Deep Reinforcement Learning (Sp25)

Instructor: Dr. Mohammad Hossein Rohban

Lecture 24 Summary

Summarized By: Arshia Gharooni

1 The Generalisation Gap

Deep RL agents trained interactively in simulators can defeat Atari at super-human level, solve continuous-control benchmarks, and learn visuomotor manipulation from scratch. Yet the same algorithms collapse when they must act in the real world without fresh interaction. The empirical disconnect between **on-line success** and **off-line brittleness** is the *generalisation gap*.

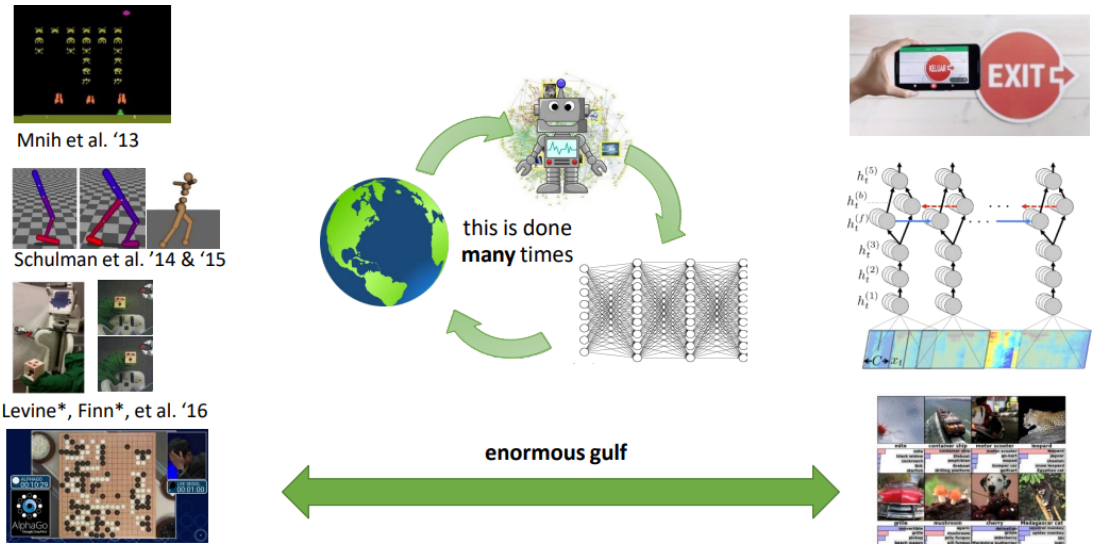


Figure 1: The generalisation gap

Offline Reinforcement Learning (ORL) addresses the question:

> **Given a fixed, finite log of past experience, can we learn a near-optimal policy *without* any further environment interaction?**

Practical motivation abounds: surgical robots cannot explore on patients; autonomous-driving data sets span petabytes; large-scale robotic fleets record everything they do. Harnessing such corpora promises "data-driven RL" — the analogue of supervised learning's ImageNet moment.

2 What Makes Modern ML Tick — and Why RL Is Different

Supervised deep learning succeeds because

1. **Massive i.i.d. data sets** supply dense coverage of the input space.
2. **Over-parameterised networks** fit those data yet generalise due to implicit regularisation by SGD.
3. **Cheap evaluation** enables rapid empirical iteration.

In RL none of these pillars holds automatically. Logged trajectories are **non-i.i.d.**, evaluation is **costly**, and the learner must predict **counterfactual outcomes** for untried actions. we should ask whether we can nevertheless replicate the supervised-learning recipe by *training on big logged corpora for many epochs, occasionally refreshing the buffer when circumstances permit*.

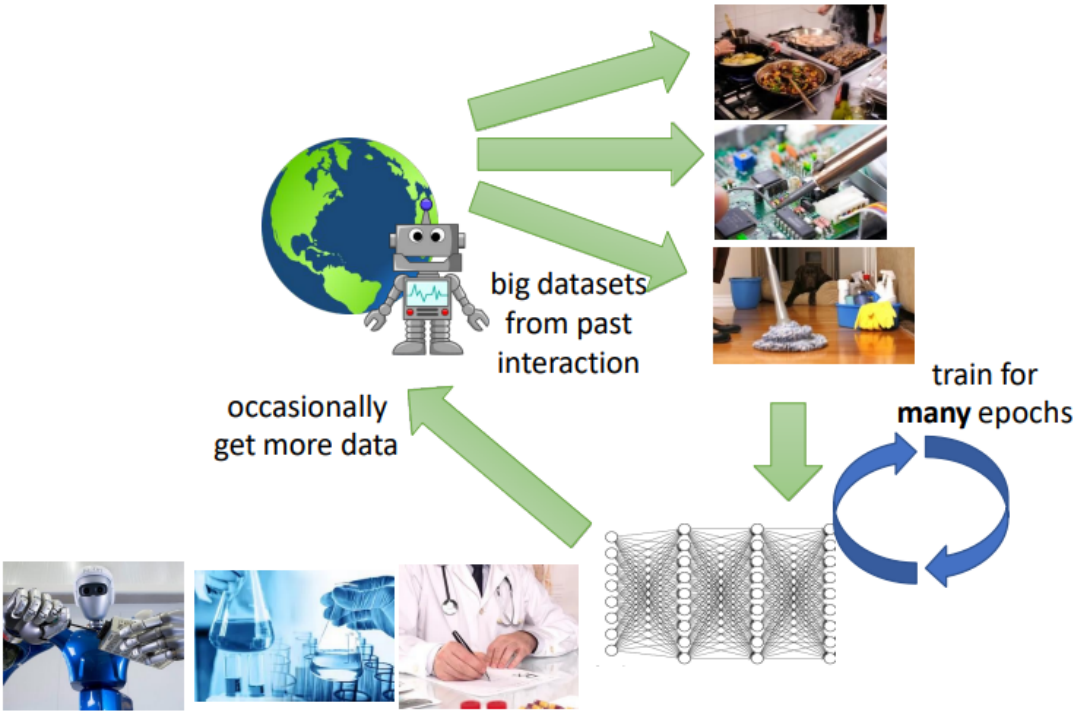


Figure 2: Data-driven RL loop

3 From On-Policy to Offline RL: Formal Foundations

Let an episodic discounted MDP be $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$. Classical *on-policy* RL iteratively samples new trajectories with its current policy. *Off-policy* RL reuses old experience but still **collects more data while learning**.

3.1 Offline RL Setting

Input A static data set

$$\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^{N} \sim d^{\beta},$$

where d^{β} is the discounted occupancy of an unknown behaviour policy β.

Goal Produce a policy

$$\pi^{\diamond} \approx \arg\max_{\pi} J(\pi), \qquad J(\pi) = \mathbb{E}_{s_0 \sim \rho_0} V^{\pi}(s_0)$$

without further interaction with P.

3.2 Distribution Shift Notation

Denote the state–action support of the data by

$$\operatorname{supp}(\mathcal{D}) = \{(s, a) : (s, a) \in \mathcal{D}\}.$$

An action a taken in state s is **out-of-distribution (OOD)** when $(s, a) \notin \operatorname{supp}(\mathcal{D})$.

4 A Taxonomy of Offline-RL Data Regimes

Regime	Composition of $\mathcal{D}$	Challenge	Typical Application
Imitation-like	Near-expert roll-outs	Avoid performance collapse	Human demos
Mixed-quality	Good + bad trajectories	Identify and prefer good	Ego-vehicle logs
Heterogeneous-skill	Fragments of high reward	"Stitch" sub-trajectories	Robot fleets

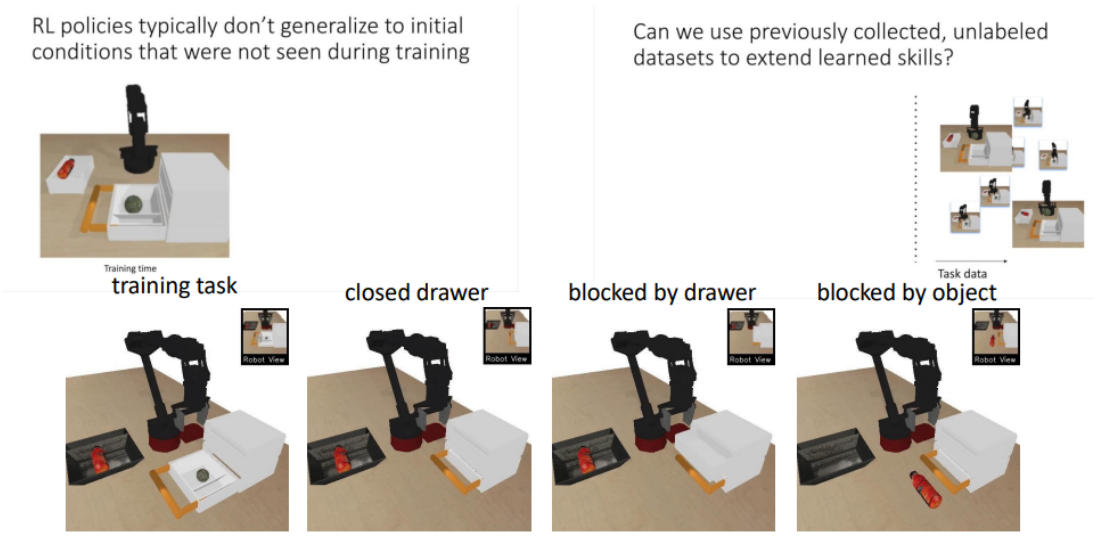
5 Intuitions, Micro- & Macro-Scale Stitching, and Case-Studies

5.1 From Imitation to Stitching

A *bad intuition* views ORL as sophisticated imitation learning. A better view is **dynamic-programming-based recombination**: propagate sparse rewards backward, connect partial successes, and synthesise **new** behaviour not literally present in any single trajectory.

5.2 COG: A Vivid Example

In COG a robot that learned to open *either* a drawer *or* a door can, offline, infer a composite strategy to *move an obstructing block, open the drawer, retrieve the key, unlock the door, and exit*.



5.3 QT-Opt Grasping

A replay buffer of **580 k previously recorded grasp episodes** suffices for an 87 % success rate; a mere **28 k on-line finetuning episodes** then lifts performance to 96 %.

Method	Dataset	Success	Failure
Offline QT-Opt	580k offline	87%	13%
Finetuned QT-Opt	580k offline + 28k online	**96%**	**4%**

6 Why Offline RL Is Hard

6.1 Counterfactual Queries

The agent must evaluate

$$Q^\pi(s,a) \quad \text{for } (s,a) \notin \text{supp}(\mathcal{D}).$$

Since the reward for such pairs is never observed, value estimates rely purely on **function-approximation extrapolation**, which may be arbitrarily wrong.

6.2 Bootstrapping Error Amplification

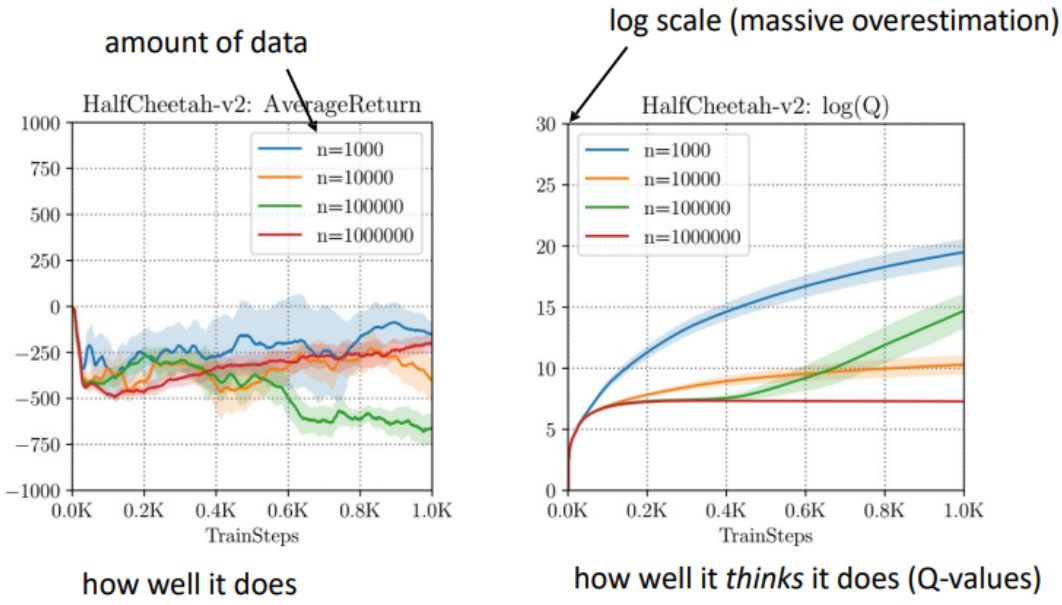
Define the *Bellman operator*

$$(\mathcal{T}Q)(s,a) = r(s,a) + \gamma \max_{a'} Q(s',a').$$

Ordinary DQN minimises $\|Q_\theta - \mathcal{T}Q_{\bar{\theta}}\|_2$. Let $\varepsilon(s,a)$ be the extrapolation error whenever a' is OOD. If $|\varepsilon| \le \delta$ on one step, repeated application yields a geometric series and an ℓ_∞ bound

$$\|Q_\theta - Q^\star\|_\infty \le \frac{\gamma}{1-\gamma}\,\delta.$$

Thus even tiny OOD errors blow up as $\gamma \to 1$. Empirically this manifests as the *massive over-estimation curve*.



7 Algorithmic Solutions

All modern techniques pursue one of two principles:

1. **Keep the learned policy inside (or close to) the data manifold.**
2. **Change the learning rule so the critic never bootstraps on OOD actions.**

7.1 Explicit Policy-Constraint Methods

We solve

$$\max_{\pi} J(\pi) \quad \text{s.t.} \quad D_{\mathrm{KL}}\big(\pi(\cdot \mid s) \,\|\, \beta(\cdot \mid s)\big) \le \epsilon, \ \forall s.$$

7.1.1 Actor-Objective Penalty

With a Lagrange multiplier λ:

$$\mathcal{L}_{\text{actor}} = \mathbb{E}_{d^\beta}\big[Q_\theta(s,a)\big] - \lambda\, \mathbb{E}_{s\sim d^\beta}\big[D_{\mathrm{KL}}(\pi \| \beta)\big].$$

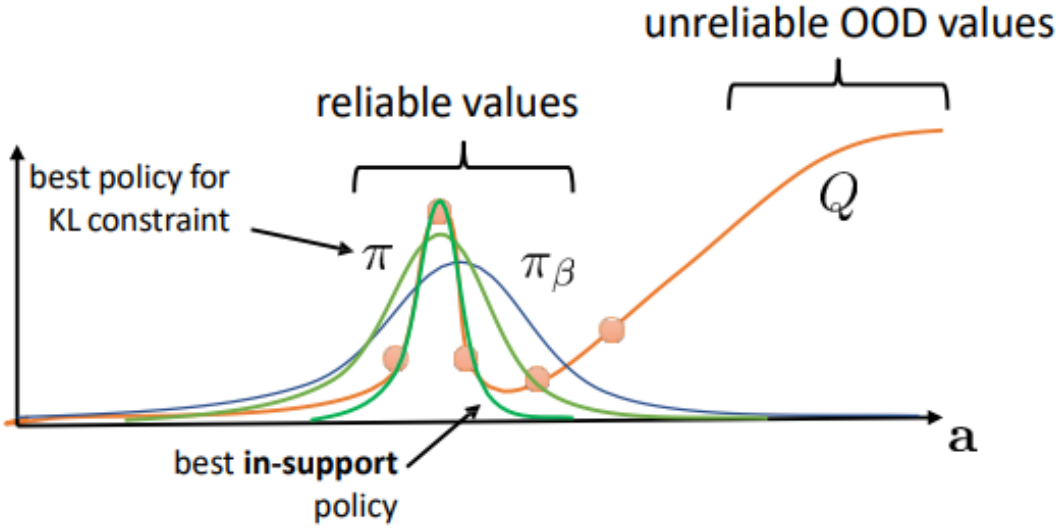
Gradient ascent requires only the log-prob density of Gaussian / categorical policies, hence is easy to implement.

7.1.2 Reward Shaping

Alternatively redefine

$$r_\lambda(s,a) = r(s,a) - \lambda \log \frac{\pi(a \mid s)}{\beta(a \mid s)}.$$

Standard Q-learning on r_λ automatically penalises future divergence.



7.2 Implicit Policy-Constraint Methods: Advantage-Weighted Regression

Start with the constrained optimisation Lagrangian

$$\mathcal{L}(\pi, \eta) = J(\pi) + \eta\big[\epsilon - \mathbb{E}_{s\sim d^\beta} D_{\mathrm{KL}}(\pi \| \beta)\big].$$

Taking functional derivatives shows the optimum obeys

$$\pi^\star(a \mid s) \propto \beta(a \mid s)\, \exp\Big(\tfrac{1}{\alpha} A_\beta(s,a)\Big), \qquad \alpha = \tfrac{1}{\eta}.$$

Hence implementation reduces to **weighted behaviour cloning** with weights

$$w(s,a) = \exp\Big(\tfrac{1}{\alpha} A_\beta(s,a)\Big),$$

where $A_\beta(s,a) = Q_\beta(s,a) - V_\beta(s)$. This is the **Advantage-Weighted Regression (AWR)** algorithm.

7.3 Support Constraints & Bootstrapping-Error Reduction: BEAR

BEAR minimises the **Maximum Mean Discrepancy (MMD)** between samples from π and β:

$$\mathrm{MMD}^2(\pi, \beta) = \left\| \mathbb{E}_{a\sim\pi}\varphi(a) - \mathbb{E}_{a\sim\beta}\varphi(a) \right\|_2^2,$$

with φ a random-Fourier feature map. The critic is then trusted only on *in-support* state–action pairs; the actor solves

$$\max_{\pi} \mathbb{E}_{d^\beta} Q_\theta(s,a) \quad \text{s.t.} \quad \mathrm{MMD}(\pi,\beta) \le \delta.$$

Empirically this yields smaller value overestimation than KL penalties while permitting more freedom than strict cloning.

7.4 Eliminating OOD in the TD Target: Implicit Q-Learning (IQL)

IQL replaces the $\max_{a'}$ in the TD target by the **expectile value**

$$V_\phi(s') = \mathrm{Expectile}_\tau\big(Q_\theta(s',a')\big), \quad \tau \in (0.5, 1).$$

Training uses

$$\mathcal{L}_Q = \big(Q_\theta(s,a) - [r + \gamma V_{\bar\phi}(s')]\big)^2.$$

Because $a' \sim \beta$, the target never steps outside the logged action support, eliminating bootstrapping error. A *deterministic greedy-in-support policy* is extracted afterwards via another advantage-weighted regression layer.

8 Theoretical Underpinnings

8.1 Duality of KL-Constrained Control

Consider the primal

$$\max_{\pi} \; J(\pi) \text{ s.t. } \mathbb{E}_{d^\beta} D_{\mathrm{KL}}(\pi \| \beta) \le \epsilon.$$

The Lagrangian is

$$\max_{\pi} \min_{\eta \ge 0} \; \mathbb{E}_{d^\beta}\Big[Q_\beta(s,a) - \eta \log \frac{\pi(a \mid s)}{\beta(a \mid s)} \Big] + \eta\epsilon.$$

Interchanging max/min and noting $\arg\max_\pi$ is exponential in the advantage yields the AWR weight formula above.

8.2 Quantitative Bootstrapping Error Amplification

Let $\Delta_0(s,a) = Q_\theta(s,a) - Q^\star(s,a)$. Assume for every in-support pair $|\Delta_0| \le \delta$ and for every OOD pair $|Q_\theta| \le B$. After one Bellman update the worst-case error obeys

$$|\Delta_1(s,a)| \le \gamma\big(\delta + (1-\kappa)B\big),$$

where $\kappa = \Pr_{a\sim\beta}\big((s,a) \in \mathrm{supp}(\mathcal{D})\big)$. Iterating shows geometric growth whenever $\kappa < 1$. This formalises slide 13's empirical plot.

8.3 Expectile Regression in IQL

For $\tau \in (0,1)$ the *expectile* is the minimiser of

$$\mathcal{L}_{\text{expectile}}(v) = \mathbb{E}_{a\sim\beta}\big|\tau - (\mathbf{1}_{Q\le v})\big|\,\big(Q_\theta(s,a) - v\big)^2.$$

Choosing $\tau > 0.5$ yields an optimistic value **within** the data-support envelope, sidestepping OOD arg-max.

9 COG Practical Implementations and Empirical Evidence

9.1 COG Skill Transfer

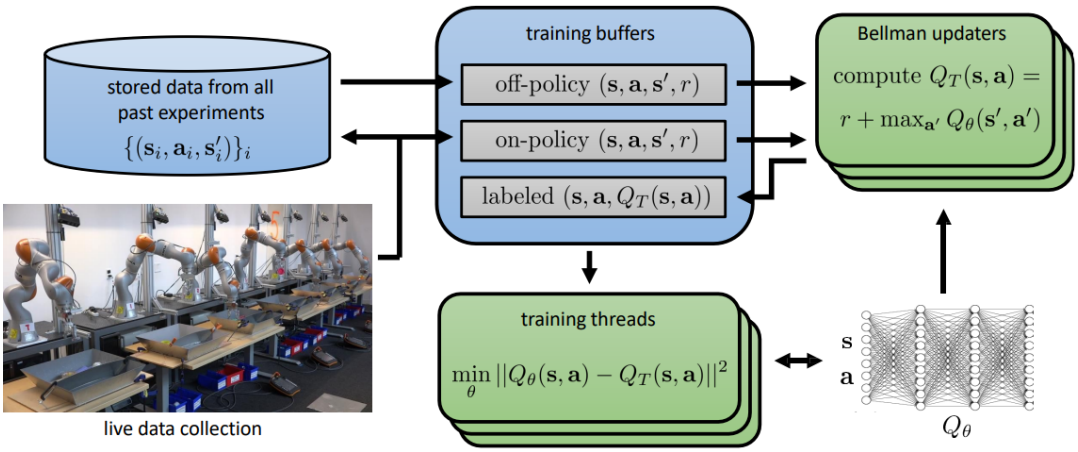
From a heterogeneous log of blocked-drawer and blocked-door interactions the agent stitches together a *five-step composite plan* achieving a task never directly demonstrated. Visual traces confirm that dynamic-programming backups propagate sparse success signals through the state graph.

9.2 QT-Opt Vision-Based Grasping

A distributed system maintains

- **Live data collection threads** (for on-line finetuning).
- **Training buffers** containing millions of high-resolution images.
- **Bellman updaters** that compute target Q-values.

Offline training to 87% success (over 580 k episodes) precedes a tiny on-line phase to 96%.



QT-Opt architecture