

Neural Networks and Deep Learning



Lecture 10: Reinforcement learning: trading, hedging and battery dispatch



Prof. dr. Daniel Traian PELE

Bucharest University of Economic Studies (ASE), Department of Statistics and Econometrics
Romanian Academy, Institute for Economic Forecasting (IPE)
IDA Institute Digital Assets ▪ AI4EFin ▪ MSCA Digital Finance

Learning objectives

- Write a business decision as a Markov decision process (MDP): state, action, reward, transition and discount.
- Solve a small MDP exactly with the Bellman equation and dynamic programming (DP), and read the value function.
- Train value-based agents (tabular Q-learning, a deep Q-network (DQN)) and a policy-gradient agent (proximal policy optimisation (PPO)) on real prices, with several seeds and an honest benchmark.
- Hedge an option with a neural policy that minimises the expected shortfall (ES) of the loss, and compare it with the Black-Scholes delta.
- Judge when reinforcement learning (RL) adds value in business and when a simple rule is enough.

Abbreviations

A2C	advantage actor-critic
AI	artificial intelligence
ASE	Bucharest University of Economic Studies
ATM	at the money (strike = current price)
BG	Bulgaria (bidding zone)
CET	Central European Time
CPU	central processing unit
DE-LU	Germany-Luxembourg bidding zone
DM	Diebold-Mariano test
DP	dynamic programming
DQN	deep Q-network
ENTSO-E	European Network of Transmission System Operators for Electricity
ES	expected shortfall
EU	European Union
EUR	euro
GAE	generalised advantage estimation
HU	Hungary (bidding zone)
IPE	Institute for Economic Forecasting, Romanian Academy
ISE	Fraunhofer Institute for Solar Energy Systems
KL	Kullback-Leibler divergence
MDP	Markov decision process
MLP	multilayer perceptron
MW	megawatt
MWh	megawatt-hour
PPO	proximal policy optimisation
ReLU	rectified linear unit
RL	reinforcement learning
RO	Romania (bidding zone)
SSRN	Social Science Research Network
TD	temporal difference
TRPO	trust region policy optimisation
USD	US dollar
VaR	value at risk
VIX	Cboe Volatility Index (30-day implied volatility of the S&P 500)

Real data in this lecture

- Three problems, three real datasets
  - Every number on the slides is computed by analysis_L10.py and stored in L10_numbers.json.

Problem	Data and source	Split (time order)
Battery dispatch	Hourly day-ahead prices, Romania (RO), Hungary (HU), Germany-Luxembourg (DE-LU), Bulgaria (BG); energy-charts.info of the Fraunhofer Institute for Solar Energy Systems (ISE), from data of the European Network of Transmission System Operators for Electricity (ENTSO-E)	train 2019–2023, validation 2024, test 1 Jan 2025 – 24 Oct 2026
Trading	S&P 500 daily closes and the Cboe Volatility Index (VIX); Yahoo Finance	train 2005–2016, validation 2017–2019, test 2020 – 29 Oct 2026
Option hedging	21-day S&P 500 paths, VIX as implied volatility	paths starting 2005–2016 / 2017–2019 / 2020 – 1 Oct 2026

- Data cut at 31 Oct 2026, the last complete month
  - Seeds 20261108 to 20261108+4
  - Training on an Apple M5 Max central processing unit (CPU), trained results cached.

Outline

- Sequential decisions in business
- Markov decision processes and the Bellman equation
- Value-based learning: Q-learning and DQN
- Policy gradients and PPO: trading the S&P 500
- Deep hedging: a policy that minimises tail risk

Reinforcement learning and supervised learning

- **In one sentence:** supervised learning predicts a label; RL chooses actions whose consequences arrive later and depend on what it did before.

- **Supervised**
  - Data $(\mathbf{x}_i, y_i)$, loss $\mathcal{L}(\hat{y}_i, y_i)$ known for each example
  - The prediction does not change the next example
- **Reinforcement**
  - The agent sees a state s_t, acts a_t, receives a reward r_{t+1}
  - The action changes the next state (a full battery cannot charge)
  - No teacher gives the right action: only a delayed, noisy reward

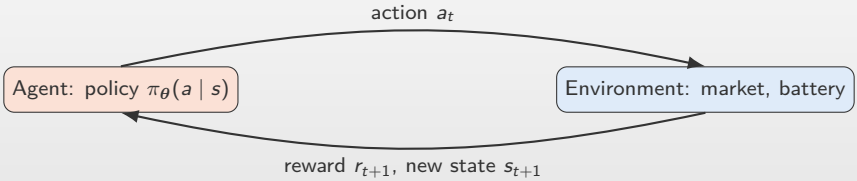
- **Business decisions of this kind**
  - Storage: when to charge and discharge
  - Trading: when to be in or out of the market
  - Hedging: how much of the underlying to hold each day
  - Inventory, pricing, credit limits

- **In practice:** use RL only when today's action changes tomorrow's options.

(Sutton & Barto, 2018, ch. 1)

The agent-environment loop

- **In one sentence:** the agent acts, the environment answers with a reward and a new state; learning means improving the policy from these answers.



- **Episode**: a delivery day, a 126-day trading window, a 21-day option life.
- **Model-free**: the agent learns from sampled transitions $(s_t, a_t, r_{t+1}, s_{t+1})$, never from the transition law.
- **Offline here**: all three agents learn from historical data.

- **In practice:** write down state, action and reward before training anything.

(Sutton & Barto, 2018, ch. 3)

Three business applications of reinforcement learning

Problem	State s_t	Action a_t	Reward r_{t+1}
Battery	hour, stored energy, yesterday's prices	charge / idle / discharge	cash flow of the hour (EUR)
Trading	past returns, volatility, VIX, trend, current position	flat / long	position $\times$ next return $-$ cost
Hedging	index level, time to maturity, VIX, previous hedge	hedge ratio $\delta_t \in [0, 1]$	terminal profit and loss, judged by its ES

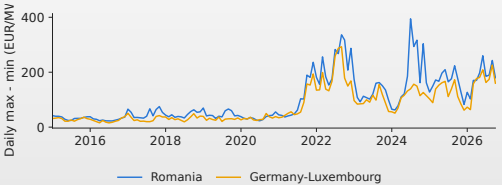
- Same mathematics: maximise expected discounted reward
  - Different information and different risk attitudes.
- Benchmarks are rules a manager already uses: yesterday's plan, buy-and-hold, the 200-day moving average, Black-Scholes delta.

- **In practice:** an RL agent is worth deploying only if it beats the rule the desk already uses, after costs and over several seeds.

(own computation)

Battery revenue from the daily price spread and the growth of the spread

- **In one sentence:** a battery buys in the cheapest hours and sells in the dearest; its income scales with the daily max-min spread.



Source: energy-charts.info (Fraunhofer ISE), from exchange and ENTSO-E data. Data: 2 Jan 2015 to 31 Oct 2026, monthly mean of the daily spread.

- Romania: mean daily spread 47.9 EUR/MWh in 2019, 164.3 in 2025, 182.8 in 2026 so far.
- Peak month Jul 2024: 394.4 EUR/MWh
  - Solar output now pushes midday prices down.

- **In practice:** the business case of a battery is a bet on future spreads, not on the price level.

(own computation) NN_L10_spread_monthly

Outline

- Sequential decisions in business
- Markov decision processes and the Bellman equation
- Value-based learning: Q-learning and DQN
- Policy gradients and PPO: trading the S&P 500
- Deep hedging: a policy that minimises tail risk

The five elements of a Markov decision process

- **In one sentence:** the future depends on the past only through the current state and action.

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$$

$$P(s' \mid s, a) = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$$

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}, \quad \pi^* = \arg\max_{\pi} \mathbb{E}_\pi[G_0]$$

- **where**
  - $\mathcal{S}, \mathcal{A}$ = states and actions, P = transition law, $R(s,a) = \mathbb{E}[r_{t+1} \mid s, a]$
  - $\gamma \in [0,1]$ = discount factor, G_t = return, $\pi(a \mid s)$ = policy.

- **Markov property**: the state must contain everything relevant
  - For the battery: hour and stored energy
  - For trading: the current position, because it determines the cost of the next trade.
- **Discounting**: $\gamma = 1$ for one-day battery episodes (24 steps), $\gamma = 0.99$ for trading.
- **Finite horizon**: the battery starts each day empty and its value at midnight is zero.

- **In practice:** a missing state variable, such as the current position, makes the agent erratic.

(Sutton & Barto, 2018, ch. 3)

Value functions measure how good a state is

- **In one sentence:** V is the expected future reward from a state; Q is the same after a given first action.

$$V^\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s]$$
$$Q^\pi(s,a) = \mathbb{E}_\pi[G_t \mid s_t = s, a_t = a]$$
$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$$

- **where**
  - A^π = advantage of action a over the policy's average action
  - The optimal policy is greedy in Q^*: $\pi^*(s) = \arg\max_a Q^*(s,a)$.

- **Value-based methods** (Q-learning, DQN) estimate Q^* and act greedily.
- **Policy-based methods** (REINFORCE, PPO) parameterise π_θ directly and use V only as a baseline.
- **Battery example**: $V_h(s)$ = the euros still to be earned today from hour h with s MWh stored.

- **In practice:** V prices an asset (a charged battery, an open position)
  - Q compares decisions.

(Sutton & Barto, 2018, ch. 3–4)

The Bellman equation: recursive decomposition of the value function

- **In one sentence:** the best value now = best immediate reward + discounted best value of where you land.

$$V^*(s) = \max_a \left\{ R(s,a) + \gamma \sum_{s'} P(s' \mid s,a)\, V^*(s') \right\}$$

$$Q^*(s,a) = R(s,a) + \gamma\, \mathbb{E}_{s'} \left[\max_{a'} Q^*(s',a') \right]$$

- **where**
  - The right-hand side is a contraction for $\gamma < 1$, so value iteration $V_{k+1} = \mathcal{T} V_k$ converges to the unique fixed point V^*
  - With a finite horizon, one backward pass is exact.

- **Principle of optimality** (Bellman, 1957): an optimal plan stays optimal from any intermediate state.
- **Curse of dimensionality**: the table has $|\mathcal{S}| \times |\mathcal{A}|$ cells
  - Neural networks replace the table when the state is large.
- **Model needed**: DP uses P and R
  - Q-learning and DQN replace the expectation by samples.

- **In practice:** every RL algorithm in this lecture is an approximate way to solve this equation.

(Bellman, 1957; Sutton & Barto, 2018, ch. 4)

The battery as an MDP

- **In one sentence:** state = (hour, stored energy); action = charge, idle or discharge; reward = the cash flow of the hour.

$$s_{h+1} = s_h - a_h, \quad s_h \in \{0, 1, 2\} \text{ MWh}, \quad a_h \in \{-1, 0, +1\}$$

$$r_h(a_h) = \begin{cases} \eta_d \, p_h & a_h = +1 \ (\textit{sell}) \\ -p_h / \eta_c & a_h = -1 \ (\textit{buy}) \\ 0 & a_h = 0 \end{cases}$$

- **where**
  - p_h = day-ahead price of hour h (EUR/MWh), $\eta_c = \eta_d = \sqrt{0.85}$ (85 % round trip)
  - 2 MWh, 1 MWh moved per hour, empty at 0:00.

- **Constraint**: no charging when full, no discharging when empty (action masking).
- **Data**: 4 292 Romanian delivery days with 24 complete hours, 2 Jan 2015 – 31 Oct 2026
  - Since Oct 2025 the hour is the mean of four 15-minute prices.
- **Simplifications**: price taker, no degradation cost, no grid fees, no ancillary services.

- **In practice:** the stored energy is what makes the decision sequential.

(own computation)

Experiment yourself: notebook §2

Dynamic programming solves the battery day exactly

- **In one sentence:** go backwards from midnight: at each hour, keep the best action for each level of stored energy.

$$V_{24}(s) = 0$$

$$V_h(s) = \max_{a:\, 0 \le s-a \le 2} \left\{ r_h(a) + V_{h+1}(s-a) \right\}, \quad h = 23, \ldots, 0$$

- **where**
  - $V_h(s)$ = maximal cash flow from hour h to midnight with s MWh stored
  - The revenue of the day is $V_0(0)$
  - The argmax at each (h, s) is the optimal action.

- **Cost**: $24 \times 3 \times 3$ evaluations per day
  - All 4 292 days in milliseconds, vectorised over days.
- **Perfect foresight**: DP uses the day's 24 prices before they are known, so its revenue is an **upper bound** for any real strategy.
- **Naive benchmark**: compute yesterday's optimal plan and apply it today.

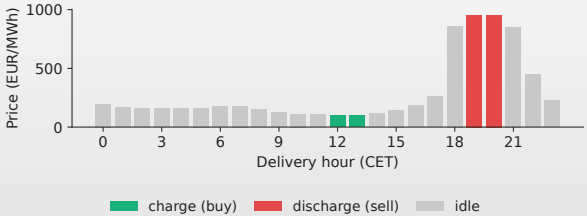
- **In practice:** DP gives the ceiling against which every learning agent is scored (capture ratio = revenue / perfect-foresight revenue).

(Bellman, 1957)

Experiment yourself: notebook §2

Optimal battery schedule for one day

- **In one sentence:** on the day with the largest spread of 2026, DP charges in the cheapest hours and discharges in the dearest.



Source: energy-charts.info (Fraunhofer ISE), from exchange and ENTSO-E data. Data: Romania, 30 Jun 2026, hourly day-ahead prices.

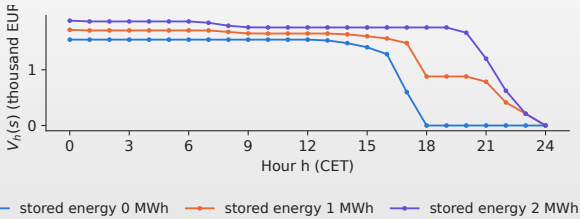
- 30 Jun 2026: prices from 100.65 (hour 13) to 954.59 EUR/MWh (hour 20)
  - 2 charging and 2 discharging hours.
- Revenue of the day for a 2 MWh battery: 1 541 EUR.

(own computation)

NN_L10_dp_day

The value function prices stored energy hour by hour

- **In one sentence:** $V_h(2) - V_h(0)$ is what two stored MWh are worth at hour h: high before the evening peak, zero after it.



Source: energy-charts.info (Fraunhofer ISE), from exchange and ENTSO-E data. Data: Romania, 30 Jun 2026; own computation (backward induction).

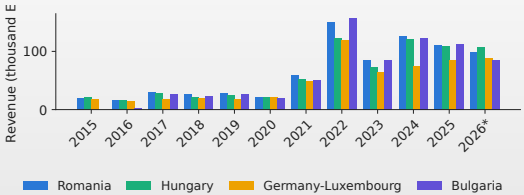
- At 12:00 on 30 Jun 2026: $V_{12}(0) = 1\,541$ EUR, $V_{12}(2) = 1\,760$ EUR, so one stored MWh is worth 109.38 EUR, the shadow price of energy in the battery.
- A good learned Q must reproduce this shape without seeing the day's prices.

(own computation)

NN_L10_dp_value

Revenue of a 2 MWh battery under perfect foresight

- **In one sentence:** the DP revenue is an upper bound; it follows the spreads, not the price level.



Source: energy-charts.info (Fraunhofer ISE), from exchange and ENTSO-E data. Data: 2015-2026 (*2026: January-October); days with 24 complete hours.

- Romania: 28.4 thousand EUR in 2019, 150.2 in 2022, 111.2 in 2025, 98.5 in 2026 over 302 days.
- 2025: Hungary 109.9, Bulgaria 112.9, Germany-Luxembourg 84.3 thousand EUR.

- **In practice:** investors size storage on these numbers
  - A real operator captures only part of them, as the next section measures.

(own computation) NN_L10_dp_revenue_year

Outline

- Sequential decisions in business
- Markov decision processes and the Bellman equation
- Value-based learning: Q-learning and DQN
- Policy gradients and PPO: trading the S&P 500
- Deep hedging: a policy that minimises tail risk

Q-learning: learn Q^* from samples, without a model

- **In one sentence:** move $Q(s,a)$ towards the reward plus the best value of the next state; the gap is the temporal-difference (TD) error.

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \underbrace{\left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)\right]}_{\text{TD error } \delta_t}$$

- **where**
  - α = step size (0.05 here)
  - The target uses the greedy action in s_{t+1} whatever the agent does next, so Q-learning is **off-policy**.

- **Convergence** (Watkins & Dayan, 1992): every pair (s,a) visited infinitely often and $\sum \alpha_t = \infty$, $\sum \alpha_t^2 < \infty$ (Appendix A).
- **Exploration**: ε-greedy
  - Random action with probability ε, decayed from 1 to 0.05.
- **Exploitation risk**: without exploration, a bad early estimate is never corrected.

- **In practice:** the TD error is a learning signal available after every hour, not only at the end of the day.

(Watkins & Dayan, 1992)

Experiment yourself: notebook §3

Definition of the state: information available to the agent

- **In one sentence:** the operator plans before the day-ahead prices are known; the agent conditions on how cheap each hour was yesterday.

$$s_h = \big(h,\ \text{stored}_h,\ q_h\big), \quad q_h = \left\lfloor \frac{\text{rank}(p_h^{\text{yesterday}})}{6} \right\rfloor \in \{0,1,2,3\}$$

- **where**
  - Rank of hour h among yesterday's 24 prices, so $q_h = 0 =$ one of yesterday's six cheapest hours
  - Table size $24 \times 4 \times 3$ states $\times$ 3 actions.

- **Training**: 1 746 days of 2019–2023, 40 passes, the epoch with the best 2024 validation revenue is kept.
- **Test**: 638 days, 1 Jan 2025 – 24 Oct 2026, used once.
- **Reward scale**: EUR/100, $\gamma = 1$ within the day.

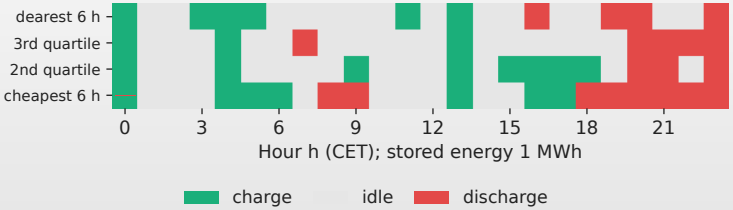
- **In practice:** a state that leaks today's prices would give a perfect but useless backtest.

(own computation)

Experiment yourself: notebook §3

The policy learned by tabular Q-learning

- **In one sentence:** with 1 MWh stored, the agent charges in hours that were cheap yesterday and sells in the evening.



Source: energy-charts.info (Fraunhofer ISE), from exchange and ENTSO-E data. Data: rows = quartile of the hour's price yesterday; seed 20261108.

- A readable policy: the table can be printed and audited hour by hour (288 states).
- The hour of the day dominates
  - Yesterday's ranking shifts the timing of charging.

(own computation)

NN_L10_q_policy

The deep Q-network: a neural approximation of the Q-table

- **In one sentence:** replace the table by a network $Q_\theta(s, a)$ and regress it on the TD target.

$$y = r + \gamma\, Q_{\theta^-}\Big(s', \arg\max_{a'} Q_\theta(s', a')\Big)$$

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\Big[\ell_{\text{Huber}}\big(y - Q_\theta(s,a)\big)\Big]$$

- **where**
  - $\mathcal{D}$ = replay memory (the last four epochs)
  - θ^- = target network, a slow copy $\theta^- \leftarrow (1-\tau)\theta^- + \tau\theta$, $\tau = 0.005$
  - Double DQN: the online network chooses a', the target network evaluates it.

- **Replay** breaks the correlation of consecutive hours (Mnih et al., 2015).
- **Double DQN** removes the upward bias of $\max_{a'}$ over noisy estimates (van Hasselt et al., 2016).
- **Our network**: 82 inputs (yesterday's and last week's 24 standardised prices, weekday, hour, stored energy), two layers of 128 rectified linear units (ReLU), 3 outputs.

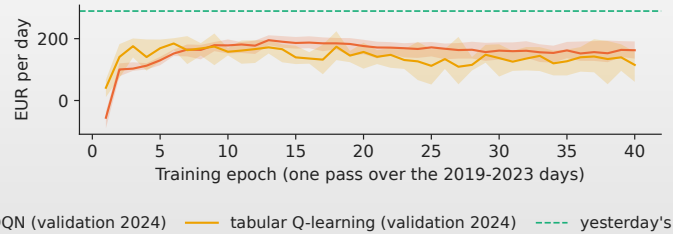
- **In practice:** function approximation + bootstrapping + off-policy data can diverge
  - Replay and target networks keep it stable.

(Mnih et al., 2015; van Hasselt, Guez & Silver, 2016)

Experiment yourself: notebook §4

Learning curves: validation revenue by training epoch

- **In one sentence:** both agents improve quickly but stay below yesterday's plan; the epoch kept is the best on 2024, never on the test years.



Source: energy-charts.info (Fraunhofer ISE), from exchange and ENTSO-E data. Data: validation year 2024; band = range over 5 seeds.

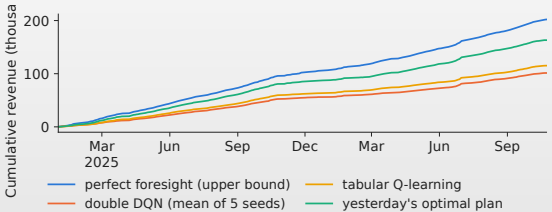
- Best validation revenue per day, mean of 5 seeds: DQN 200 EUR, Q-learning 211 EUR, yesterday's plan 289 EUR.

(own computation)

NN_L10_battery_learning

Share of the perfect-foresight revenue captured in the test years

- **In one sentence:** cumulative revenue from 1 Jan 2025 to 24 Oct 2026; the gap to the top line is the value of perfect information.



Source: energy-charts.info (Fraunhofer ISE), from exchange and ENTSO-E data. Data: 1 Jan 2025 to 24 Oct 2026 (test); trained on 2019-2023, validated on 2024.

- Perfect foresight 202.1
  - DQN 101.3
  - Q-learning 115.2
  - Yesterday's plan 163.1 thousand EUR.

(own computation)

NN_L10_battery_test_cum

Battery results on the test years, five seeds

Strategy	Revenue (k EUR)	Capture (%)	Seed range (%)
Perfect foresight (DP, upper bound)	202.1	100	–
Double DQN	101.3	50.1	43.9 – 58.5
Tabular Q-learning	115.2	57.0	50.2 – 61.9
Yesterday's optimal plan (naive)	163.1	80.7	–

- **Test of DQN against the naive plan**: daily revenue differences, Diebold-Mariano (DM) statistic with Newey-West variance (7 lags): $t = -18.99$, $p = 0.000$
  - Mean difference −96.8 EUR per day.
- DQN beats the naive plan in 0 of 5 seeds, Q-learning in 0 of 5: here the simple rule wins, because it optimises on all 24 of yesterday's prices, while the agents learn one average policy from a compressed state.

- **In practice:** report the capture ratio with its range over seeds and a test against the naive plan, never a single best run.

(Diebold & Mariano, 1995; own computation)

Experiment yourself: notebook §4

Outline

- Sequential decisions in business
- Markov decision processes and the Bellman equation
- Value-based learning: Q-learning and DQN
- Policy gradients and PPO: trading the S&P 500
- Deep hedging: a policy that minimises tail risk

Policy-gradient methods

- **In one sentence:** parameterise the policy directly and climb the gradient of the expected return.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\left(G_t - b(s_t)\right)\right]$$

- **where**
  - $J(\theta) = \mathbb{E}_{\pi_\theta}[G_0]$
  - $\nabla \log \pi$ = score function
  - $b(s_t)$ = baseline, any function of the state, that leaves the gradient unbiased and lowers its variance
  - $b = V_\phi(s_t)$ gives an actor-critic.

- **REINFORCE** (Williams, 1992): Monte Carlo G_t
  - Unbiased, high variance.
- **Policy gradient theorem** (Sutton et al., 2000): the gradient needs no derivative of the transition law, so the market can stay a black box.
- **Stochastic policies** explore by themselves and handle continuous actions.

- **In practice:** the log-likelihood trick is the same one used to train classifiers, weighted by how good the outcome was.

(Williams, 1992; Sutton et al., 2000)

Actor-critic and the advantage estimate

- **In one sentence:** a critic V_ϕ predicts the return, and the actor learns from the surprise $\hat{A}_t$.

$$\delta_t = r_{t+1} + \gamma V_\phi(s_{t+1}) - V_\phi(s_t), \qquad \hat{A}_t = \sum_{l \geq 0} (\gamma\lambda)^l \, \delta_{t+l}$$

$$\mathcal{L}_V(\phi) = \mathbb{E}\Big[\big(\hat{A}_t + V_\phi^{\text{old}}(s_t) - V_\phi(s_t)\big)^2\Big]$$

- **where**
  - Generalised advantage estimation (GAE): $\lambda = 0$ gives the one-step TD advantage (low variance, biased), $\lambda = 1$ the Monte Carlo advantage
  - We use $\gamma = 0.99$, $\lambda = 0.95$.

- **In practice:** the critic is a forecast of future reward
  - A poor critic gives noisy advantages and slow learning.

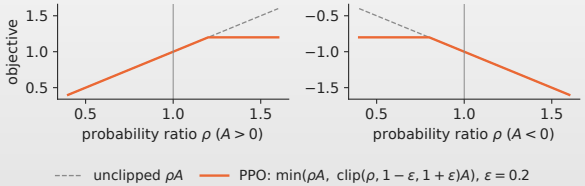
(Schulman et al., 2016)

- **Shared body**: actor and critic share two layers of 64 tanh units
  - Heads: 2 action logits and 1 value.
- **Advantage normalisation** in each batch keeps the step size comparable across market regimes.
- **Entropy bonus** 0.01 keeps the policy from collapsing too early.

PPO: clipped policy updates

- **In one sentence:** the clipped objective removes the incentive to move the probability ratio outside $[1-\epsilon, 1+\epsilon]$.

$$\mathcal{L}^{\text{CLIP}}(\boldsymbol{\theta}) = \mathbb{E}_t\Big[\min\Big(\rho_t \hat{A}_t,\ \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)\Big], \quad \rho_t = \frac{\pi_{\boldsymbol{\theta}}(a_t \mid s_t)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(a_t \mid s_t)}$$



Source: Schulman et al. (2017), clipped surrogate objective. Data: analytic function, no data.

- **SIMULATED DATA (teaching illustration)** Analytic function, no data
  - $\epsilon = 0.2$
  - A cheap substitute for the Kullback-Leibler (KL) trust region of trust region policy optimisation (TRPO).

(Schulman et al., 2017)

NN_L10_ppo_clip

Trading as an MDP: flat or long the S&P 500

- **In one sentence:** each evening the agent chooses tomorrow's position; it earns the next day's return and pays a cost when it changes position.

$$r_{t+1} = a_t \ln \frac{P_{t+1}}{P_t} - c\,|a_t - a_{t-1}|, \qquad a_t \in \{0, 1\},\ c = 5 \text{ bp}$$

- **where**
  - $s_t = 7$ features standardised on 2005–2016 (log returns over 1, 5, 21, 63 days, 21-day volatility, VIX, distance to the 200-day average) + a_{t-1}.

- **Training**: 150 PPO iterations of 64 random 126-day windows from 2005–2016, clip 0.2.
- **Selection**: the iteration with the best 2017–2019 Sharpe ratio.
- **Test**: 1 716 days from 2 Jan 2020, five seeds, against buy-and-hold and the 200-day moving average.

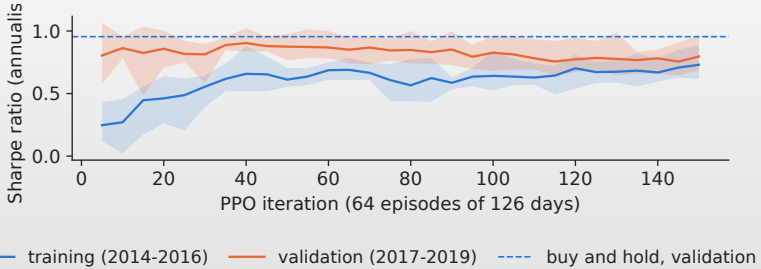
- **In practice:** put the cost in the reward, or the agent trades far too often.

(Moody & Saffell, 2001; own computation)

Experiment yourself: notebook §5

Training and validation Sharpe ratios during PPO training

- **In one sentence:** the training Sharpe ratio rises; the validation ratio is noisy, and the best iteration is kept.



Source: Yahoo Finance, daily quotes. Data: training windows from 17 Oct 2005 to 30 Dec 2016; band = range over 5 seeds.

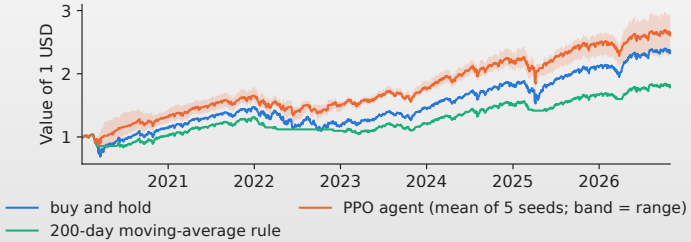
- Buy-and-hold Sharpe on validation (2017–2019): 0.95
  - The selected iteration is on average 48.

(own computation)

NN_L10_trading_learning

Test period: the PPO agent against two simple rules

- **In one sentence:** value of 1 USD invested on 2 Jan 2020, after a 5 basis-point cost per change of position.



Source: Yahoo Finance, daily quotes. Data: 2 Jan 2020 to 29 Oct 2026 (test), S&P 500, 5 bp cost per trade.

- Sharpe ratio: buy-and-hold 0.61
  - Moving-average rule 0.68
  - PPO 0.83 (seeds from 0.72 to 0.95).

(own computation)

NN_L10_trading_equity

Trading results on the test period

Strategy	Return %/yr	Vol. %/yr	Sharpe	Drawdown %	Invested %	Trades
Buy and hold	13.2	20.2	0.61	−33.9	100	1
200-day moving average	8.9	12.4	0.68	−20.5	79	41
PPO (mean of 5 seeds)	15.1	17.0	0.83	−23.2	82	147

- **Uncertainty**: Sharpe difference of the seed-averaged PPO strategy minus buy-and-hold +0.26
  - 95 % stationary block-bootstrap interval [−0.08; 0.56] (2,000 resamples, mean block 21 days).
- PPO beats buy-and-hold on the Sharpe ratio in 5 of 5 seeds.

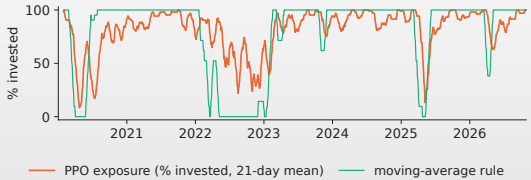
- **In practice:** an interval that contains zero means: no evidence that the agent adds value, whatever the backtest chart suggests.

(Politis & Romano, 1994; own computation)

Experiment yourself: notebook §5

Periods in which the agent is out of the market

- **In one sentence:** exposure of the PPO agent (share of seeds and days invested) against the moving-average rule.



Source: Yahoo Finance, daily quotes. Data: 2 Jan 2020 to 29 Oct 2026 (test).

- Invested: PPO 82 % of test days
  - 20 Feb – 30 Apr 2020 42 % (moving average 14 %)
  - 2022 63 % (moving average 15 %).

- **In practice:** read the policy, not only its profit: an agent that is simply always long is buy-and-hold with extra model risk.

(own computation) NN_L10_trading_exposure

Limitations of backtests for RL trading agents

- **Low signal-to-noise**: daily equity returns are close to unpredictable
  - The agent mostly learns the noise of its training windows.
- **Non-stationarity**: 2005–2016 contains 2008
  - 2020–2026 contains a pandemic crash, an inflation shock and a long rally.
- **Seed and selection risk**: five seeds already give a range of Sharpe ratios
  - Reporting the best seed is backtest overfitting.
- **Price impact and costs**: a price-taking simulator ignores market impact
  - Costs must be in the reward, not added afterwards.
- **Where RL helps more**: execution (splitting a large order), market making, hedging, storage: problems with a real state and a known reward.

- **In practice:** ask for the seed distribution, the cost assumption and a bootstrap interval before trusting any RL trading result.

(Hambly, Xu & Yang, 2023)

Outline

- Sequential decisions in business
- Markov decision processes and the Bellman equation
- Value-based learning: Q-learning and DQN
- Policy gradients and PPO: trading the S&P 500
- Deep hedging: a policy that minimises tail risk

Hedging a short call: the profit and loss of the desk

- **In one sentence:** the desk sells a one-month at-the-money (ATM) call and trades the index every day to offset the risk.

$$\mathrm{PL}(\delta) = C_0 + \sum_{k=0}^{20} \delta_k \,(S_{k+1} - S_k) - (S_{21} - K)^+ - c \sum_{k=0}^{21} S_k \,|\delta_k - \delta_{k-1}|$$

- **where**
  - S_k = S&P 500 close on day k of a real 21-day path, $S_0 = K = 1$
  - C_0 = Black-Scholes premium (VIX of day 0, rate 0), δ_k = units held over $(k, k+1]$, c = 10 bp.

- **Paths**: one per start day
  - Training 3 021 (2005–2016), validation 754, test 1 696 paths from 2020.
- **Test**: mean premium 2.38 %, mean payoff 2.53 % of the index (mean VIX 20.7).
- **Real, not simulated**: the paths keep the jumps and volatility clusters of 2008, 2020 and 2022.

- **In practice:** the premium barely covers the mean payoff
  - The risk desk manages the tail.

(Buehler et al., 2019)

Experiment yourself: notebook §6

Black-Scholes delta and the deep-hedging objective

- **In one sentence:** the benchmark hedges the model sensitivity; the network instead minimises the expected shortfall of the real loss, costs included.

$$\delta_k^{\mathrm{BS}} = \Phi(d_1), \quad d_1 = \frac{\ln(S_k/K) + \frac{1}{2}\sigma_k^2 \tau_k}{\sigma_k \sqrt{\tau_k}}$$

$$\min_{\theta, w} \; w + \frac{1}{1-\alpha}\mathbb{E}\Big[(-\mathrm{PL}(\delta_\theta) - w)^+\Big] = \min_{\theta} \mathrm{ES}_\alpha\Big(-\mathrm{PL}(\delta_\theta)\Big)$$

- **where**
  - $\sigma_k = \mathrm{VIX}/100$, $\tau_k = (21-k)/252$, $\alpha = 0.95$
  - At the optimum w is the value at risk (VaR)
  - $\delta_\theta(\ln S_k, \tau_k, \sigma_k, \delta_{k-1}) \in [0,1]$: 2 layers of 32 ReLU units, sigmoid output.

- **Policy optimisation with a known reward**: the profit and loss is differentiable in θ, so we back-propagate through all 21 days.
- **Previous hedge as input**: lets the network learn a no-trade band when costs are high.
- **Training**: 600 full-batch Adam steps, five seeds.

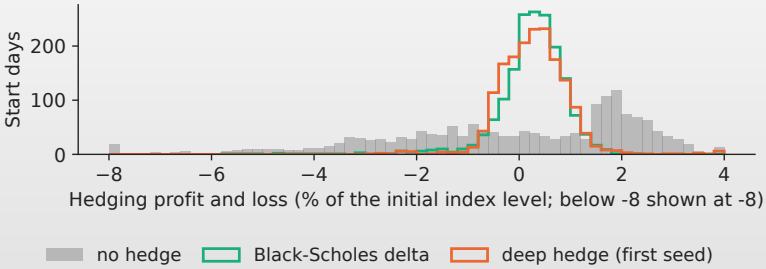
- **In practice:** the objective is the risk measure the desk reports.

(Buehler et al., 2019; Rockafellar & Uryasev, 2000)

Experiment yourself: notebook §6

Hedging profit and loss on the test paths

- **In one sentence:** hedging removes most of the left tail; the deep hedge cuts the remaining tail further, at lower turnover.



Source: Yahoo Finance, daily quotes. Data: 21-day paths starting 2 Jan 2020 to 1 Oct 2026 (test); S&P 500 and VIX; 10 bp cost.

- Standard deviation: no hedge 2.60, delta 0.74, deep hedge 0.69 (% of the index).

(own computation)

NN_L10_hedge_pnl

Hedging results on the test paths, 10 bp cost

Strategy	Mean PL (%)	Std. dev. (%)	ES_{95} of loss (%)	Turnover
No hedge	−0.15	2.60	6.82	0
Black-Scholes delta (VIX)	0.24	0.74	1.94	1.71
Deep hedge (mean of 5 seeds)	0.27	0.69	1.15	1.32

- ES_{95} of the deep hedge across seeds: 1.10 – 1.24
  - Lower than the delta hedge in 5 of 5 seeds.
- PL = profit and loss in % of the initial index level
  - Turnover = mean $\sum_k |\delta_k - \delta_{k-1}|$ per path.

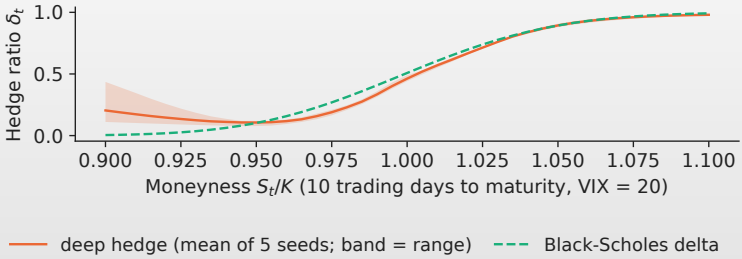
- **In practice:** judge a hedge by the tail of the loss and by its turnover, not by its mean profit.

(own computation)

Experiment yourself: notebook §6

The learned hedge ratio: flatter and adjusted for costs

- **In one sentence:** hedge ratio as a function of moneyness, 10 days before maturity, with the VIX at 20.



Source: Yahoo Finance, daily quotes. Data: policy trained on 21-day paths starting 2005-2016.

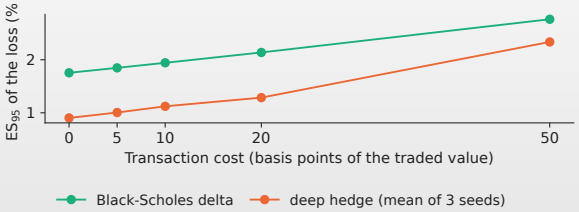
- At the money: deep hedge 0.46 against Black-Scholes 0.51
  - The band shows the spread across five seeds.

(own computation)

NN_L10_hedge_delta

Transaction costs and the advantage of the learned hedge

- **In one sentence:** ES_{95} of the hedging loss for cost levels from 0 to 50 bp; one network trained per cost level.



Source: Yahoo Finance, daily quotes. Data: test paths starting 2 Jan 2020 to 1 Oct 2026.

- No cost: delta 1.75, deep hedge 0.90
  - 50 bp: delta 2.76, deep hedge 2.34 (% of the index).

- **In practice:** the value of a learned hedge grows with frictions: illiquid underlyings, wide spreads, energy and commodity hedges.

(own computation) NN_L10_hedge_costs

Reinforcement learning in production: model risk

- **Reward design**: the agent optimises exactly what you write
  - Omit costs, risk or constraints and it will exploit the gap.
- **Simulator gap**: a historical replay ignores the agent's own price impact and future regimes.
- **Offline evaluation**: one test period used once
  - Several seeds
  - A test against the rule in use.
- **Guardrails**: action masks (state of charge, position limits), kill switches, human sign-off.
- **Explainability**: a table or a hedge-ratio curve can be audited
  - Lecture 11 explains neural policies.
- **Monitoring**: compare live capture ratio and turnover with the backtest every month.

In practice: treat an RL policy like any model of the bank: documented, validated independently, monitored, with a fallback rule.

(Hambly, Xu & Yang, 2023)

Quiz 1: perfect foresight

- A vendor shows that its battery agent earned 95 % of the dynamic-programming revenue in a backtest in which the DP used the day's own prices.
  - What do you ask?

Answer

- Whether the agent's state contained the day's prices: if so, the 95 % is look-ahead bias
  - A fair agent sees only yesterday and is compared with yesterday's optimal plan, as in our test (DQN 50.1 %, naive 80.7 %).

Quiz 2: PPO clipping

- In a PPO update an action has advantage $\hat{A}_t > 0$ and the new policy already gives it 1.5 times its old probability.
  - What gradient does the clipped objective pass to this sample with $\epsilon = 0.2$?

Answer

- Zero: $\rho_t = 1.5 > 1 + \epsilon$, the minimum picks the clipped term $1.2\,\hat{A}_t$, which does not depend on $\boldsymbol{\theta}$
  - The policy is not pushed further in this step.

Quiz 3: reading a trading result

- Your PPO agent shows a test Sharpe ratio above buy-and-hold for its best seed.
  - The bootstrap interval of the Sharpe difference of the seed-averaged strategy is $[-0.08; 0.56]$.
  - Do you deploy it?

Answer

- No: the best seed is a selected outcome, and an interval that contains zero gives no evidence of added value
  - Keep the simple rule.

Wrap-up

- **MDP + Bellman**: write state, action, reward
  - DP solves small problems exactly and gives the ceiling (battery: 202.1 thousand EUR on the test years).
- **Value-based**: Q-learning and DQN learn from yesterday's prices
  - Capture ratios 57.0 % and 50.1 %, naive plan 80.7 %.
- **Policy gradient**: PPO on the S&P 500 gives a Sharpe ratio of 0.83 against 0.61 for buy-and-hold, with a bootstrap interval that contains zero.
- **Deep hedging**: ES_{95} of the loss 1.15 % against 1.94 % for the delta hedge at 10 bp.
- **Verdict on real data**: battery — the simple rule wins
  - Trading — higher Sharpe, but not significant
  - Hedging — the learned policy wins in every seed.
- **Next**: Lecture 11 opens the black box: explaining what a neural network has learned.

Research from our group: storage, prices and learning in energy finance

- **Battery arbitrage on day-ahead markets**: joint probabilistic forecasts of the hourly price vector feed the charge-discharge decision of a battery in Central and Eastern European day-ahead markets
  - Forecasting the joint distribution, not only point prices, is what the dispatch needs (Andrei, Cepoi, Dumitrescu, Lessmann, Chiriac (Luchian), Pele & Petukhina, 2026, Social Science Research Network (SSRN) 6872381).
- **Four generations of price models**: a comparative evaluation from statistical models to machine learning, deep learning and transformer models for energy prices (Andrei, Velev, Toma, Pele & Lessmann, 2024, arXiv 2411.03372) — the forecasts a dispatch agent consumes.
- **Artificial intelligence (AI) in energy finance**: a systematic literature review of how machine learning and reinforcement learning reshape trading, risk and storage decisions in energy markets (Anagnoste et al., 2025, Energy Strategy Reviews).
- Findings as stated in the papers; numbers on the other slides come from our own data.

(Bucharest University of Economic Studies (ASE); Romanian Academy, Institute for Economic Forecasting (IPE))

Further reading and resources

- **Sutton & Barto (2018)**, the reference text, free online: incompleteideas.net/book.
- **Spinning Up in Deep RL** (OpenAI), clear derivations of policy gradients and PPO: spinningup.openai.com.
- **Hambly, Xu & Yang (2023)**: survey of RL in finance.
- **This course**: code, data, notebooks — github.com/danpele/NEURAL_BIZ.
- **Experiment yourself**: notebooks/L10_notebook.ipynb reproduces every method of this lecture
  - The seminar adds efficiency, battery size and cost experiments.
- **Deep hedging**: Buehler et al. (2019), the paper behind the last section.

References (1/2)

- Andrei, A.-V., Cepoi, C. O., Dumitrescu, B. A., Lessmann, S., Chiriac (Luchian), A.-C., Pele, D. T. & Petukhina, A. (2026). Joint probabilistic forecasting for battery arbitrage in day-ahead power markets. SSRN Working Paper 6872381. doi:10.2139/ssrn.6872381
- Anagnoste, S., Andrei, A.-V., Bolovăneanu, V., Cepoi, C.-O., Clodnitchi, R., Cramer, A.-A., Grecu, R.-A., Lessmann, S., Pele, D. T., Petukhina, A. & Strat, V. A. (2025). The role of AI in (re)shaping energy finance: A systematic literature review. *Energy Strategy Reviews*, 61, 101833. doi:10.1016/j.esr.2025.101833
- Andrei, A.-V., Velev, G., Toma, F.-M., Pele, D. T. & Lessmann, S. (2024). Energy price modelling: A comparative evaluation of four generations of forecasting methods. arXiv:2411.03372
- Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.
- Buehler, H., Gonon, L., Teichmann, J. & Wood, B. (2019). Deep hedging. *Quantitative Finance*, 19(8), 1271–1291. doi:10.1080/14697688.2019.1571683
- Diebold, F. X. & Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3), 253–263. doi:10.1080/07350015.1995.10524599
- Hambly, B., Xu, R. & Yang, H. (2023). Recent advances in reinforcement learning in finance. *Mathematical Finance*, 33(3), 437–503. doi:10.1111/mafi.12382
- Mnih, V. et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518, 529–533. doi:10.1038/nature14236
- Moody, J. & Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, 12(4), 875–889. doi:10.1109/72.935097

References (2/2)

- Politis, D. N. & Romano, J. P. (1994). The stationary bootstrap. *Journal of the American Statistical Association*, 89(428), 1303–1313. doi:10.1080/01621459.1994.10476870
- Rockafellar, R. T. & Uryasev, S. (2000). Optimization of conditional value-at-risk. *Journal of Risk*, 2(3), 21–41. doi:10.21314/JOR.2000.038
- Schulman, J., Moritz, P., Levine, S., Jordan, M. & Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. ICLR. arXiv:1506.02438
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv:1707.06347
- Sutton, R. S. & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press. incompleteideas.net/book
- Sutton, R. S., McAllester, D., Singh, S. & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1057–1063.
- van Hasselt, H., Guez, A. & Silver, D. (2016). Deep reinforcement learning with double Q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1). doi:10.1609/aaai.v30i1.10295
- Watkins, C. J. C. H. & Dayan, P. (1992). Q-learning. *Machine Learning*, 8, 279–292. doi:10.1007/BF00992698
- Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8, 229–256. doi:10.1007/BF00992696

Appendix A: convergence of Q-learning

- Bellman optimality operator $(\mathcal{T}Q)(s,a) = R(s,a) + \gamma \mathbb{E}_{s'} \max_{a'} Q(s',a')$ is a γ-contraction in the sup norm: $\|\mathcal{T}Q_1 - \mathcal{T}Q_2\|_\infty \le \gamma \|Q_1 - Q_2\|_\infty$.
- Q-learning is a stochastic approximation of the fixed point: $Q_{t+1} = (1-\alpha_t)Q_t + \alpha_t(\mathcal{T}Q_t + \varepsilon_t)$ with zero-mean noise ε_t.
- Robbins-Monro conditions $\sum_t \alpha_t(s,a) = \infty$, $\sum_t \alpha_t^2(s,a) < \infty$ and infinite visits give $Q_t \to Q^*$ with probability one (Watkins & Dayan, 1992).
- With $\gamma = 1$ on a finite horizon the contraction argument is replaced by backward induction over the 24 steps.
- A constant $\alpha = 0.05$ (as here) converges only to a neighbourhood of Q^*
  - The validation set picks the best epoch.

(Watkins & Dayan, 1992; Sutton & Barto, 2018, ch. 6)

Appendix B: settings of the three agents

Setting	Q-learning / DQN	PPO	Deep hedge
Network	table; 2×128 ReLU	2×64 tanh, shared	2×32 ReLU, sigmoid
Optimiser	$\alpha = 0.05$; Adam $3 \cdot 10^{-4}$	Adam $3 \cdot 10^{-4}$	Adam $3 \cdot 10^{-3}$
Discount / horizon	$\gamma = 1$, 24 h	$\gamma = 0.99$, $\lambda = 0.95$, 126 d	21 days
Exploration	ε: $1 \to 0.05$	entropy 0.01	–
Training budget	40 epochs; 40 updates of 512 per hour	150 iterations × 64 episodes	600 full-batch steps
Selection	best 2024 revenue	best 2017–2019 Sharpe	none (fixed budget)
Run time (5 seeds)	1 838 s	160 s	728 s

(own computation)

Experiment yourself: notebook §3–§6

Appendix C: the generalised advantage estimator

- k-step advantage: $\hat{A}_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l r_{t+l+1} + \gamma^k V(s_{t+k}) - V(s_t) = \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}$.
- GAE is the exponentially weighted average $(1-\lambda) \sum_{k \geq 1} \lambda^{k-1} \hat{A}_t^{(k)} = \sum_{l \geq 0} (\gamma\lambda)^l \delta_{t+l}$.
- Computed backwards in one pass: $\hat{A}_t = \delta_t + \gamma\lambda \hat{A}_{t+1}$, $\hat{A}_T = 0$ (the loop of analysis_L10.py).
- Bias-variance trade-off: small λ trusts the critic (biased if V is wrong), large λ trusts the sampled rewards (noisy).

(Schulman et al., 2016)