AI Powered Credit Limit Decisions

LUXEMBOURG 7 AUGUST 2025

Konstantinos Bousoulas

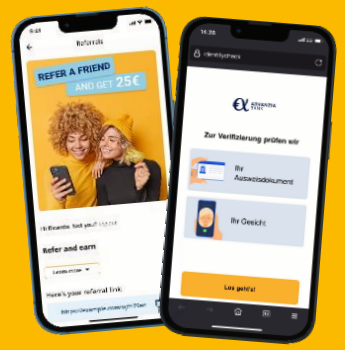
Luca Grassitelli

TABLE OF CONTENTS

ADVANZIA, A LEADING EUROPEAN DIGITAL BANK

European go-to bank for Cards-as-a-Service solutions

Partnerships with companies, associations and financial institutions to enhance brand loyalty



Digital transformation

Fully digital onboarding
Omni-channel UX
Mobile services
Digital banking platform
Data connectivity

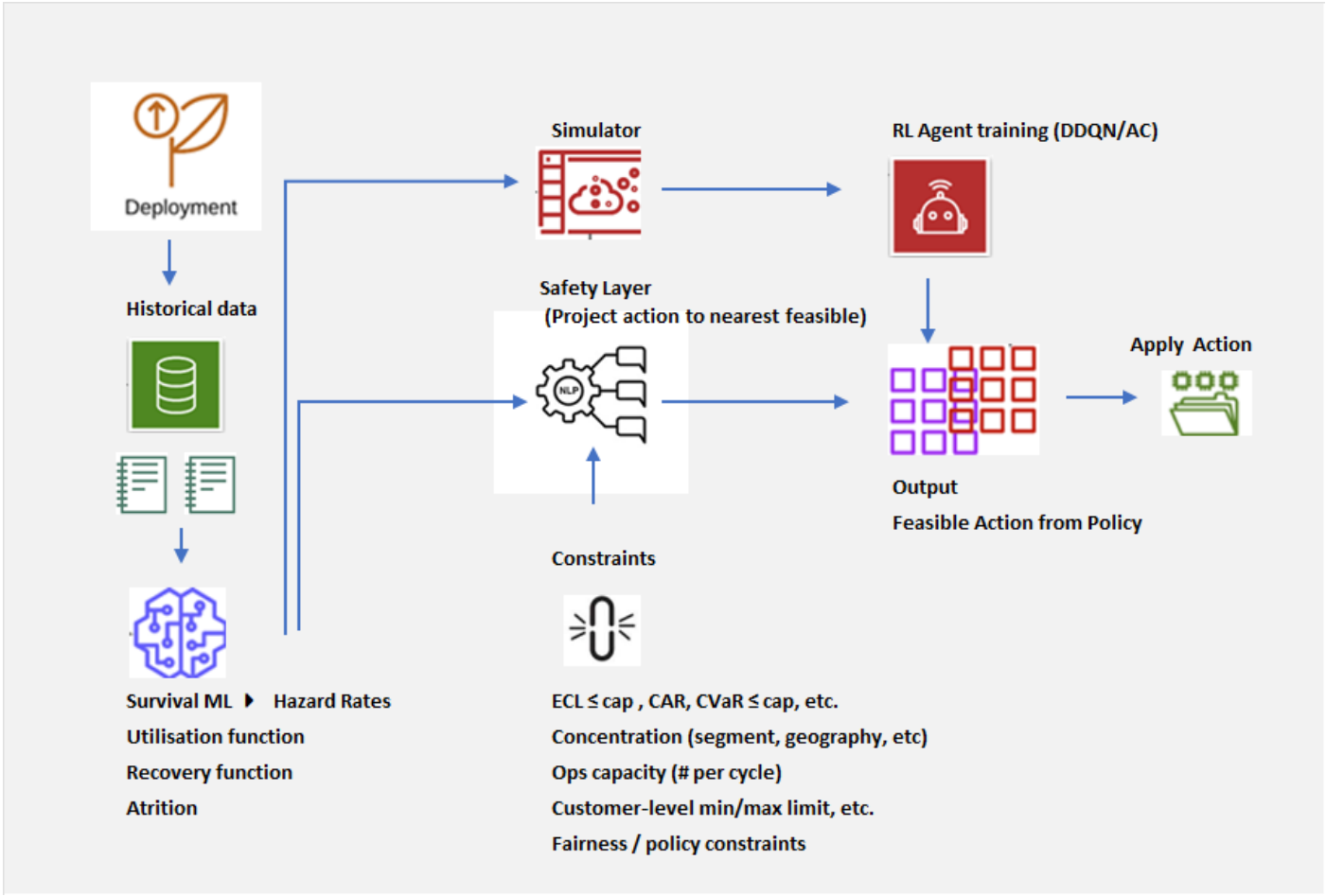
2.8 million
credit card customers

>230 employees
29 nationalities

9.9%
annual growth rate[1]

[1] Based on customer growth

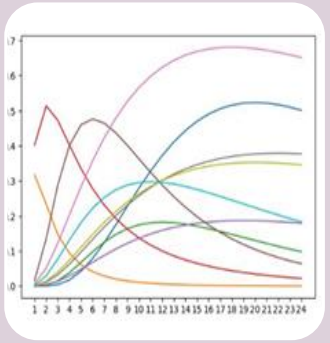
MOTIVATION



- **A data-driven limit-management framework** that combines economic optimality with operational resilience.
- **RL agent (DDQN or Actor-Critic)** learns the long-term value of limit moves under uncertainty (utilization, attrition, losses, macro shocks).
- **Deterministic NLP** solver guarantees hard constraints (capital, loss limits, concentration, fairness, ops capacity) and converts the agent's "intent" into a feasible portfolio action.

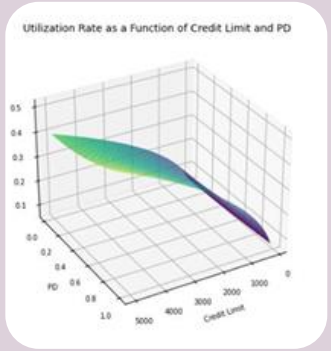
EXPERIMENTAL DESIGN - DDQN & AC AGENTS

State Input Vector



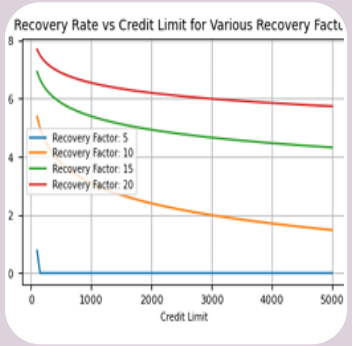
PD
Current &
Long-term view

(12-month cum-PD est. on Hazard Rates)



UR
Current &
Last 3M trend

(when UR increases Trend3M>0, indicating worsening credit risk)

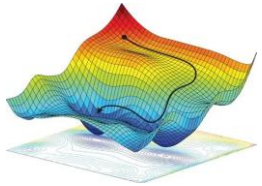


Limit
Current &
Maximum allowed

Both Agents deploy two neural nets,

DDQN: Both networks have the same architecture. One is the **online network** that learns every step; the other is a **target network** whose weights are copied from the online net only every N steps. Using the target net to compute the bootstrap term prevents the online net from chasing its own moving estimates and curbs Q over-estimation

Actor Critic: To separate policy and value roles, one network (**the actor**) outputs the policy $\pi(a \mid s)$ the other (**the critic**) outputs a state-value V(s) or an advantage estimate. The critic's evaluation supplies the gradient signal that updates the actor toward better actions.



MILP Benchmark: a piecewise-linear mixed-integer linear programming model that optimally allocates credit limits under identical portfolio conditions, including initial limits.

Action Output Space

There are 10 possible actions per customer
defining an action ladder from
no change (1.00×) to 40% increase (1.40×) in limit

[1.00, 1.04, 1.09, 1.13, 1.17, 1.22, 1.26, 1.30, 1.35, 1.40]

GOVERNING EQUATIONS – DDQN & AC AGENTS

Feature	DDQN	Actor-Critic
Core Approach	Value-based (Q-learning)	Combined Policy and Value-based
Action Space	Primarily discrete	Can handle continuous and discrete
Policy	Not explicitly learned	Actor learns a policy (stochastic or deterministic)
Value Function	Learned Q-values (action-value function)	Critic learns a value function (state-value or action-value)
Bias	Overestimation in DQN, mitigated by DDQN	Can be less prone to overestimation depending on implementation
Complexity	Relatively simple	Can be more complex, especially with off-policy methods like SAC (Soft AC)
Exploration	Relies on exploration strategies like epsilon-greedy	Exploration can be managed by the actor (e.g., entropy regularization in SAC)
Learning	Primarily on-policy or off-policy depending on the specific algorithm	—

DDQN

$$L_{DDQN}(\theta) = \mathrm{E}_{(\mathrm{s,a,r,s'})}\left[\left(\mathrm{r} + \gamma(1-\mathrm{d})\mathrm{Q}_{\bar{\theta}}(\mathrm{s}', arg\max_{a'} Q_{\theta}(a', s')\right) - Q_{\theta}(s, a)\right)\text{^}2\right]$$

Minimize this loss w.r.t. the online network θ (the target $\bar{\theta}$ is held fixed and updated periodically).

Actor–Critic

$$\theta \leftarrow \theta + \alpha\, \nabla_{\theta} \log \pi_{\theta}(\mathrm{a|s})\, [\mathrm{r} + \gamma(1-\mathrm{d})\, V_W(\mathrm{s}') - V_W(\mathrm{s})]$$

This is the actor update using the TD-error as the advantage; the critic parameters w are trained to fit V_W (usually by minimizing the squared TD-error).

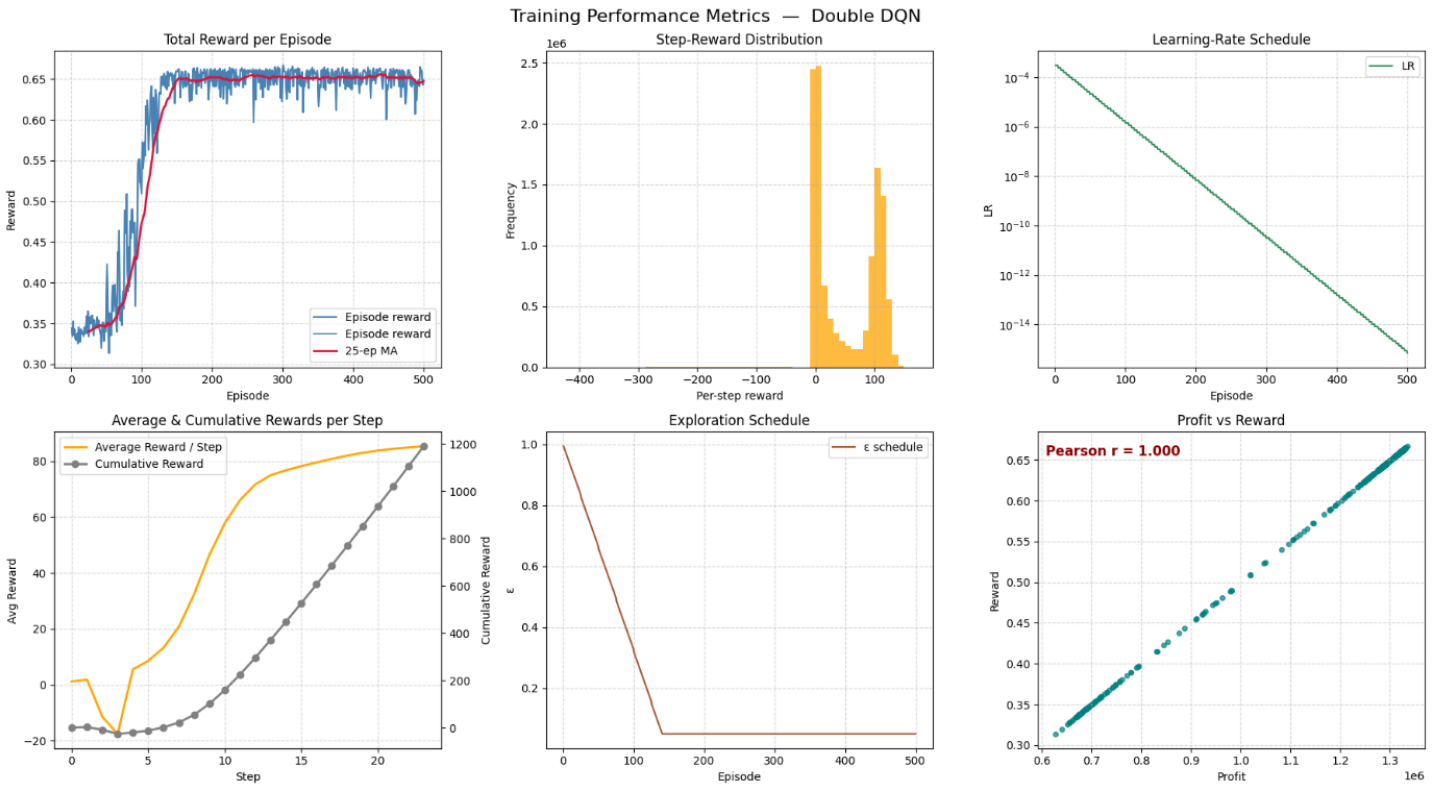
Reward function

$$r_n = \mathrm{E}(profit_{\mathrm{n}} | a_{\mathrm{n}}) = \sum_{i=1}^{m} \sum_{j=1}^{n} (Rev_{\mathrm{ij}}(\cdot) - Loss_{\mathrm{ij}}(\cdot))$$

with $Rev_{\mathrm{in}} = \sum_{j}^{n} (CL_{\mathrm{ij}} * UR_{\mathrm{ij}} * \left(1 - PD_{\mathrm{ij}}\right) * APR_{\mathrm{ij}})$ and $Loss_{\mathrm{in}} = \sum_{j}^{n} (CL_{\mathrm{ij}} * UR_{\mathrm{ij}} * PD_{\mathrm{ij}} * \left(1 - RR_{\mathrm{ij}}\right))$

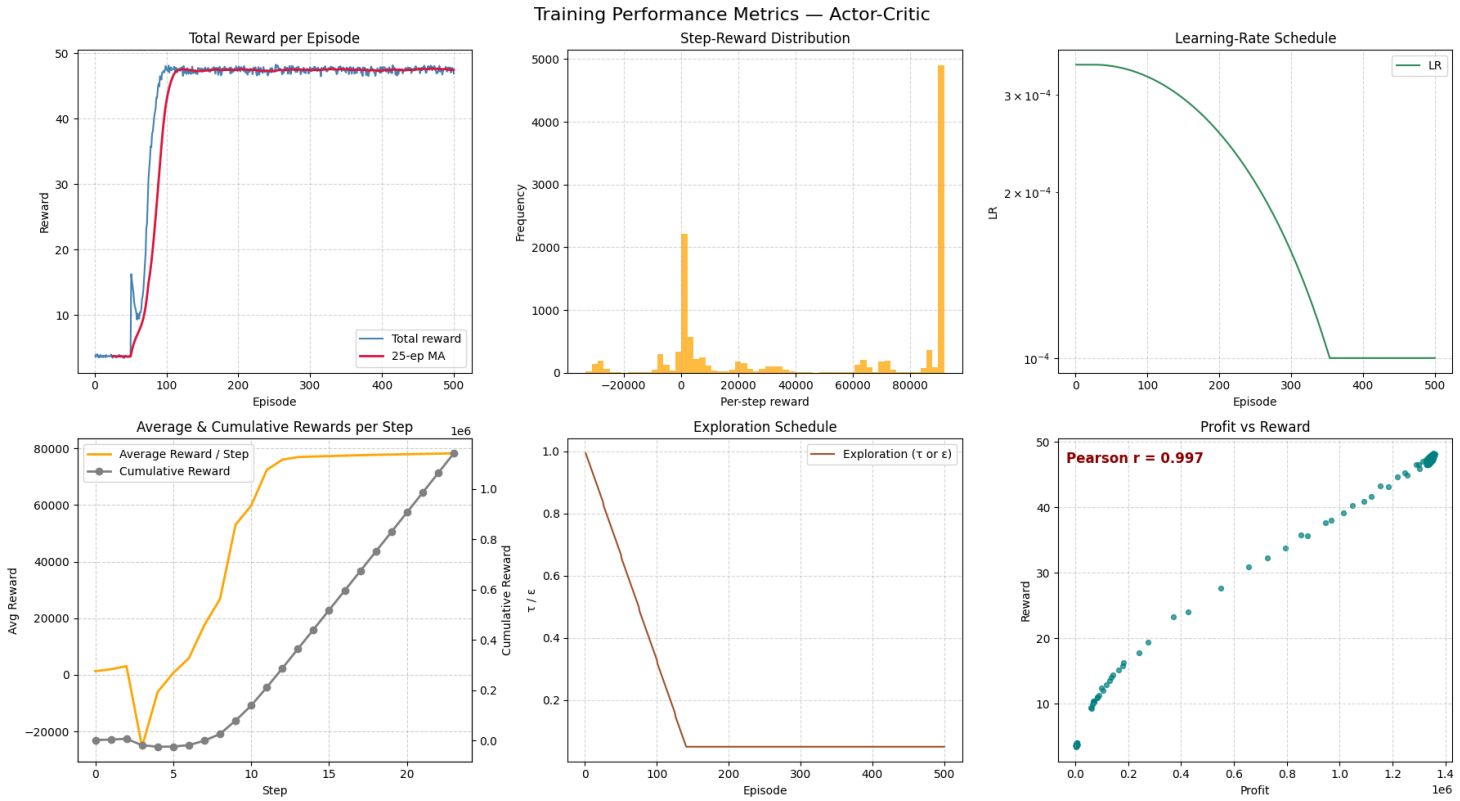
for the i -customer at time step n after performing the actions a_1 *to* a_n up to that point.

DOUBLE DQN AGENT – TRAINING PERFORMANCE



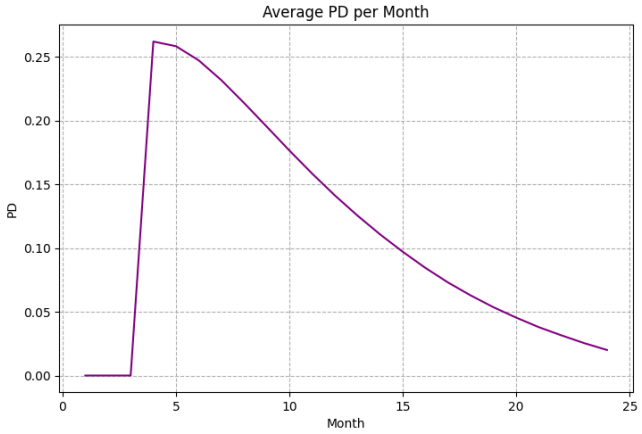
- **Rapid lift & settle**: Rewards climb from ≈ 0.3 → 0.65 within ~100 episodes and stay there. Early plateau shows strong sample efficiency.
- **Positively skewed rewards**: Step-reward histogram peaks at 50-100 with a thin negative tail. Profitable actions dominate while riskier moves keep exploration alive.
- **Exponential LR taper**: Learning rate drops smoothly by ~10^5× across training. Big early updates accelerate learning, tiny later steps lock gains.
- **Step-level turnaround**: Avg-reward flips from negative to solidly positive by step 10-15; cumulative curve rises steadily. Agent quickly pinpoints profitable credit limits and sustains them.
- **Linear ε-decay**: Exploration rate slides 1.0 → 0.05 over 350 episodes. Ample early probing followed by confident exploitation yields stable policy.
- **Profit aligned**: Pearson r = 1.000 between reward and €-profit. Signal maps exactly to real financial gain.

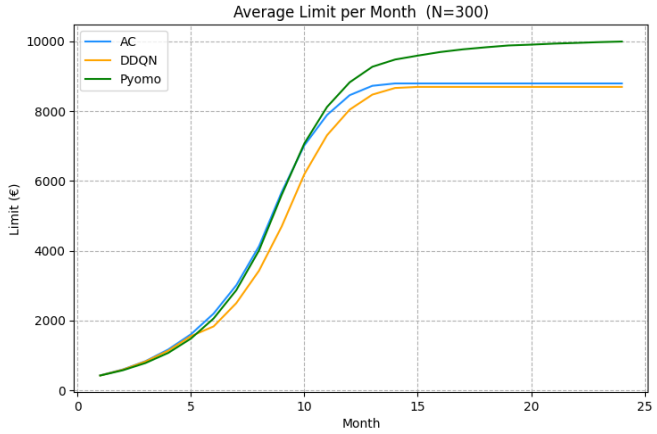
ACTOR CRITIC AGENT – TRAINING PERFORMANCE

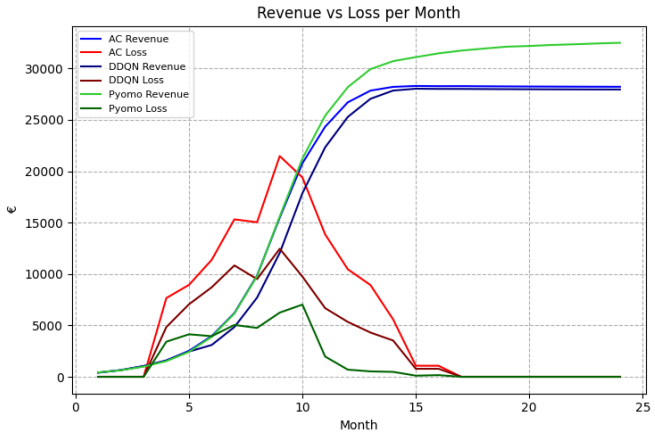


- **AC surge:** Rewards leap ~0 → 48 by ≈120 episodes and stick there. Solid, high plateau shows reliable convergence.
- **Lively exploration:** Skewed step-reward histogram signals big early variance, however, with strong gradients without premature collapse.
- **Agile learning rate:** Cosine decay (3e-4 → 1e-4) keeps adaptability. Avoids stagnation after the big jump.
- **Hockey-stick gains:** Per-step rewards spike at steps 6-10, then level; cumulative line steady. Stable marginal returns once policy locks in.
- **Smooth τ anneal:** Linear 1 → 0.05 cooling maintains graded randomness and prevents early over-commitment versus ε-greedy.
- **Profit aligned:** Reward vs. profit r = 0.997. Training signal perfectly tracks real money.

PROFIT BASED COMPARISON







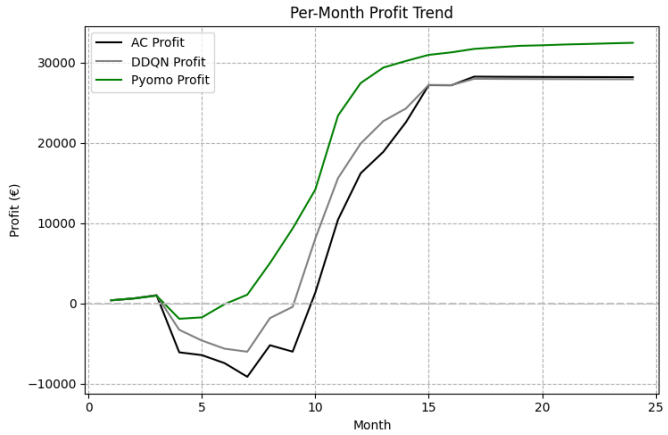


Table 1: Profit-based comparison, DDQN and Actor–Critic agents

	Pyomo (benchmark)	Actor–Critic	Double DQN
Profit at steady state			
Monthly profit	≈€31 k	≈€27 k	≈€24 k
Gap closed	100 %	85–88 %	75–78 %
Profit over 24-month window			
Cumulative profit	€458 k	€312 k*	€349 k

- **Traditional nonlinear programming remains the "standard"** when the environment is static and well-understood.
- **Learning cost versus long-run gain**. DDQN converges within ~100 episodes, incurs the smallest write-off spike, and is cash-positive from month 8. Actor-Critic needs an extra quarter of exploration and ~€25 k of additional early losses, yet by month 15 it overtakes DDQN's monthly profit and keeps closing the gap to the LP frontier
- **Limit policy matters more than PD modelling**. All three approaches face the same PD "hump" (25 % at month 4). The difference is how far they dare to lift limits through it. A 5 % gap in the final average limit makes a 10 – 12 % difference in monthly profit. Small policy tweaks compound quickly.
- **RL is competitive within bounds**. Without any hard-coded strategy the best RL agent now earns > €380 k in the same scenario (> 85 % of the deterministic optimum).

SUMMARY & RECOMMENDATIONS

DDQN learns fastest (≈100 episodes), is cash-positive by month 8, and has the smallest early write-off spike. Actor-Critic needs an extra quarter and ~€25k more early losses, but by month 15 it overtakes DDQN's monthly profit. The deterministic LP remains the frontier. The best RL agent still earns >€380k (≈85% of LP). We keep the survival-model PD/hazard outside the limit engine for governance and enforce risk, capital or ops limits via the NLP safety layer.

- **Policy choices > small model tweaks**: A 5% higher *average limit* yields 10–12% more monthly profit. All methods face the same 25% PD spike at month 4, however the gap comes from how boldly limits are raised through that spike, not from tiny PD-model refinements.

- **Implementation:** Use LP periodically to set the benchmark and let RL steer day-to-day adjustments through the NLP projector. Keep the hazard model separate so Risk can validate and Stress can replay scenarios without retraining.

- **Risk-adjusted rewards:** Train with *Reward = Profit − λ·Risk* (e.g., Profit − λ·Volatility or Profit − $\lambda \cdot VaR_{95}$).
  Choose λ so the trained policy meets your target loss volatility or VaR limit.

- **Lagrangian constraints** (soft budgets): Add penalties $\lambda_c \cdot$(usage − cap)+ for capital or ops constrains. Increase λ_c when a cap is breached, decrease when safe. This guides Actor-Critic away from costly trial-and-error in first year.

- **Speed & robustness**: offline model-based RL, hierarchical continuous actions (then snap to regulatory break-points), and macro stress tests to verify generalization.

Thank you for your attention!