Delta Hedging Optimization with Reinforcement Learning

Jens Cancio Youssef Chelaifa Szymon Ścibior

April 2026

1 Project Overview

This project aims to build a Reinforcement Learning agent that learns optimal dynamic hedging strategies for derivatives, outperforming classical Black-Scholes delta hedging under realistic market conditions (transaction costs, discrete rebalancing). Beyond reproducing existing work, the goal is to test the agent under conditions that haven't been explored in the literature: cross-model robustness, diverse asset classes, multi-asset portfolios, and uncertainty-aware hedging, with the aim of producing an original contribution.

Why this matters. In a frictionless world, BS delta hedging is perfect. In reality, every rebalance costs money. Recent research (JP Morgan, Hull et al.) shows RL agents can reduce hedging costs by 30–42% by learning when it's worth paying the rebalancing cost and when it's better to tolerate some risk. This is an active research area at the intersection of quantitative finance and machine learning.

2 Background

Classical delta hedging: You sell a call option, compute its BS delta (δ), and buy that many shares of the underlying. As the stock moves, δ changes, so you rebalance periodically. Problem: each rebalance incurs transaction costs. Rebalance too often = high costs. Too rarely = poor hedge.

The RL approach: Frame hedging as a Markov Decision Process. At each time step, the agent observes the market state (stock price, current position, time left) and decides how many shares to hold. It's trained to minimize a risk-adjusted cost function. Several candidate metrics exist (mean-variance, CVaR (tail risk), entropic risk) and part of the project is to compare which metric produces the most robust and cost-efficient hedging behavior.

Key insight: The agent doesn't invent a radically new strategy. It learns to be a smarter, more cost-aware version of delta hedging, smoothing the hedge trajectory and avoiding unnecessary trades. It systematically under-hedges when rebalancing is expensive and gamma is high, and hedges more aggressively when delta is stable.

3 Project Phases

Phase	Tasks	Output
1. Environment & Baseline	Simulate stock paths (GBM, Heston, Merton); implement BS delta hedging baseline; build P&L tracker with proportional costs; set up evaluation metrics (mean-std, CVaR, entropic risk, quadratic variation)	Working simulation environment + BS baseline results
2. RL Agent Development	Build DDPG and SAC agents using PyTorch; use PFHedge library as scaffolding; define state/action/reward (accounting P&L); train on simulated episodes (100k+ paths); tune hyperparameters, validate convergence	Trained agents that converge and produce valid hedging policies
3. Benchmark & Comparison	Compare RL vs BS delta across: different transaction cost levels, volatility regimes (low / high / crisis), option maturities (1m, 3m, 6m); DDPG vs SAC vs TD3 head-to-head	Benchmark tables, P&L distributions, performance charts
4. Novel Experiments	Cross-model robustness: train on GBM, test on Heston or Merton (and vice versa); cross-asset hedging: FX options, commodity futures, index options; multi-asset portfolio hedging with correlated underlyings; stress testing: agent behavior during vol spikes, regime changes, and jump events; uncertainty quantification via deep ensembles	Original results not covered in existing literature
5. Paper & Presentation	Write up methodology and results; position contribution vs. existing work; code cleanup, documentation, GitHub repo; final presentation	Working paper + clean codebase + presentation

4 Technical Details

Market Models

Three simulation environments are used, each adding realism over the last. **GBM** is the standard baseline: constant drift and volatility, smooth continuous paths, no jumps. **Merton jump-diffusion** adds a Poisson jump component on top of GBM, producing the sudden price gaps that occur overnight, over weekends, or on earnings announcements. **Heston stochastic volatility** lets volatility evolve and mean-revert, capturing the volatility smile and the leverage effect. It is the standard benchmark in the deep hedging literature.

Component	Definition
State $s(t)$	Current holding $H(t)$, stock price $S(t)$, time to maturity
Action $a(t)$	Number of shares to hold next period (continuous, 0 to 1 for a call)
Reward $r(t)$	Accounting P&L: change in option value + stock profit − transaction costs
Objective	Minimize risk-adjusted hedging cost (comparing multiple metrics)

MDP Formulation

Risk Measures (To Be Compared)

- **Mean + $\lambda\times$ Std** [3]. Simple and interpretable. Penalizes overall volatility of hedging cost. Limitation: treats upside and downside variance equally.
- **CVaR (Conditional Value at Risk)** [2]. The expected loss in the worst $\alpha\%$ of scenarios (e.g. worst 5%). Focuses specifically on tail risk, meaning the catastrophic hedging failures a trader actually fears.
- **Entropic risk measure.** A convex risk measure from utility theory. Smoothly interpolates between risk-neutral and worst-case depending on a risk aversion parameter. Used in PFHedge as a default.
- **Quadratic variation penalty** [1]. Penalizes the path-wise variability of the P&L, not just the final distribution. Captures the idea that a hedge that fluctuates wildly before converging is worse than one that's smooth throughout.

One of the project's outputs will be a direct comparison of how these metrics affect the agent's learned behavior and out-of-sample performance. Different metrics may produce agents with very different hedging styles (aggressive vs. conservative, frequent vs. sparse rebalancing).

Platform & Implementation

Core framework: We'll use PFHedge (`github.com/pfnet-research/pfhedge`), a PyTorch-based library built specifically for deep hedging. It provides modular components for building hedgers, simulating market environments (GBM, Heston), defining risk measures (entropic, CVaR, mean-variance), and training agents with any PyTorch optimizer.

Custom agents: For the algorithm comparison (DDPG vs SAC vs TD3), we'll implement the agents in PyTorch directly, using PFHedge's environment and instrument modules but swapping in our own policy networks. This lets us benchmark algorithms fairly on identical environments.

Training: All training runs on simulated data (Monte Carlo paths). For the novel experiments (cross-asset, cross-model), we'll extend PFHedge's simulator classes to support FX and commodity dynamics. GPU acceleration is supported natively by PyTorch for large batch training.

Data

Simulated (primary): Monte Carlo stock price paths from GBM, Heston, and Merton models. Options priced with BS. Standard approach used in Hull et al. (2021) and Buehler et al. (2019). No external data required.

Empirical (Phase 4): Historical price data for S&P 500 constituents, major FX pairs, and

commodity ETFs. The exact data sources and option pricing approach for the cross-asset experiments will be determined as the project progresses.

Uncertainty Quantification

A trained agent outputs a single hedge ratio at each step with no measure of reliability. We apply deep ensembles [4], a well-established uncertainty quantification technique in which multiple networks are trained independently and their disagreement at test time serves as a confidence signal. Concretely, we train a small ensemble of M RL agents: at each time step, the standard deviation across their recommended hedge ratios measures how much the agents disagree. When disagreement is low the ensemble mean is used directly; when disagreement is high the blend shifts toward the classical BS delta. The blending weight is optimised to minimise CVaR on held-out paths. A similar approach has recently been explored in the supervised deep hedging setting [6]; our contribution is to bring it into the RL framework.

5 Tools & Stack

Component	**Tool**
Language	Python 3.10+
Deep learning	PyTorch
Deep hedging	PFHedge (pfnet-research/pfhedge)
Simulation	NumPy, SciPy
Visualization	Matplotlib, seaborn
Version control	Git + GitHub

6 What Makes This Original

Existing deep hedging papers almost exclusively train and test under the same model, hedge vanilla European calls on a single equity underlying, and commit to a single risk measure upfront. Our project pushes beyond this in five directions:

- **Risk measure comparison:** Train agents under different objectives (mean-std, CVaR, entropic, quadratic variation) and compare how each metric shapes the agent's hedging style, cost efficiency, and tail-risk behavior. No paper systematically compares all four.
- **Cross-model robustness:** Train on GBM, test on Heston or Merton (and vice versa). How fragile is the agent when deployed in a world that doesn't match its training? This is the key question for any real-world deployment and is underexplored in the literature.
- **Cross-asset generalization:** Test the agent on FX options, commodity futures options, and index options, assets with fundamentally different dynamics (mean-reversion, jumps, seasonality). Can an agent trained on equities transfer to other asset classes?
- **Multi-asset portfolio hedging:** Hedge a portfolio of options on correlated underlyings simultaneously. This is where dimensionality explodes and RL could genuinely outperform Greeks-based approaches that treat each position independently.

- **Uncertainty quantification:** Apply deep ensembles [4] to RL hedging agents, using per-step disagreement to construct a confidence-weighted blended strategy. No existing RL hedging paper reports model confidence alongside hedge recommendations, making this a practical step toward real deployment.

End goal. A working paper documenting novel experimental results on RL-based hedging under conditions not previously tested (cross-model, cross-asset, and multi-asset) with a clean, reproducible codebase. If the results are strong, this is publishable at a venue like ICAIF (ACM International Conference on AI in Finance) or in *Quantitative Finance*.

Key References

[1] Brini, A., Domeniconi, G., and Fathi, A. Data-driven derivative hedging with quadratic variation penalty. In *Proceedings of the 5th ACM International Conference on AI in Finance (ICAIF '24)*, 2024. `https://doi.org/10.1145/3677052.3698664`

[2] Bühler, H., Gonon, L., Teichmann, J., and Wood, B. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019. `arxiv.org/abs/1802.03042`

[3] Hull, J., Cao, J., Chen, J., and Poulos, Z. Deep hedging of derivatives using reinforcement learning. *arXiv:2103.16409*, 2021.

[4] Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, 30, 2017.

[5] PFHedge – PyTorch Deep Hedging framework. `github.com/pfnet-research/pfhedge`

[6] Poddar, M. Uncertainty-aware deep hedging. *arXiv:2603.10137*, 2026.