting **Regime-Aware Generation** to capture market non-stationarity, **DORAEMON** to enforce robustness via entropy maximization, and **MED-RL** to ensure ensemble diversity, we can bridge the Sim-to-Real gap. Furthermore, grounding the position sizing logic in the **Kelly Criterion** provides a theoretical safety net that pure RL lacks. This holistic approach targets the root causes of the failure—overfitting, mode collapse, and gradient variance—offering a viable path to a positive Test Sharpe ratio.

#### Works cited

1. Bridging the Sim-to-Real Gap from the Information Bottleneck Perspective - arXiv, accessed on January 10, 2026, <https://arxiv.org/html/2305.18464v2>
2. Observational Overfitting in Reinforcement Learning - Google Research, accessed on January 10, 2026, <https://research.google/pubs/observational-overfitting-in-reinforcement-learning/>
3. Reinforcement Learning for Finance — Insights from Yves J. Hilpisch | by ODSC, accessed on January 10, 2026, <https://odsc.medium.com/reinforcement-learning-for-finance-insights-from-yves-j-hilpisch-f4c45a113a03>
4. Machine Learning for Synthetic Data Generation: A Review - arXiv, accessed on January 10, 2026, <https://arxiv.org/pdf/2302.04062>
5. Seeing is not Believing: Robust Reinforcement Learning against Spurious Correlation - OpenReview, accessed on January 10, 2026, <https://openreview.net/pdf/89b125b6e617164f9544bf7df323a05a5e9a8b45.pdf>
6. Maximizing Ensemble Diversity in Deep Reinforcement Learning - OpenReview, accessed on January 10, 2026, <https://openreview.net/forum?id=hjd-kcpDpf2>
7. Ensemble Reinforcement Learning: A Survey - arXiv, accessed on January 10, 2026, <https://arxiv.org/html/2303.02618v3>
8. MiT-WGAN: Financial Time-Series Generation GAN Based on Multi-Convolution Dynamic Fusion and iTransformer - MDPI, accessed on January 10, 2026, <https://www.mdpi.com/2073-8994/17/10/1740>
9. Systematic comparison of deep generative models applied to multivariate financial time series. - arXiv, accessed on January 10, 2026, <https://arxiv.org/pdf/2412.06417>
10. Enhancing Equity Strategy Backtesting with Synthetic Data: An Agent-Based Model Approach | AWS HPC Blog, accessed on January 10, 2026, <https://aws.amazon.com/blogs/hpc/enhancing-equity-strategy-backtesting-with-synthetic-data-an-agent-based-model-approach/>
11. Quant GANs: deep generation of financial time series | Request PDF - ResearchGate, accessed on January 10, 2026, <https://www.researchgate.net/publication/340462995_Quant_GANs_deep_generation_of_financial_time_series>
12. Robust Reinforcement Learning against Spurious Correlation - arXiv, accessed on January 10, 2026, <https://arxiv.org/pdf/2307.07907>
13. [2307.07907] Seeing is not Believing: Robust Reinforcement Learning against Spurious Correlation - arXiv, accessed on January 10, 2026, <https://arxiv.org/abs/2307.07907>
14. Robot Learning from Randomized Simulations: A Review - arXiv, accessed on January 10, 2026, <https://arxiv.org/pdf/2111.00956>
15. Domain Randomization via Entropy Maximization - OpenReview, accessed on January 10, 2026, <https://openreview.net/forum?id=GXtmuiVrOM>
16. [2311.01885] Domain Randomization via Entropy Maximization - arXiv, accessed on January 10, 2026, <https://arxiv.org/abs/2311.01885>
17. Synthetic Financial Time Series Generation with Regime Clustering - ResearchGate, accessed on January 10, 2026, <https://www.researchgate.net/publication/376489448_Synthetic_Financial_Time_Series_Generation_with_Regime_Clustering>
18. DOMAIN RANDOMIZATION VIA ENTROPY MAXIMIZATION - ICLR Proceedings, accessed on January 10, 2026, <https://proceedings.iclr.cc/paper_files/paper/2024/file/56adf9cb91aedfa41ce24398782a012f-Paper-Conference.pdf>
19. Domain Randomization via Entropy Maximization - arXiv, accessed on January 10, 2026, <https://arxiv.org/html/2311.01885v2>
20. Explainable Domain Adaptation Learning Framework for Credit Scoring in Internet Finance Through Adversarial Transfer Learning and Ensemble Fusion Model - MDPI, accessed on January 10, 2026, <https://www.mdpi.com/2227-7390/13/7/1045>
21. Model Robustness to Adversarial Data in Financial Time Series - ResearchGate, accessed on January 10, 2026, <https://www.researchgate.net/publication/392131746_Model_Robustness_to_Adversarial_Data_in_Financial_Time_Series>
22. Adversarial Deep Reinforcement Learning in Portfolio Management - arXiv, accessed on January 10, 2026, <https://arxiv.org/pdf/1808.09940>
23. Robust Reinforcement Learning via Adversarial training with Langevin Dynamics - NeurIPS, accessed on January 10, 2026, <https://papers.neurips.cc/paper_files/paper/2020/file/5cb0e249689cd6d8369c4885435a56c2-Paper.pdf>
24. Domain Randomization for Sim2Real Transfer | Lil'Log, accessed on January 10, 2026, <https://lilianweng.github.io/posts/2019-05-05-domain-randomization/>
25. Time-series Generative Adversarial Networks - NIPS papers, accessed on January 10, 2026, <http://papers.neurips.cc/paper/8789-time-series-generative-adversarial-networks.pdf>
26. Fiaingen: A financial time series generative method matching real-world data quality - arXiv, accessed on January 10, 2026, <https://arxiv.org/html/2510.01169v1>
27. The Signature-Wasserstein GAN for Time Series Generation and Beyond - UCL Discovery, accessed on January 10, 2026, <https://discovery.ucl.ac.uk/id/eprint/10181123/1/PhD_Thesis_Final_Baoren_Xiao_18031028.pdf>
28. Generative model for financial time series trained with MMD using a signature kernel - arXiv, accessed on January 10, 2026, <https://arxiv.org/html/2407.19848v3>
29. Deep reinforcement learning for optimal trading with partial information - arXiv, accessed on January 10, 2026, <https://arxiv.org/html/2511.00190v1>
30. Using Reinforcement learning for Regime Switching - YouTube, accessed on January 10, 2026, <https://www.youtube.com/watch?v=LJVZqaRpw2Q>
31. [2512.21395] A Reinforcement Learning Approach to Synthetic Data Generation - arXiv, accessed on January 10, 2026, <https://www.arxiv.org/abs/2512.21395>
32. Regime-Specific Quant Generative Adversarial Network: A Conditional Generative Adversarial Network for Regime-Specific Deepfakes of Financial Time Series - MDPI, accessed on January 10, 2026, <https://www.mdpi.com/2076-3417/13/19/10639>
33. Building an Adaptive Trading System with Regime Switching, GA's & RL : r/quant - Reddit, accessed on January 10, 2026, <https://www.reddit.com/r/quant/comments/1jhhk3c/building_an_adaptive_trading_system_with_regime/>
34. Pairs Trading with Markov Regime-Switching Model - Hudson & Thames, accessed on January 10, 2026, <https://hudsonthames.org/pairs-trading-with-markov-regime-switching-model/>
35. Overfitting In Reinforcement Learning - Meegle, accessed on January 10, 2026, <https://www.meegle.com/en_us/topics/overfitting/overfitting-in-reinforcement-learning>
36. Ensemble-based Deep Reinforcement Learning for Vehicle Routing Problems under Distribution Shift - NeurIPS, accessed on January 10, 2026, <https://proceedings.neurips.cc/paper_files/paper/2023/file/a68120d2eb2f53f7d9e71547591aef11-Paper-Conference.pdf>
37. Financial AI: Synthetic Data Augmentation for Deep Reinforcement Learning and Graph Neural Networks - King's College London Research Portal, accessed on January 10, 2026, <https://kclpure.kcl.ac.uk/portal/en/studentTheses/financial-ai-synthetic-data-augmentation-for-deep-reinforcement-l/>
38. ICLR Poster Maximizing Ensemble Diversity in Deep Reinforcement Learning, accessed on January 10, 2026, <https://iclr.cc/virtual/2022/poster/6498>
39. davide97l/rl-policies-attacks-defenses: Adversarial attacks on Deep Reinforcement Learning (RL) - GitHub, accessed on January 10, 2026, <https://github.com/davide97l/rl-policies-attacks-defenses>
40. The Evolution of Reinforcement Learning in Quantitative Finance: A Survey - arXiv, accessed on January 10, 2026, <https://arxiv.org/html/2408.10932v3>
41. Reinforcement Learning in Financial Decision Making: A Systematic Review of Performance, Challenges, and Implementation Strategies - arXiv, accessed on January 10, 2026, <https://arxiv.org/html/2512.10913v1>
42. Reinforcement Learning with Kelly Criterion for Optimal Portfolio Management of AAPL Stock - ResearchGate, accessed on January 10, 2026, <https://www.researchgate.net/publication/391166658_Reinforcement_Learning_with_Kelly_Criterion_for_Optimal_Portfolio_Management_of_AAPL_Stock>
43. Kelly Criterion vs Fixed Fractional: Which Risk Model Maximizes Long‑Term Growth?, accessed on January 10, 2026, <https://medium.com/@tmapendembe_28659/kelly-criterion-vs-fixed-fractional-which-risk-model-maximizes-long-term-growth-972ecb606e6c>
44. Beware of Excessive Leverage – Introduction to Kelly and Optimal F - QuantPedia, accessed on January 10, 2026, <https://quantpedia.com/beware-of-excessive-leverage-introduction-to-kelly-and-optimal-f/>
45. The Reinforcement Learning Kelly Strategy - UWSpace - University of Waterloo, accessed on January 10, 2026, <https://uwspace.uwaterloo.ca/bitstreams/51b3b16e-e397-47bf-9b38-7bf2d55b629e/download>
46. Deep Reinforcement Learning for Optimal Asset Allocation Using DDPG with TiDE - arXiv, accessed on January 10, 2026, <https://arxiv.org/html/2508.20103v1>
47. Sizing the Risk: Kelly, VIX, and Hybrid Approaches in Put-Writing on Index Options, accessed on January 10, 2026, <https://ideas.repec.org/p/arx/papers/2508.16598.html>
48. The Risk-Constrained Kelly Criterion: From the foundations to trading - QuantInsti Blog, accessed on January 10, 2026, <https://blog.quantinsti.com/risk-constrained-kelly-criterion/>
49. Position Sizing in Trend-Following: Comparing Volatility Targeting, Volatility Parity, and Pyramiding - Concretum Group, accessed on January 10, 2026, <https://concretumgroup.com/position-sizing-in-trend-following-comparing-volatility-targeting-volatility-parity-and-pyramiding/>
50. An Introduction to Volatility Targeting - QuantPedia, accessed on January 10, 2026, <https://quantpedia.com/an-introduction-to-volatility-targeting/>
51. Advancing Investment Frontiers: Industry-grade Deep Reinforcement Learning for Portfolio Optimization - arXiv, accessed on January 10, 2026, <https://arxiv.org/html/2403.07916v1>
52. VARIATIONAL RECURRENT ADVERSARIAL DEEP DOMAIN ADAPTATION - OpenReview, accessed on January 10, 2026, <https://openreview.net/pdf?id=rk9eAFcxg>
53. Market Regime using Hidden Markov Model - QuantInsti Blog, accessed on January 10, 2026, <https://blog.quantinsti.com/regime-adaptive-trading-python/>
54. Tutorial — hmmlearn 0.3.3 documentation - Read the Docs, accessed on January 10, 2026, <https://hmmlearn.readthedocs.io/en/stable/tutorial.html>
55. SPQR: Controlling Q-ensemble Independence with Spiked Random Model for Reinforcement Learning - NeurIPS, accessed on January 10, 2026, <https://proceedings.neurips.cc/paper_files/paper/2023/file/cdcaf772b4f8eda0385d0930517de64a-Paper-Conference.pdf>
56. Deep generative modeling for financial time series with application in VaR: a comparative review Corresponding author - arXiv, accessed on January 10, 2026, <https://arxiv.org/html/2401.10370v1>
57. Hey Kelly, Optimize My Portfolio - Medium, accessed on January 10, 2026, <https://medium.com/@jlevi.nyc/hey-kelly-optimize-my-portfolio-64e835fbced9>
58. W-Mrt/Wassersteain\_Gan\_financial\_time\_series: Neural Network and Deep Learning course - GitHub, accessed on January 10, 2026, <https://github.com/W-Mrt/Wassersteain_Gan_financial_time_series>