Robot Learning with `lerobot`: A Tutorial

Francesco Capuano ... Adil Zouitine Pepijn Kooijmans Thomas Wolf Michel Aractingi

École Normale Supérieure Paris-Saclay, Hugging Face

Abstract

Recent advancements in foundation models have enabled unprecedented progress in robotics, particularly in the development of increasingly more capable generalist robotics models. Leveraging modern learning-based approaches, the robotics community produced the first general models for tasks including locomotion and manipulation, and many said results have been openly released to the open-source community. The premise of generalist models is to learn control strategies across tasks and different robot embodiments, a long-standing challenge in robot learning. This work aims at being a tutorial on the most common techiniques used for modern robot learning, covering both the conceptual underpinnings of the most prominent approaches, and pairing them with reproducible implementations using `lerobot` (`https://github.com/huggingface/lerobot`), the full-stack open-source robotics library developed by Hugging Face. This tutorial is designed to serve as a self-contained, structured reference for robotics practictioners working with `lerobot`, as well as a practical guide for researchers in the field of robot learning.

The best material model for a cat is another, or preferably the same cat

Norbert Wiener

1 Robot Learning

Learning-based techniques for robotics naturally address the limitations presented in **??** (Figure **??**). Learning-based techniques typically rely on prediction-to-action (*visuomotor policies*), thereby directly mapping sensorimotor inputs to predicted actions, streamlining the harmonization of control policies. Mapping sensorimotor inputs to actions directly also allows to add diverse input modalities over time, leveraging the automatic feature extraction characteristic of most common learning systems. Further, learning-based approaches can in principle entirely bypass modeling efforts and instead rely exclusively on interactions data, proving transformative when dynamics are challenging to model or unknown. Lastly, we argue learning for robotics (*robot learning*) is naturally well posed to leverage the growing amount of robotics data openly available, just as computer vision first and natural language processing later did historically benefit from large scale corpora of (possibly non curated) data. As it largely leverages techniques and results developed within more mature subfields within ML (computer vision-based feature extractors, language-based semantics, etc.), robot learning is thus posed to potentially benefit from the growing amount of motion data available, in great part overlooked by dynamics-based approaches.

Robot learning is a field at its relative nascent stages, and no prevalent technique(s) proved distinctly better than others. Still, two major classes of methods gained prominence in applications of ML to robotics: reinforcement learning (RL) and behavioral cloning (BC). In this section, we provide a conceptual overview of applications of the former to robotics, as well as introduce practical examples of how to use RL within `lerobot`. We then introduce the major limitations RL suffers from, introducing BC shortly after. Importantly, (Figure **??**) we decided to include generalist robot models (**?**Shukor et al., 2025) alongside task-specific BC methods. While very different in spirit—generalist models learn to take instructions as input and to use them to generate motion valid across many tasks—foundation models like π_0, for instance, are largely trained to reproduce trajectories contained in a large training set of input demonstrations. This distinction being made, we argue generalist policies can indeed be grouped alongside other BC methods.

Figure **??** illustrates this classification graphically, explicitly listing all the robot learning policies currently available in `lerobot`: Action Chunking with Transformers (ACT) (Zhao et al., 2023), Diffusion Policy (Chi et al., 2024), Vector-Quantized Behavior Transformer (VQ-BeT) (Lee et al., 2024), π_0 (Black et al., 2024), SmolVLA (Shukor et al., 2025), Human-in-the-loop Sample-efficient RL (HIL-SERL) (Luo et al., 2024) and TD-MPC (Hansen et al., 2022).

1.1 RL for Robotics

Applications of RL to robotics have been long studied, to the point the relationship between these two disciplines has been compared to that between physics and matematics (Kober et al.). Indeed, due to their interactive and sequential nature, many robotics problems can be directly mapped to RL problems. Figure **??** depicts two of such cases. Manipulating a tiling belt into its correct position is a sequential problem where at each cycle the controller needs to adjust the position of the robotic arms based on their current configuration and the state of the tiling belt. In this example, sequentiality arises from the impossibility of placing the belt in its goal position right away from any starting position. Figure **??** also shows an example of a locomotion problem, where sequentiality is inherent in the problem formulation. Sliding to the side, the controller has to constantly keeps adjusting to the robot's propioperception, avoiding failure (falling).

1.1.1 A (Concise) Introduction to RL

The RL framework (Sutton and Barto, 2018), which we briefly introduce here, has often been used to model robotics problems (Kober et al.). RL is a subfield within ML fundamentally concerned with the development of autonomous systems (*agents*) learning how to *continuously behave* in an evolving environment, developing control *policies* to better act when in a given situation. Crucially for robotics, agents improve via trial-and-error only, entirely bypassing the need to develop explicit models of the problem dynamics, and rather exploiting interaction data only. In RL, this feedback loop between actions and outcomes is established through the agent sensing a scalar quantity (*reward*, Figure **??**).

Formally, interactions between an agent and its environment are typically modeled via a Markov Decision Process (MDP). Representing robotics problems via MDPs offers several advantages, including (1) incorporating uncertainty through MDP's inherently stochastic nature and (2) providing a theoretically sound framework for learning without an explicit dynamic model. While accommodating also a continuous time formulation, MDPs are typically considered in discrete time in RL, considering interactions to atomically take place over the course of discrete *timestep* $t = 0, 1, 2, 3, \dots, T$. MDPs allowing for an unbounded number of interactions ($T \to +\infty$) are typically termed *infinite-horizon*, and opposed to *finite-horizon* MDPs in which T cannot grow unbounded. Unless diversely specified, we will only be referring to discrete-time episodic MDPs.

Formally, a Markov Decision Process (MDP) is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{D}, r, \gamma, \rho)$, where:

- $\mathcal{S}$ is the *state space*; $s_t \in \mathcal{S}$ denotes the (possibly non-directly observable) environment state at time t. In robotics, states often comprise robot configuration and velocities $(q_t, \dot{q}_t)$, and can accomodate sensor readings such as camera or audio streams.
- $\mathcal{A}$ is the *action space*; $a_t \in \mathcal{A}$ may represent joint torques, joint velocities, or even end-effector commands. In general, actions correspond to commands intervenings on the configuration of the robot.
- $\mathcal{D}$ represents the (possibly non-deterministic) environment dynamics, with $\mathcal{D} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ corresponding to $\mathcal{D}(s_{t+1}, s_t, a_t) = \mathbb{P}(s_{t+1} | s_t, a_t)$. For instance, for a planar manipulator dynamics can typically be considered deterministic when the environment is fully described (Figure **??**), and stochastic when disturbances depending on non-observable parameters intervene (Figure **??**).
- $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the *reward function*, weighing (s_{t+1}, s_t, a_t) in the context of an arbitrary goal. For instance, a simple reward function for quickly moving the along the x axis in 3D-space could be based on the position of the robot along the x axis (p_x), present negative penalties for falling over (measured from p_z) and a introduce bonuses $\dot{p}_x$ for speed.

Lastly, $\gamma \in [0, 1)$ represent the discount factor regulating preference for immediate versus long-term reward (with a horizon of $\frac{1}{1-\gamma}$), and ρ is the distribution defined over $\mathcal{S}$, to sample the initial state, $s_0 \sim \rho$.

For MDPs, a *trajectory* of length T is the (random) sequence

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T), \tag{1}$$

with per-step rewards defined as $r_t = r(s_{t+1}, s_t, a_t)$ for ease of notation.

Interestingly, assuming both the environment dynamics and conditional distribution over actions given states—termed *policy*—to be *Markovian*:

$$\mathbb{P}(s_{t+1} | s_t, a_t, s_{t-1}, a_{t-1}, \dots s_0, a_0) = \mathbb{P}(s_{t+1} | s_t, a_t) \tag{2}$$

$$\mathbb{P}(a_t | s_t, a_{t-1}, s_{t-1}, s_0, a_0) = \mathbb{P}(a_t | s_t) \tag{3}$$

The probability of observing a given trajectory τ can be piece-wise factorized into

$$\mathbb{P}(\tau) = \mathbb{P}(s_0) \prod_{t=0}^{T-1} \mathbb{P}(s_{t+1} | s_t, a_t)\ \mathbb{P}(a_t | s_t). \tag{4}$$

Policies $\mathbb{P}(a_t | s_t)$ are typically indicated as $\pi(a_t | s_t)$, and often parametrized via θ, yielding $\pi_\theta(a_t | s_t)$. Thus, one can define the (discounted) *return* associated to τ as the (random) sum of the reward measured over it,

$$G(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t$$

Control agents seek to learn strategies (policies, π_θ) maximizing the expected return $\mathbb{E}G(\tau)$. For a given dynamics $\mathcal{D}$—i.e., for a given problem—taking the expectation over the (possibly random) trajectories resulting from acting according to a certain policy provides a direct, goal-conditioned ordering of all the possible policies obtained, yielding the (maximization) target $J : \Pi \mapsto \mathbb{R}$

$$J(\pi_\theta) = \mathbb{E}_{\tau \sim \mathbb{P}_{\theta;\mathcal{D}}}[G(\tau)], \tag{5}$$

$$\mathbb{P}_{\theta;\mathcal{D}}(\tau) = \rho \prod_{t=0}^{T-1} D(s_{t+1}, s_t, a_t)\ \pi_\theta(a_t | s_t). \tag{6}$$

MDPs thus naturally provide a framework to optimize over the space of the possible behaviors an agent might enact ($\pi \in \Pi$), searching for the *optimal policy* $\pi^* = \arg\max_\theta J(\pi_\theta)$. A variety of methods have been developed in RL as standalone attemps to find (approximate) solutions to the problem of maximizing cumulative reward (Figure **??**. Popular approaches to continuous state and action space—such as those studied within robotics—include Schulman et al. (2017a,b); Haarnoja et al. (2018). Across manipulation (OpenAI et al., 2019) and locomotion (Lee et al., 2020) tasks, RL proved very effective in providing a platform to (1) adopt a unified, streamlined perception-to-action pipeline, (2) natively integrate propioperception with multi-modal high-dimensional streams, learning data-driven pre-processing pipelines, (3) disregard a description of the environment dynamics, by focusing on observed interaction data rather than modeling, and (4) anchor policies in the experience collected in offline datasets. For a more complete survey of applications of RL to robotics, we refer the reader to Kober et al.; Tang et al. (2025).

1.1.2 Real-world RL for Robotics

Streamlined end-to-end control pipelines, data-driven feature extraction and a disregard of explicit modeling in favor of interaction data are all features of RL for robotics. However, in the context of real-world robotics RL suffers from limitations concerning machine safety and learning efficiency.

RL explore erratically, which makes learning at the beginning dangerous for the hardware considered Also, even the most sample efficient RL methods needs thousands of samples to learn to perform tasks from scratch. Further, accommodating for the RL setup is non trivial learning directly on the real robot because of challenges with ensuring reset etc. So people have studied learning in simulation. This resolves issues with safety inherently, and improves on the practical aspects of using RL because many samples in simulation are attainable in a fraction of the time compared to real-world training. But simulation is different from the real world, and transferability problems arise. People tried strategies including randomizing simulators to induce robustness to domain changes, or use real-world data to align simulations with the end result. However simulators are also complex to build, and as a result using RL in real-world is still challenging.

1.1.3 RL in `lerobot`: data-driven, sample-efficient, and with human in-the-loop

Recent work proposed leveraging for RL in the real-world using HIL-SERL

HIL-SERL is ... HIL-SERL addresses ... HIL-SERL works by ... HIL-SERL presents limitations ...

1.1.4 Code Example: Training RL Policies with `lerobot`

Outside of simulation environments, lerobot implements SERL

coding example of how HIL-SERL works

1.1.5 Limitations of RL in Real-World Robotics: Simulators and Reward Design

simulators for dexterous tasks are complex to build. Also, soft materials are very hard to model reward design is also increasingly complex to derive for practical problem, and it is a big source of brittleness

References

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. \$$\pi$_0\$: A Vision-Language-Action Flow Model for General Robot Control, October 2024.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion, March 2024.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, August 2018.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal Difference Learning for Model Predictive Control, July 2022.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement Learning in Robotics: A Survey.

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning Quadrupedal Locomotion over Challenging Terrain. *Science Robotics*, 5(47):eabc5986, October 2020. ISSN 2470-9476. doi: 10.1126/scirobotics.abc5986.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior Generation with Latent Actions, June 2024.

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and Dexterous Robotic Manipulation via Human-in-the-Loop Reinforcement Learning, October 2024.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's Cube with a Robot Hand, October 2019.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust Region Policy Optimization, April 2017a.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017b.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A Vision-Language-Action Model for Affordable and Efficient Robotics, June 2025.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning Series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018. ISBN 978-0-262-03924-6.

Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep Reinforcement Learning for Robotics: A Survey of Real-World Successes. *Annual Review of Control, Robotics, and Autonomous Systems*, 8 (Volume 8, 2025):153–188, May 2025. ISSN 2573-5144. doi: 10.1146/annurev-control-030323-022510.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware, April 2023.