Robot Learning: A Tutorial

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

École Normale Supérieure Paris-Saclay, Hugging Face

Abstract

Robot learning is at an inflection point, driven by rapid advancements in machine learning and the growing availability of large-scale robotics data. This shift from classical, model-based methods to data-driven, learning-based paradigms is unlocking unprecedented capabilities in autonomous systems. This tutorial navigates the landscape of modern robot learning, charting a course from the foundational principles of Reinforcement Learning and Behavioral Cloning to generalist, language-conditioned models capable of operating across diverse tasks and even robot embodiments. This work is intended as a guide for researchers and practitioners, and our goal is to equip the reader with the conceptual understanding and hands-on tools necessary to understand and contribute to developments in robot learning.

Code: https://github.com/huggingface/lerobot
Date: **September 9, 2025**

Contents

Foreword

Robotics, an inherently multidisciplinary field, has witnessed unprecedented advancements since its inception in the 1960s. Over the decades, numerous disciplines have shown immense promise in tackling the challenges of creating autonomous systems. Yet, more than sixty years after the debut of Unimate, robots have still not fully integrated into the rich, unstructured, and dynamic world we humans inhabit. This tutorial takes a clear stance in the debate on whether modern Machine Learning can play a pivotal role in the development of autonomous robot systems: we believe this to be the case.

However, we also hold that the wealth of research from both academia and industry in classical robotics over the past six decades is too valuable to be cast aside for purely learning-based methods. However, the interplay between classical robotics and modern machine learning is still in its nascent stages, with the path to integration yet to be clearly defined. While this tutorial focuses on learning-based approaches, we acknowledge the immense potential in combining these two techniques. Our goal in here is to present what we consider the most relevant and impactful approaches within robot learning today, and we warmly extend an invite to collaborate to expand the breadth of this work.

This tutorial covers a diverse range of topics in robotics and deep learning. Our aim is to provide a comprehensive conceptual guide, which has necessarily required us to omit certain details for the sake of clarity and focus.

This tutorial...

- Does *not* aim to be a comprehensive guide to robotics or manipulation: Siciliano and Khatib (2016) and Tedrake (a) do this better than we ever could.
- Does *not* aim to be an introduction to statistical or deep learning: Shalev-Shwartz and Ben-David (2014) and Prince (2023) cover these subjects better than we ever could.
- Does *not* aim to be a deep dive into Reinforcement Learning, Diffusion Models, or Flow Matching: invaluable works such as Sutton and Barto (2018), Nakkiran et al. (2024), and Lipman et al. (2024) do this better than we ever could.

Instead, our goal is to illuminate how these disparate ideas have converged to form the exciting field of modern robot learning. We aim to provide intuition about why these techniques have been so effective together, driving the unprecedented progress we see today. In this spirit, we follow the adage: "a jack of all trades is a master of none, *but oftentimes better than a master of one.*"

We sincerely hope this tutorial serves as a valuable starting point for your journey into robot learning

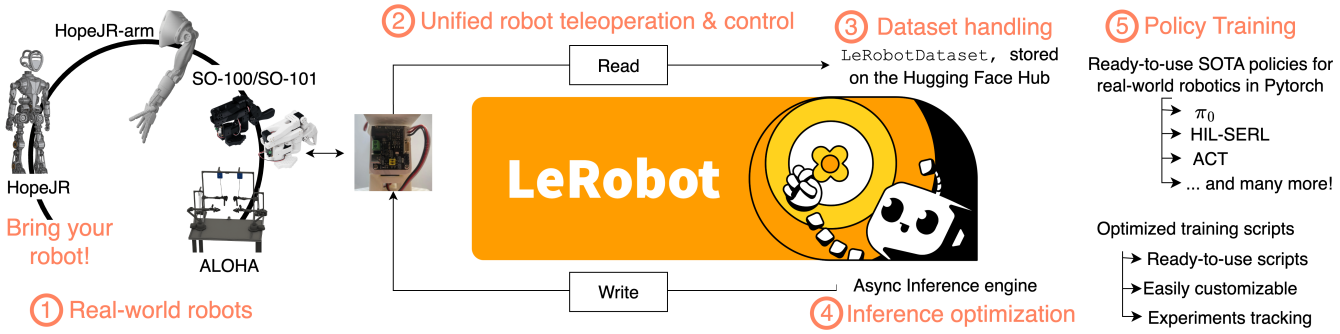


Figure 1 | `lerobot` is the open-source library for end-to-end robotics developed by Hugging Face. The library is vertically integrated on the entire robotics stack, supporting low-level control of real-world robot devices, advanced data and inference optimizations, as well as SOTA robot learning methods ported in pure Pytorch.

1 Introduction

Autonomous robotics holds the premise of relieving humans from repetitive, tiring or dangerous manual tasks. Consequently, the field of robotics has been widely studied since its first inception in the 1950s. Lately, advancements in Machine Learning (ML) have sparked the development of a relatively new class of methods used to tackle robotics problems, leveraging large amounts of computation and data rather than human expertise and modeling to develop autonomous systems.

Robotics in 2025 sits is increasingly moving away from classical model-based control paradigm, embracing the advancements made in ML, thus unlocking (1) monolithic perception-to-action action pipelines and (2) multi-modal data-driven feature extraction strategies, together with (3) reduced reliance on precise models of the world and (4) a better positioning to benefit from the growing availability of robotics data openly available. While central problems in manipulation, locomotion and whole-body control demand knowledge of rigid-body dynamics, contact modeling, planning under uncertainty, recent results seem to indicate learning can prove as effective as explicit modeling, sparking interest in the field of *robot learning*. This interest can be largely justified considering the significant challenges related to deriving accurate models of robot-environment interactions. Also, because end-to-end learning on large and increasing amounts of data produced *foundation models* capable to semantically reason over multiple input modalities (images, text, audio, etc.), deriving methods for robotics based on learning seems particularly promising, given the current trends related to the growth of openly available datasets.

At its core, robotics is also an inherently multi-disciplinary field, to the very least considering the diverse skills required in terms of *software* and *hardware*. Integrating learning-based techniques increases the heterogeneity of skills needed to tackle robotics, whether in applications or research. `lerobot`is an open-source library end-to-end integrated with the entire robotics stack. With a strong focus on accessible, real-world robots, (1) `lerobot`supports many, openly available, robotic platforms for manipulation, locomotion and even whole-body control. `lerobot`'s (2) unified, low-level approach to reading/writing robot configurations also allows to extend support for other robots relatively easily. The library also supports (3) a native robotics dataset's format—`LeRobotDataset`—that is used to efficiently share datasets. `lerobot`also supports many state-of-the-art (SOTA) algorithms in robot learning—Reinforcement Learning (RL) and Behavioral Cloning (BC)—with efficient implementations in Pytorch and extended support to experimentation and experiments tracking. Lastly, `lerobot`defines a custom, optimized inference stack for robotic policies decoupling action planning from action execution, proving effective in guaranteeing more adaptability at runtime.

This tutorial serves the double purpose of providing useful references for the science and practical use of common robot learning techniques. To this aim, we strike to provide rigorous yet concise overviews of the core concepts behind the techniques presented, paired with practical examples of how to use these in applications, for practitioners and researchers in the field of robot learning. This tutorial is structured as follows:

- Section 2 reviews classical robotics foundations, introducing the limitations of dynamics-based approaches to robotics.
- Section 3 elaborates on the limitations of dynamics-based methods, and introduces learning-based techniques, their upsides and potential limitations.

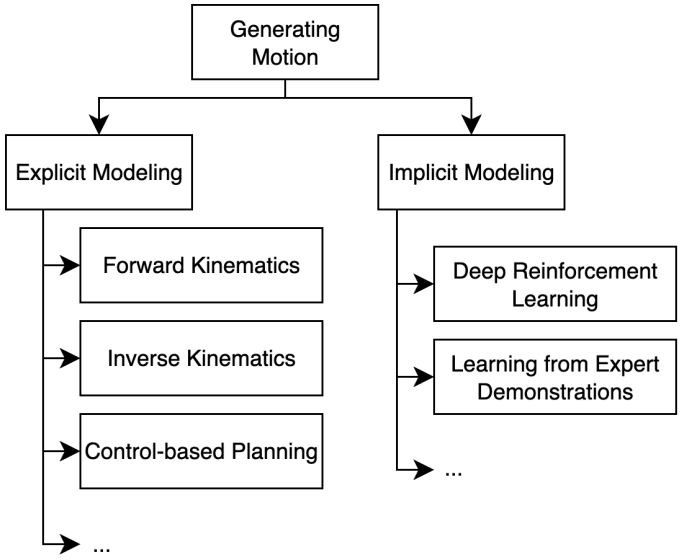


Figure 2 | Overview of methods to generate motion (clearly non-exhausitve, see Bekris et al. (2024)). The different methods can be grouped based on whether they explicitly (*dynamics-based*) or implicitly (*learning-based*) model robot-environment interactions.

- Section 4 further describes robot learning techniques that aim at solving single tasks learning from specific expert demonstrations.
- Section 5 presents more recent contributions on developing generalist models for robotics applications learning from large corpora of multi-task & multi-robot data.

We complement our presentation of the most common and recent approaches in robot learning with practical code implementations using `lerobot`.

2 Classical Robotics

Know your enemy [...]

Sun Tzu

TL;DR

Learning-based approaches to robotics are motivated by the need to (1) generalize across tasks and embodiments (2) reduce dependency on human expertise (3) leverage historical trends on the production of data—all traditionally overlooked by dynamics-based techniques.

2.1 Explicit and Implicit Models

Robotics is concerned with producing artificial motion in the physical world in useful, reliable and safe fashion. Thus, robotics is an inherently multi-disciplinar domain: producing autonomous motion in the physical world requires, to the very least, interfacing different software (motion planners) and hardware (motion executioners) components. Further, knowledge of mechanical, electrical, and software engineering, as well as rigid-body mechanics and control theory have therefore proven quintessential in robotics since the field first developed in the 1950s. More recently, Machine Learning (ML) has also proved effective in robotics, complementing these more traditional disciplines (Connell and Mahadevan, 1993). As a direct consequence of its multi-disciplinar nature, robotics has developed as a rather wide array of methods, all concerned with the main purpose of producing artificial motion in the physical world.

Methods to produce robotics motion range from traditional *explicit* models—dynamics-based methods, leveraging precise descriptions of the mechanics of robots' rigid bodies and their interactions with eventual obstacles in the

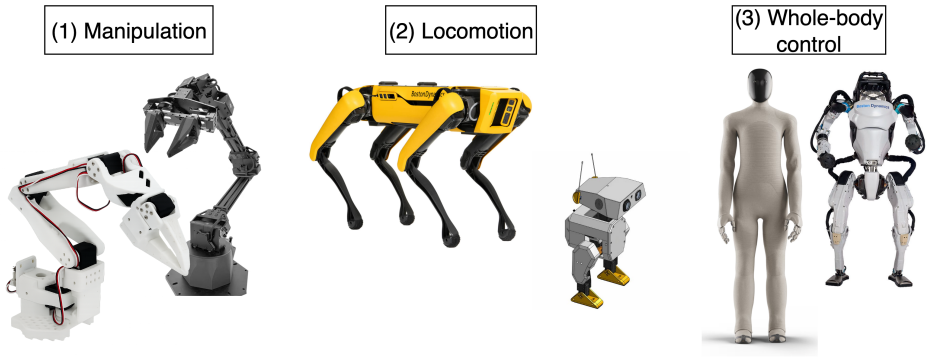


Figure 3 | Different kinds of motions are achieved with potentially very different robotic platforms. From left to right, top to bottom: ViperX, SO-100, Boston Dynamics' Spot, Open-Duck, 1X's NEO, Boston Dynamics' Atlas. This is an example list of robotic platforms and is (very) far from being exhaustive.

environment—to *implicit* models—learning-based methods, treating artificial motion as a statistical pattern to learn given multiple sensorimotor readings (Agrawal; Bekris et al., 2024). A variety of methods have been developed between these two extrema. For instance, Hansen et al. (2022) show how learning-based systems can benefit from information on the physics of problems, complementing a traditional learning method such as Temporal Difference (TD)-learning Sutton and Barto (2018) with Model-Predictive Control (MPC). Conversely, as explicit models may be relying on assumptions proving overly simplistic—or even unrealistic—in practice, learning can prove effective to improve modeling of complex phenomena or complement perception (McCormac et al., 2016). Such examples aim at demonstrating the richness of approaches to robotics, and Figure 2 graphically illustrates some of the most relevant techniques. Such a list is clearly far from being exhaustive, and we refer to Bekris et al. (2024) for a more comprehensive overview of both general and application-specific methods for motion generation. In this section, we wish to introduce the inherent benefits of learning-based approaches to robotics—the core focus on this tutorial.

2.2 Different Types of Motion

At its very core, robotics deals with producing motion via actuating joints connecting nearly entirely-rigid links. A key distinction between focus areas in robotics is based on whether the generated motion modifies (1) the absolute state of the environment (via dexterity), (2) the relative state of the robot with respect to its environment (exercising mobility skills), or (3) a combination of the two (Figure 3).

Effects such as (1) are typically achieved *through* the robot, i.e. generating motion to perform an action inducing a desirable modification, effectively *manipulating* the environment (manipulation). Motions like (2) may result in changes in the robot's physical location within its environment. Generally, modifications to a robot's location within its environment may be considered instances of the general *locomotion* problem, further specified as *wheeled* or *legged* locomotion based on whenever a robot makes use of wheels or leg(s) to move in the environment. Lastly, an increased level of dynamism in the robot-environment interactions can be obtained combining (1) and (2), thus designing systems capable to interact with *and* move within their environment. This category is problems is typically termed *whole-body control*, and is characterized by a typically much larger set of control variables compared to either locomotion or manipulation alone.

The traditional body of work developed since the very inception of robotics is increasingly more complemented by learning-based approaches. ML has indeed proven particularly transformative across the entire robotics stack, first empowering planning-based techniques with improved state estimation used for traditional planning (Tang et al., 2023) and then end-to-end replacing controllers, effectively yielding perception-to-action methods (Kober et al.). Work in producing robots capable of navigating a diverse set of terrains demonstrated the premise of both dynamics and learning-based approaches for locomotion (Griffin et al., 2017; Ji et al., 2023; Lee et al., 2020; Margolis et al., 2022), and recent works on whole-body control indicated the premise of learning-based approaches to generate rich motion on complex robots, including humanoids (Zhang et al., 2024; **?**). Manipulation has also been widely studied, particularly considering its relevance for many impactful applications ranging from high-risk applications for humans (Fujita et al., 2020; Alizadeh and Zhu, 2024; Fujita et al., 2020) to manufacturing (Sanneman et al., 2020). While explicit models have proven fundamental in achieving important milestones towards the development of modern robotics, recent works leveraging implicit models proved particularly promising in surpassing scalability and practical applicability challenges via learning (Kober et al.).

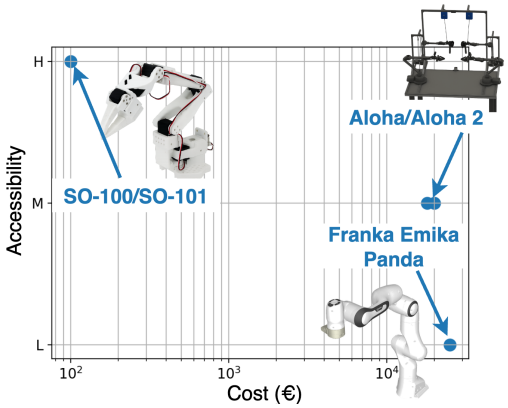


Figure 4 | Cheaper, more accessible robots are starting to rival traditional platforms like the Panda arm platforms in adoption in resource-constrained scenarios. The SO-100, in particular, has a cost in the 100s of Euros, and can be entirely 3D-printed in hours, while the industrially-manufactured Panda arm costs tens of thousands of Euros and is not openly available.

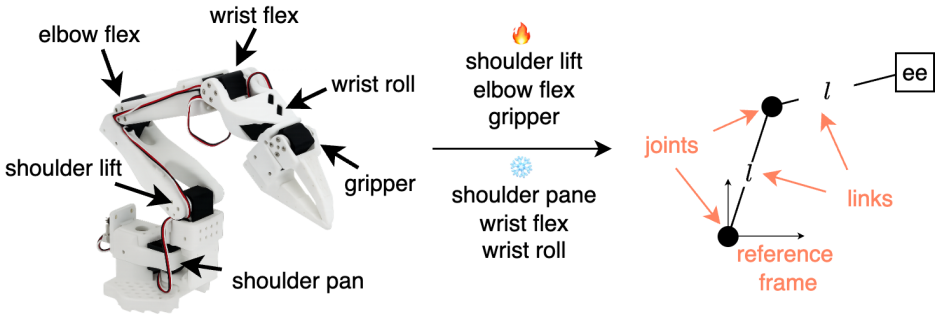


Figure 5 | The SO-100 arm is a 6-dof manipulator arm. Preventing some of its joints (shoulder pane, wrist flex and wrist roll) from actuating, it can be represented as a traditional 2-dof planar manipulator (the gripper joint in the end-effector is not considered towards the count of the degrees of freedom used to produce motion).

2.3 Example: Planar Manipulation

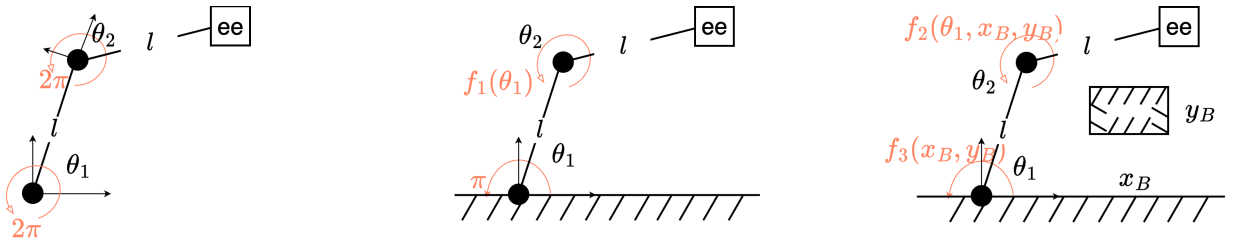
Robot manipulators typically consist of a series of links and joints, articulated in a chain finally connected to an *end-effector*. Links and joints are considered responsible for generating motion, while the end effector is instead used to perform specific actions at the target location (e.g., grasping/releasing objects via closing/opening a gripper end-effector, using a specialized tool like a screwdriver, etc.).

Recently, the development of low-cost manipulators like the Aloha (Zhao et al., 2023) Aloha-2 (Aldaco et al.) and SO-100/SO-101 (Knight et al.) platforms significantly lowered the barrier to entry to robotics, considering the increased accessibility of these robots compared to more traditional platforms like the Franka Emika Panda arm (Figure 4).

Deriving an intuition as per why learning-based approaches are gaining popularity in the robotics community requires briefly analyzing traditional approaches for manipulation, leveraging tools like forward and inverse kinematics (FK, IK) and control theory. Providing a detailed overview of these methods falls (well) out of the scope of this tutorial, and we refer the reader to works including Siciliano and Khatib (2016); Lynch and Park (2017); Tedrake (a,b) for a much more comprehensive description of these techniques. Here, we mostly wish to highlight the benefits of ML over these traditional techniques

Consider the (simple) case where a SO-100 is restrained from actuating (1) the shoulder pane and (2) the wrist flex and roll motors. This effectively reduces the degrees of freedom of the SO-100 from the original 5+1 (5 joints + 1 gripper) to 2+1 (shoulder lift, elbow flex + gripper). As the end-effector does not impact motion in this model, the SO-100 is effectively reduced to the planar manipulator robot presented in Figure 5, where spheres represent actuators, and solid lines indicate links from the base of the SO-100 to the end-effector (*ee*).

Further, let us make the simplifying assumption that actuators can produce rotations up to 2π radians. In practice,



(a) | Free to move

(b) | Constrained by the surface

(c) | Constrained by surface and (fixed) obstacle

Figure 6 | Planar, 2-dof schematic representation of the SO-100 manipulator under diverse deployment settings. From left to right: completely free of moving; constrained by the presence of the surface; constrained by the surface and presence of obstacles. Circular arrows around each joint indicate the maximal rotation feasible at that joint.

this is seldom the case due to movement obstructions caused by the robot body itself (for instance, the shoulder lift cannot produce counter-clockwise movement due to the presence of the robot's base used to secure the SO-100 to its support and host the robot bus), but we will introduce movement obstruction at a later stage.

All these simplifying assumptions leave us with the planar manipulator of Figure 6a, free of moving its end-effector by controlling the angles θ_1 and θ_2, jointly referred to as the robot's *configuration*, and indicated with $q = [\theta_1, \theta_2] \in [-\pi, +\pi]^2$. The axis attached to the joints indicate the associated reference frame, whereas circular arrows indicate the maximal feasible rotation allowed at each joint. In this tutorial, we do not cover topics related to spatial algebra, and we instead refer the reader to Lynch and Park (2017, Chapter 2) and Tedrake (a, Chapter 3) for excellent explanations of the mechanics and theoretical foundations of producing motion on rigid bodies.

Considering the (toy) example presented in Figure 6a, then we can analytically write the end-effector's position $p \in \mathbb{R}^2$ as a function of the robot's configuration, $p = p(q), p : \mathcal{Q} \mapsto \mathbb{R}^2$. In particular, we have:

$$p(q) = \begin{pmatrix} p_x(\theta_1, \theta_2) \\ p_y(\theta_1, \theta_2) \end{pmatrix} = \begin{pmatrix} l\cos(\theta_1) + l\cos(\theta_1 + \theta_2) \\ l\sin(\theta_1) + l\sin(\theta_1 + \theta_2) \end{pmatrix} \in S^{n=2}_{l_1+l_2} = \{p(q) \in \mathbb{R}^2 : \|p(q)\|_2^2 \leq (2l)^2, \ \forall q \in \mathcal{Q}\}$$

Deriving the end-effector's *pose* in some m-dimensional space $\boldsymbol{p} \in \mathcal{P} \subset \mathbb{R}^m$ (position and orientation) starting from the configuration $\mathrm{q} \in \mathcal{Q} \subset \mathbb{R}^n$ of a n-joints robot is referred to as *forward kinematics* (FK), whereas identifying the configuration corresponding to any given target pose is termed *inverse kinematics* (IK). In that, FK is used to map a robot configuration into the corresponding end-effector pose, whereas IK is used to reconstruct the configuration(s) given an end-effector pose.

In the simplified case here considered (for which $\boldsymbol{p} \equiv p$, as the orientation of the end-effector is disregarded for simplicity), one can solve the problem of controlling the end-effector's location to reach a goal position p^* by solving analytically for $q : p(q) = f_{\text{FK}}(q) = p^*$. However, in the general case, one might not be able to solve this problem analytically, and can typically resort to iterative optimization methods comparing candidate solutions using a loss function (in the simplest case, $\|p(q) - p^*\|_2^2$ is a natural candidate), yielding:

$$\min_{q \in \mathcal{Q}} \|p(q) - p^*\|_2^2 \tag{1}$$

Analytical solutions are even less appealing when one considers the presence of obstacles in the robot's workspace, resulting in constraints on the possible values of $q \in \mathcal{Q} \subseteq [-\pi, +\pi]^n \subset \mathbb{R}^n$ in the general case of n-links robots.

For instance, the robot in Figure 6b is (very naturally) obstacled by the presence of the surface upon which it rests: θ_1 can now exclusively vary within $[0, \pi]$, while possible variations in θ_2 depend on θ_1 (when $\theta_1 \to 0$ or $\theta_1 \to \pi$, further downwards movements are restricted). Even for a simplified kinematic model, developing techniques to solve 1 is in general non-trivial in the presence of constraints, particularly considering that the feasible set of solutions $\mathcal{Q}$ may change across problems. Figure 6c provides an example of how the environment influences the feasible set considered, with a new set of constraints deriving from the position of a new obstacle.

Further, IK—solving 1 for a feasible q—only proves useful in determining information regarding the robot's configuration in the goal pose, and crucially does not provide information on the *trajectory* to follow over time to reach a target pose.

Expert-defined trajectories obviate to this problem providing a length-K succession of goal poses $\tau_K = [p_0^*, p_1^*, \dots p_K^*]$ for tracking. In practice, trajectories can also be obtained automatically through *motion planning* algorithms, thus avoiding expensive trajectory definition from human experts. However, tracking τ_K via IK can prove prohibitively expensive, as tracking would require K resolutions of 1 (one for each target pose). *Differential* inverse kinematics (diff-IK) complements IK by enabling adaptive trajectory tracking. Let $J(q)$ denote the Jacobian matrix of (partial) derivatives of the FK-function $f_{\text{FK}} : \mathcal{Q} \mapsto \mathcal{P}$, such that $J(q) = \frac{\partial f_{FK}(q)}{\partial q}$. Then, one can apply the chain rule to any $p(q) = f_{\text{FK}}(q)$, deriving $\dot{p} = J(q)\dot{q}$, and thus finally relating variations in the robot configurations to variations in pose, thereby providing a platform for control.

Given a desired end-effector trajectory $\dot{p}^*(t)$ (1) indicating ancor regions in space and (2) how much time to spend in each region, diff-IK finds $\dot{q}(t)$ solving for joints' *velocities* instead of *configurations*,

$$\dot{q}(t) = \arg\min_{\nu} \|J(q(t))\nu - \dot{p}^*(t)\|_2^2 \tag{2}$$

Unlike 1, solving for $\dot{q}$ is much less dependent on the environment (typically, variations in velocity are constrained by physical limits on the actuators represented with box-like constraints in 2). Conveniently, 2 also often admits the closed-form solution $\dot{q} = J(q)^+\dot{p}^*$, where $J^+(q)$ denotes the Moore-Penrose pseudo-inverse of $J(q)$. Finally, discrete-time joint configurations q can be reconstructed from joint velocities $\dot{q}$ using forward-integration on the continuous-time joint velocity , $q_{t+1} = q_t + \Delta t\, \dot{q}_t$ for a given Δt, resulting in tracking via diff-IK.

Following trajectories with diff-IK is a valid option in well-controlled and static environments (e.g., industrial manipulators in controlled manufacturing settings), and relies on the ability to define a set of target velocities to track $[\dot{p}_0^*, \dot{p}_1^*, \dots, \dot{p}_k^*]$—an error-prone task largely carried out by human experts. Furthermore, diff-IK relies on the ability to (1) access $J(q)\,\forall q \in \mathcal{Q}$ and (2) compute its pseudo-inverse at every iteration of a given control cycle—a challenging assumption in highly dynamical settings, or for complex kinematic chains, or nonlinear dynamics.

2.3.1 Adding Feedback Loops

While very effective when a goal trajectory has been well specified, the performance of diff-IK can degrade significantly in the presence of modeling/tracking errors, or in the presence of non-modeled dynamics in the environment.

One such case is presented in Figure 7, where another rigid body other than the manipulator is moving in the environment along the horizontal axis, with velocity $\dot{x}_B$. Accounting analytically for the presence of this disturbance—for instance, to prevent the midpoint of l_1 from ever colliding with the object—requires access to $\dot{x}_B$ at least, to derive the equation characterizing the motion of the environment.

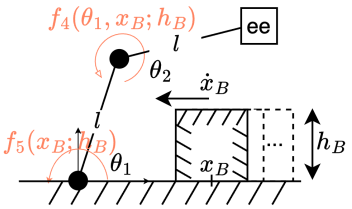


Figure 7 | Planar manipulator robot in the presence of a moving obstacle.

Less predictable disturbances however (e.g., $\dot{x}_B \leftarrow \dot{x}_B + \varepsilon, \varepsilon \sim N(0,1)$) may prove challenging to model analytically, and one could attain the same result of preventing link-object collision by adding a condition on the distance between the midpoint of l_1 and x_B, enforced through a feedback loop on the position of the robot and object at each control cycle.

To mitigate the effect of modeling errors, sensing noise and other disturbances, classical pipelines indeed do augment diff-IK with feedback control looping back quantities of interest. In practice, following a trajectory with a closed feedback loop might consist in backwarding the error between the target and measured pose, $\Delta p = p^* - p(q)$, hereby modifying the control applied to $\dot{q} = J(q)^+(\dot{p}^* + k_p \Delta p)$, with k_p defined as the (proportional) gain.

More advanced techniques for control consisting in feedback linearization, PID control, Linear Quatratic Regulator (LQR) or Model-Predictive Control (MPC) can be employed to stabilize tracking and reject moderate perturbations ((Siciliano and Khatib, 2016, Chapter 8) for in-detail explanation of these concepts, or (Tedrake, a, Chapter 8) for a simple, intuitive example in the case of a point-mass system). Nonetheless, feedback control presents its challenges as well: tuning gains remains laborious and system-specific. In particular, manipulation tasks present intermittent contacts inducing hybrid dynamics (mode switches) and discontinuities in the effective Jacobian, challenging the stability guarantees of the controller and thus often necessitating rather conservative gains and substantial hand-tuning.

We point the interested reader to Siciliano and Khatib (2016, Chapter 2,7,8), Lynch and Park (2017, Chapter 6,11), and Tedrake (a, Chapter 3,8) for extended coverage of FK, IK, diff-IK and control for (diff-)IK.

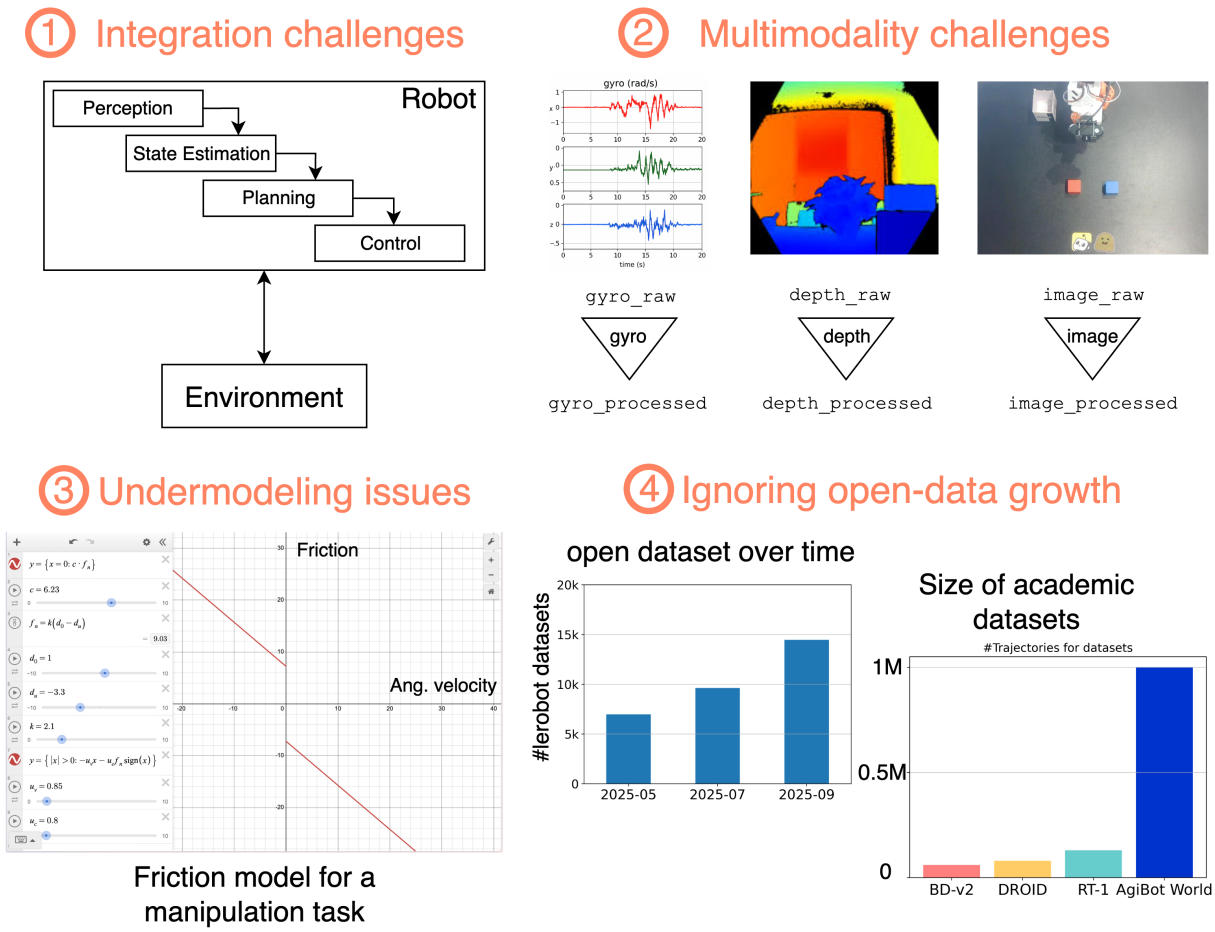


Figure 8 | Dynamics-based approaches to robotics suffer from several limitations: (1) orchestrating multiple components poses integration challenges; (2) the need to develop custom processing pipelines for the sensing modalities and tasks considered hinders scalability; (3) simplified analytical models of physical phenomena (here friction at the gripper; credits to Antonova et al. (2017)) limit real-world performance. Lastly, (4) dynamics-based methods overlook trends in the availability and growth of robotics data.

2.4 Limitations of Dynamics-based Robotics

Despite the last 60+ years of robotics research, autonomous robots are still largely incapable of performing tasks at human-level performance in the physical world generalizing across (1) robot embodiments (different manipulators, different locomotion platforms, etc.) and (2) tasks (tying shoe-laces, manipulating a diverse set of objects). While essential in the early development of robotics, the aforementioned methods require significant human expertise to be used in practice, and are typically specific to a particular applicative problem.

Dynamics-based robotics pipelines have historically been developed sequentially, engineering the different blocks now within most architectures for specific purposes. That is, sensing, state estimation, mapping, planning, (diff-)IK, and low-level control have been traditionally developed as distinct modules with fixed interfaces. Pipelining these specific modules proved error-prone, and brittleness emerges—alongside error compounding—whenever changes incur (e.g., changes in lighting for sensing, occlusion/failure of sensors, control failures). Adapting such a stack to new tasks or robotic platforms often entails re-specifying objectives, constraints, and heuristics at multiple stages, incurring significant engineering overhead.

Moreover, classical planners operate on compact, assumed-sufficient state representations; extending them to reason directly over raw, heterogeneous and noisy data streams is non-trivial. This results in a limited scalability to multimodal data and multitask settings, as incorporating high-dimensional perceptual inputs (RGB, depth, tactile, audio) traditionally required extensive engineering efforts to extract meaningful features for control. Also, the large number of tasks, coupled with the adoption of *per-task* planners, goal parameterizations, and safety constraints, results in an explosion in design and validation options, with little opportunity to reuse solutions across tasks.

Setting aside integration and scalability challenges: developing accurate modeling of contact, friction, and compliance for complicated remains difficult. Rigid-body approximations are often insufficient in the presence of deformable objects, and relying on approximated models hinders real-world applicability of the methods developed. In the case of complex, time-dependent and/or non-linear dynamics, even moderate mismatches in parameters, unmodeled evolutions, or grasp-induced couplings can qualitatively affect the observed dynamics.

Lastly, dynamics-based methods (naturally) overlook the rather recent increase in availability of openly-available robotics datasets. The curation of academic datasets by large centralized groups of human experts in robotics (Ope; **?**; AgiBot-World-Contributors et al., 2025) is now increasingly complemented by a growing number of robotics datasets contributed in a decentralized fashion by individuals with varied expertises. If not tangentially, dynamics-based approaches are not posed to maximally benefit from this trend, which holds the premise of allowing generalization in the space of tasks and embodiments, like data was the cornerstone for advancements in vision (Alayrac et al., 2022) and natural-language understanding (Brown et al., 2020).

Taken together, these limitations (Figure 8) motivate the exploration of learning-based approaches that can (1) integrate perception and control more tightly, (2) adapt across tasks and embodiments with reduced expert modeling interventions and (3) scale gracefully in performance as more robotics data becomes available.

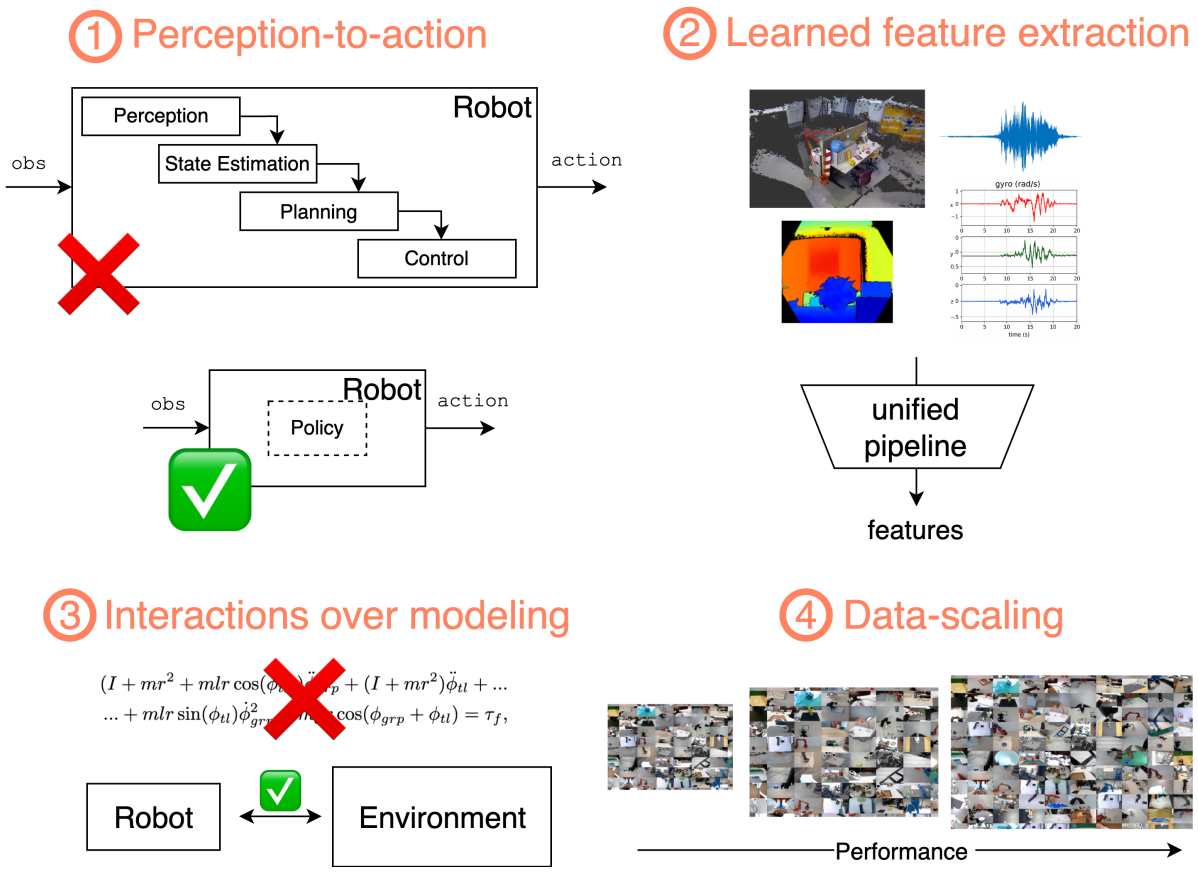


Figure 9 | Learning-based robotics streamlines perception-to-action by learning a (1) unified high-level controller capable to take (2) high-dimensional, unstructured sensorimotor information. Learning (3) does not require a dynamics model and instead focuses on interaction data, and (4) empirically correlates with the scale of the data used.

3 Robot (Reinforcement) Learning

Approximate the solution, not the problem [...]

Richard Sutton

TL;DR

The need for expensive high-fidelity simulators can be obviated by learning from real-world data, using sample-efficient algorithms that can safely train directly on hardware.

Learning-based techniques for robotics naturally address the limitations presented in 2 (Figure 9). Learning-based techniques typically rely on prediction-to-action (*visuomotor policies*), thereby directly mapping sensorimotor inputs to predicted actions, streamlining control policies by removing the need to interface multiple components. Mapping sensorimotor inputs to actions directly also allows to add diverse input modalities, leveraging the automatic feature extraction characteristic of most modern 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 even entirely unknown. Lastly, 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, in great part overlooked by dynamics-based approaches.

Being a field at its relative nascent stages, no prevalent technique(s) proved distinctly better better in robot learning. Still, two major classes of methods gained prominence: reinforcement learning (RL) and Behavioral Cloning (BC)

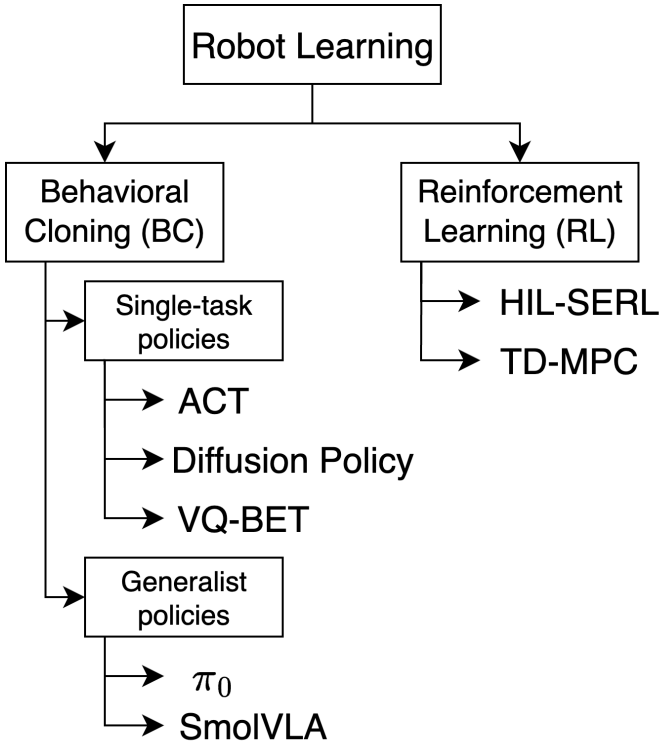


Figure 10 | Overview of the robot learning methods implemented in `lerobot`.

(Figure 10). 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, to introduce BC techniques in the next sections (**??**).

In Figure 10 we decided to include generalist robot models (Black et al., 2024; Shukor et al., 2025) alongside task-specific BC methods. While significant different in spirit—*generalist* models are language-conditioned and use instructions to generate motion valid across many tasks, while *task-specific* models are typically not language-conditioned and used to perform a single task—foundation models are largely trained to reproduce trajectories contained in a large training set of input demonstrations. Thus, we argue generalist policies can indeed be grouped alongside other task-specific BC methods, as they both leverage similar training data and schemas.

Figure 10 illustrates this categorization 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).

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 11 depicts two of such cases. Reaching for an object to move somewhere else in the scene is an indeed sequential problem where at each cycle the controller needs to adjust the position of the robotic arm based on their current configuration and the (possibly varying) position of the object. Figure 11 also shows an example of a locomotion problem, where sequentiality is inherent in the problem formulation. While sliding to the side, the controller has to constantly keeps adjusting to the robot's propioperception to avoid failure (falling).

3.1 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 (ideally, well-performing) control strategies (*policies*). Crucially for robotics, RL agents can improve via trial-and-error only, thus entirely bypassing the need to develop explicit models of the problem dynamics, and rather exploiting interaction data only.

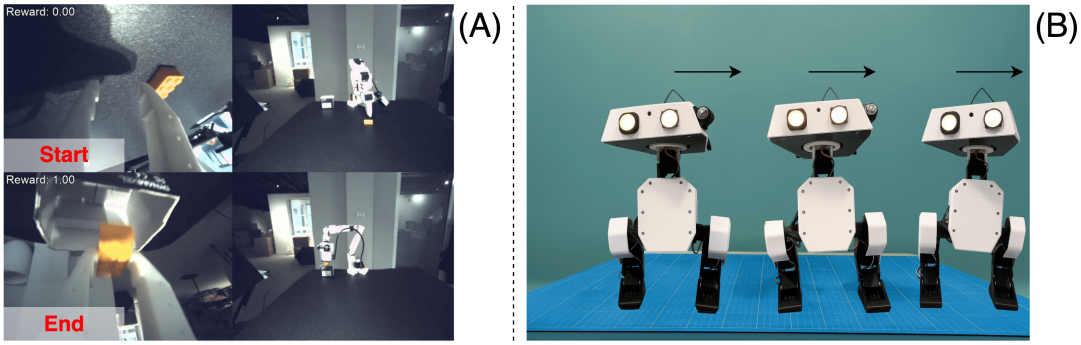


Figure 11 | Examples of two different robotics tasks performed using RL. In the manipulation task (A) an agent learns to reach for a yellow plastic block in its environment, and to put it inside of a box. In the locomotion task (B) an agent learns to move its center of mass sideways without falling.

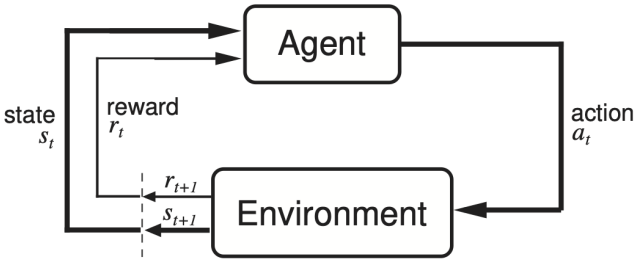


Figure 12 | Agent-Environment interaction diagram (image credits to Sutton and Barto (2018)).

In RL, this feedback loop (Figure 12) between actions and outcomes is established through the agent sensing a scalar quantity (*reward*).

Formally, interactions between an agent and its environment are typically modeled via a Markov Decision Process (MDP) (Bellman, 1957). Representing robotics problems via MDPs offers several advantages, including (1) incorporating uncertainty through MDP's inherently stochastic formulation 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, thus assuming 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 finite-horizon (*episodic*) MDPs here.

Formally, a lenght-T Markov Decision Process (MDP) is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{D}, r, \gamma, \rho, T \rangle$, 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 $\mathcal{D}(s_t, a_t, s_{t+1}) = \mathbb{P}(s_{t+1} | s_t, a_t)$. For instance, for a planar manipulator dynamics could be considered deterministic when the environment is fully described (Figure 6a), and stochastic when unmodeled disturbances depending on non-observable parameters intervene (Figure 7).
- $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the *reward function*, weighing the transition (s_t, a_t, s_{t+1}) in the context of the achievement of an arbitrary goal. For instance, a simple reward function for quickly moving the along the x axis in 3D-space (Figure 11) could be based on the absolute 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, $r(s_t, a_t, s_{t+1}) \equiv r(s_t) = p_{x_t} \cdot \dot{p}_{x_t} - \frac{1}{p_{z_t}}$.

Lastly, $\gamma \in [0, 1]$ represent the discount factor regulating preference for immediate versus long-term reward (with an effective horizon equal to $\frac{1}{1-\gamma}$), and ρ is the distribution, defined over $\mathcal{S}$, the MDP's *initial* state is sampled from,

$s_0 \sim \rho$.

A length-T *trajectory* is the (random) sequence

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_{T-1}, a_{T-1}, r_{T-1}, s_T), \tag{3}$$

with per-step rewards defined as $r_t = r(s_t, a_t, s_{t+1})$ for ease of notation.Interestingly, assuming both the environment dynamics and conditional distribution over actions given states—the *policy*—to be *Markovian*:

$$\mathbb{P}(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots s_0, a_0) = \mathbb{P}(s_{t+1}|s_t, a_t) \tag{4}$$

$$\mathbb{P}(a_t|s_t, a_{t-1}, s_{t-1}, s_0, a_0) = \mathbb{P}(a_t|s_t) \tag{5}$$

The probability of observing a given trajectory τ factorizes into

$$\mathbb{P}(\tau) = \mathbb{P}(s_0) \prod_{t=0}^{T-1} \mathbb{P}(s_{t+1}|s_t, a_t)\ \mathbb{P}(a_t|s_t). \tag{6}$$

Policies $\mathbb{P}(a_t|s_t)$ are typically indicated as $\pi(a_t|s_t)$, and often parametrized via θ, yielding $\pi_\theta(a_t|s_t)$. Policies are trained optimizing the (discounted) *return* associated to a given τ, i.e. the (random) sum of measured rewards over trajectory:

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

In that, agents seek to learn control strategies (*policies*, π_θ) maximizing the expected return $\mathbb{E}_{\tau\sim\pi_\theta} G(\tau)$. 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 in the space of all the possible policies Π, yielding the (maximization) target $J : \Pi \mapsto \mathbb{R}$

$$J(\pi_\theta) = \mathbb{E}_{\tau\sim\mathbb{P}_{\theta;\mathcal{D}}}[G(\tau)], \tag{7}$$

$$\mathbb{P}_{\theta;\mathcal{D}}(\tau) = \rho \prod_{t=0}^{T-1} \mathcal{D}(s_t, a_t, s_{t+1})\ \pi_\theta(a_t|s_t). \tag{8}$$

Because in the RL framework the agent is assumed to only be able to observe the environment dynamics and not to intervene on them, 7 varies exclusively with the policy followed. In turn, MDPs 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* $\pi^* = \arg\max_\theta J(\pi_\theta)$, where θ is the parametrization adopted by the policy set $\Pi : \pi_\theta \in \Pi,\ \forall\theta$. Other than providing a target for policy search, $G(\tau)$ can also be used as a target to discriminate between states and state-action pairs. Given any state $s \in \mathcal{S}$—e.g., a given configuration of the robot—the *state-value* function

$$V_\pi(s) = \mathbb{E}_{\tau\sim\pi}\big[G(\tau)\big|s_0 = s\big]$$

can be used to discriminate between desirable and undesirable state in terms of long-term (discounted) reward maximization, under a given policy π. Similarily, the *state-action* value function also conditions the cumulative discounted reward on selecting action a when in s, and thereafter act according to π:

$$Q_\pi(s, a) = \mathbb{E}_{\tau\sim\pi}\big[G(\tau)\big|s_0 = s, a_0 = a\big]$$

Crucially, value functions are interrelated:

$$Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}\sim\mathbb{P}(\bullet|s_t,a_t)}[r_t + \gamma V_\pi(s_{t+1})] \tag{9}$$

$$V_\pi(s_t) = \mathbb{E}_{a_t\sim\pi(\bullet|s_t)}[Q_\pi(s_t, a_t)] \tag{10}$$

Inducing an ordering over states and state-action pairs under π, value functions are central to most RL algorithms. A variety of methods have been developed in RL as standalone attemps to find (approximate) solutions to the problem of maximizing cumulative reward (Figure 13).

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 (Akkaya et al., 2019) and locomotion (Lee et al., 2020) problems, RL proved extremely effective in providing a platform to (1) adopt a unified, streamlined perception-to-action pipeline, (2) natively integrate propioperception with multi-modal high-dimensional sensor streams (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 and stored in datasets. For a more complete survey of applications of RL to robotics, we refer the reader to Kober et al.; Tang et al. (2024).

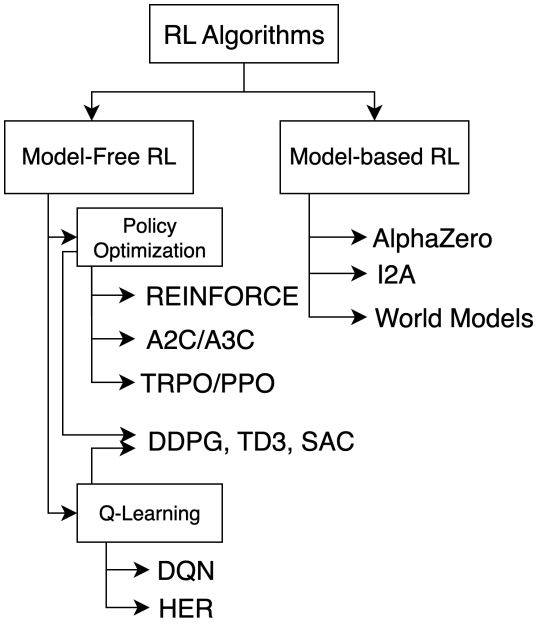


Figure 13 | Popular RL algorithms. See Achiam (2018) for a complete list of citations.

3.2 Real-world RL for Robotics

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

First, especially early in training, actions are typically explorative, and thus erractic. On physical systems, untrained policies may command high velocities, self-collisiding configurations, or torques exceeding joint limits, leading to wear and potential hardware damage. Mitigating these risks requires external safeguards (e.g., watchdogs, safety monitors, emergency stops), often incuring in a high degree of human supervision. Further, in the typical episodic setting considered in most robotics problems, experimentation is substantially slowed down by the need to manually reset the environment over the course of training, a time-consuming and brittle process.

Second, learning with a limited number of samples remains problematic in RL, limiting the applicability of RL in real-world robotics due to consequently prohibitive timescales of training. Even strong algorithms such as SAC (Haarnoja et al., 2018) typically require a large numbers of transitions $\{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{N}$. On hardware, generating these data is time-consuming and can even be prohibitive.

Training RL policies in simulation (Tobin et al., 2017) addresses both issues: it eliminates physical risk and dramatically increases throughput. Yet, simulators require significant modeling effort, and rely on assumptions (simplified physical modeling, instantaneous actuation, static environmental conditions, etc.) limiting transferring policies learned in simulation due the discrepancy between real and simulated environments (*reality gap*, Figure 14). *Domain randomization* (DR) is a popular technique to overcome the reality gap, consisting in randomizing parameters of the simulated environment during training, to induce robustness to specific disturbances. In turn, DR is employed to increase the diversity of scenarios over the course of training, improving on the chances sim-to-real transfer (Akkaya et al., 2019; Antonova et al., 2017; Ji et al., 2023). In practice, DR is performed further parametrizing the *simulator*'s dynamics $\mathcal{D} \equiv \mathcal{D}_\xi$ with a *dynamics* (random) vector ξ drawn an arbitrary distribution, $\xi \sim \Xi$. Over the course of training—typically at each episode's reset—a new ξ is drawn, and used to specify the environment's dynamics for that episode. For instance, one could decide to randomize the friction coefficient of the surface in a locomotion task (Figure 15), or the center of mass of an object for a manipulation task.

While effective in transfering policies across the reality gap in real-world robotics (Tobin et al., 2017; Akkaya et al., 2019; Ji et al., 2023; Tiboni et al., 2024), DR often requires extensive manual engineering. First, identifying which parameters to randomize—i.e., the *support* supp(Ξ) of Ξ—is an inherently task specific process. When locomoting over different terrains, choosing to randomize the friction coefficient is a reasonable choice, yet not completely resolutive as other factors (lightning conditions, external temperature, joints' fatigue, etc.) may prove just as important, making selecting these parameters yet another source of brittlness.

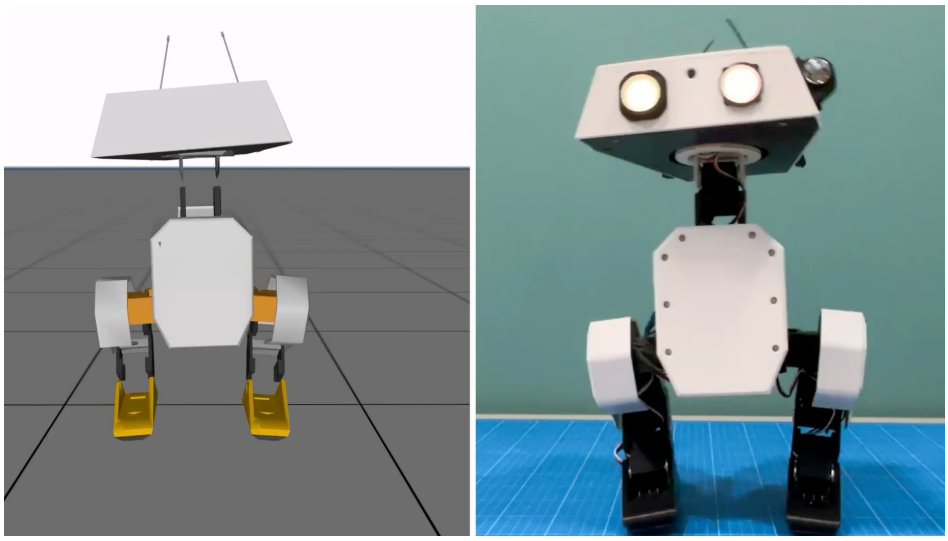


Figure 14 | Simulated (left) vs. real-world (right) OpenDuck. Discrepancies in the simulation dynamics (*reality gap*) pose risks to policy transfer.

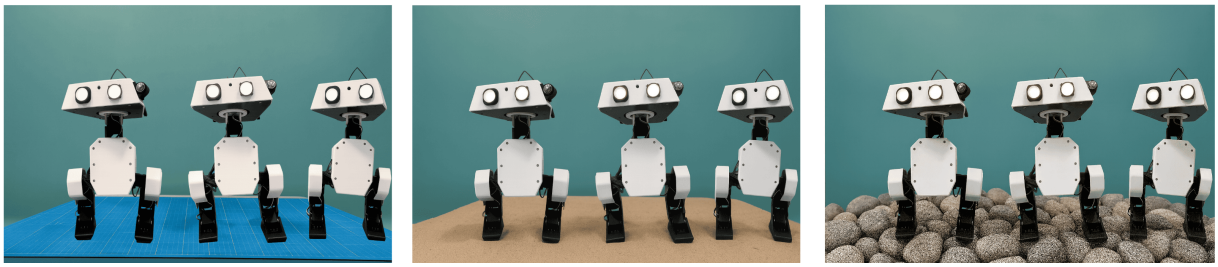


Figure 15 | The same locomotion task can be carried out in different (simulated) domains (exemplified by the difference in terrains) at training time, resulting to increased robustness over diverse environment dynamics.

Selecting the dynamics distribution Ξ is also non-trivial. On the one hand, distributions with low entropy might risk to cause failure at transfer time, due to the limited robustness induced over the course of training. On the other hand, excessive randomization may cause over-regularization and hinder performance. Consequently, the research community investigated approaches to automatically select the randomization distribution Ξ, using signals from the training process or tuning it to reproduce observed real-world trajectories. Akkaya et al. (2019) use a parametric uniform distribution $\mathcal{U}(a, b)$ as Ξ, widening the bounds as training progresses and the agent's performance improves (AutoDR). While effective, AutoDR requires significant tuning—the bounds are widened by a fixed, pre-specified amount Δ—and may disregard data when performance *does not* improve after a distribution update (Tiboni et al., 2024). Tiboni et al. (2024) propose a similar method to AutoDR (DORAEMON) to evolve Ξ based on training signal, but with the key difference of explicitly maximizing the entropy of parametric Beta distributions, inherently more flexible than uniform distributions. DORAEMON proves particularly effective at dynamically increasing the entropy levels of the training distribution by employing a max-entropy objective, under performance constraints formulation. Other approaches to automatic DR consist in specifically tuning Ξ to align as much as possible the simulation and real-world domains. For instance, Chebotar et al. (2019) interleave in-simulation policy training with repeated real-world policy rollouts used to adjust Ξ based on real-world data, while Tiboni et al. (2023) leverage a single, pre-collected set of real-world trajectories and tune Ξ under a simple likelihood objective.

While DR has shown promise, it does not address the main limitation that, even under the assumption that an ideal distribution Ξ to sample from was indeed available, many robotics problems cannot be simulated with high-enough fidelity under practical computational constraints in the first place. Simulating contact-rich manipulation of possibly deformable or soft materials—i.e., *folding a piece of clothing*—can be costly and even time-intensive, limiting the benefits of in-simulation training.

A perhaps more foundamental limitation of RL for robotics is the general unavailability of complicated tasks' *dense* reward function, the design of which is essentially based on human expertise and trial-and-error. In practice, *sparse* reward functions can be used to conclude whether one specific goal has been attained—*has this t-shirt been correctly folded?*—but unfortunately incur in more challenging learning. As a result, despite notable successes, deploying RL directly on real-world robots at scale remains challenging.

To make the most of (1) the growing number of openly available datasets and (2) relatively inexpensive robots like the SO-100, RL could (1) be anchored in already-collected trajectories—limiting erratic and dangerous exploration—and (2) train in the real-world directly—bypassing the aforementioned issues with low-fidelity simulations. In such a context, sample-efficient learning is also paramount, as training on the real-world is inherently time-bottlenecked.

Off-policy algorithms like Soft Actor-Critic (SAC) (Haarnoja et al., 2018) tend to be more sample efficient then their on-policy counterpart (Schulman et al., 2017b), due to the presence a *replay buffer* used over the course of the training. Other than allowing to re-use transitions (s_t, a_t, r_t, s_{t+1}) over the course of training, the replay buffer can also accomodate for the injection of previously-collected data in the training process (Ball et al., 2023). Using expert demonstrations to guide learning together with learned rewards, RL training can effectively be carried out in the real-world (Luo et al., 2025). Interestingly, when completed with in-training human interventions, real-world RL agents have been shown to learn policies with near-perfect success rates on challenging manipulation tasks in 1-2 hours (Luo et al., 2024).

Sample-efficient RL In an MDP, the optimal policy π^* can be derived from its associated Q-function, Q_{π^*}, and in particular the optimal action(s) $\mu(s_t)$ can be selected maximizing the optimal Q-function over the action space,

$$\mu(s_t) = \max_{a_t \in \mathcal{A}} Q_{\pi^*}(s_t, a_t).$$

Interestingly, the Q^*-function satisfies a recursive relationship (*Bellman equation*) based on a very natural intuition[1]:

> [...] If the optimal value $Q^*(s_{t+1}, a_{t+1})$ of the [state] s_{t+1} was known for all possible actions a_{t+1}, then the optimal strategy is to select the action a_{t+1} maximizing the expected value of $r_t + \gamma Q^*(s_{t+1}, a_{t+1})$

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim \mathbb{P}(\bullet | s_t, a_t)} \left[r_t + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^*(s_{t+1}, a_{t+1}) \middle| s_t, a_t \right]$$

In turn, the optimal Q-function is guaranteed to be self-consistent by definition. *Value-iteration* methods exploit this relationship (and/or its state-value counterpart, $V^*(s_t)$) by iteratively updating an initial estimate of Q^*, Q_k using

[1] Quote from Mnih et al. (2013). The notation used has slightly been adapted for consistency with the rest of this tutorial.

the Bellman equation as update rule (*Q-learning*):

$$Q_{i+1}(s_t, a_t) \leftarrow \mathbb{E}_{s_{t+1}\sim\mathbb{P}(\bullet|s_t,a_t)}\left[r_t + \gamma \max_{a_{t+1}\in\mathcal{A}} Q_i(s_{t+1}, a_{t+1})\middle|s_t, a_t\right], \quad i = 0, 1, 2, \dots, K$$

Then, one can derive the (ideally, near-optimal) policy by explicitly maximizing over the action space the final (ideally, near-optimal) estimate $Q_K \approx Q^*$ at each timestep. In fact, under certain assumptions on the MDP considered, $Q_K \to Q^*$ as $K \to \infty$.

Effective in its early applications to small-scale discrete problems and theoretically sound, vanilla Q-learning was found complicated to scale to large $\mathcal{S} \times \mathcal{A}$ problems, in which the storing of $Q : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ alone might result prohibitive. Also, vanilla Q-learning is not directly usable for *continuous*, unstructured state-action space MPDs, such as those considered in robotics. In their seminal work on *Deep Q-Learning* (DQN), Mnih et al. (2013) propose learning Q-values using deep convolutional neural networks, thereby accomodating for large and even unstructured *state* spaces. DQN parametrizes the Q-function using a neural network with parameters θ, updating the parameters by sequentially minimizing the expected squared temporal-difference error (TD-error, δ_i):

$$\mathcal{L}(\theta_i) = \mathbb{E}_{(s_t,a_t)\sim\chi(\bullet)}\big[\underbrace{(y_i - Q_{\theta_i}(s_t, a_t)}_{\delta_i})^2\big], \tag{11}$$

$$y_i = \mathbb{E}_{s_{t+1}\sim\mathbb{P}(\bullet|s_t,a_t)}\big[r_t + \gamma \max_{a_t\in\mathcal{A}} Q_{\theta_{i-1}}(s_{t+1}, a_{t+1})\big], \tag{12}$$

Where χ represents a behavior distribution over state-action pairs. Crucially, χ can in principle be different from the policy being followed, effectively allowing to reuse prior data stored in a *replay buffer* in the form of (s_t, a_t, r_t, s_{t+1}) transitions, used to form the TD-target y_i, TD-error δ_i and loss function 11 via Monte-Carlo (MC) estimates.

While effective in handling large, unstructured state spaces for discrete action-space problems, DQN application's to continous control problems proved challenging. Indeed, in the case of high-capacity function approximators such as neural networks, solving $\max_{a_t\in\mathcal{A}} Q_\theta(s_t, a_t)$ at each timestep is simply unfeasible due to the (1) continous nature of the action space ($\mathcal{A} \subset \mathbb{R}^n$ for some n) and (2) impossibility to express the find a cheap (ideally, closed-form) solution to Q_θ. Silver et al. (2014) tackle this fundamental challenge by using a *deterministic* function of the state s_t as policy, $\mu_\phi(s_t) = a_t$, parametrized by ϕ. Thus, policies can be iteratively refined updating ϕ along the direction:

$$d_\phi = \mathbb{E}_{s_t\sim\mathbb{P}(\bullet)}\big[\nabla_\phi Q(s_t, a_t)|_{a_t=\mu_\phi(s_t)}\big] = \mathbb{E}_{s_t\sim\mathbb{P}(\bullet)}\big[\nabla_{a_t} Q(s_t, a_t)|_{a_t=\mu_\phi(s_t)} \cdot \nabla_\phi \mu(s_t)\big] \tag{13}$$

Provably, 13 is the *deterministic policy gradient* (DPG) of the policy μ_ϕ (Silver et al., 2014), so that updates $\phi_{k+1} \leftarrow \phi_k + \alpha d_\phi$ are guaranteed to increase the (deterministic) cumulative discounted reward, $J(\mu_\phi)$. Lillicrap et al. (2019) extended DPG to the case of (1) high-dimensional unstructured observations and (2) continuous action spaces, introducing Deep Deterministic Policy Gradient (DDPG), an important algorithm RL and its applications to robotics. DDPG adopts a modified TD-target compared to the one defined in 12, by maintaining a policy network used to select actions, yielding

$$y_i = \mathbb{E}_{s_{t+1}\sim\mathbb{P}(\bullet|s_t,a_t)}\big[r_t + \gamma Q_{\theta_{i-1}}(s_{t+1}, \mu_\phi(s_{t+1}))\big]. \tag{14}$$

Similarily to DQN, DDPG also employs the same replay buffer mechanism, to reuse past transitions over training for increased sample efficiency and estimate the loss function via MC-estimates.

Soft Actor-Critic (SAC) (Haarnoja et al., 2018) is a derivation of DDPG in the max-entropy (MaxEnt) RL framework, in which RL agents are tasked with maximizing the discounted cumulative reward, while acting as randomly as possible. MaxEnt RL (Haarnoja et al., 2017) has proven particularly robust thanks to the development of diverse behaviors, incentivized by its entropy-regularization formulation. In that, MaxEnt revisits the RL objective $J(\pi)$ to specifically account for the policy entropy,

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\chi}[r_t + \alpha\mathcal{H}(\pi(\bullet|s_t))] \tag{15}$$

This modified objective results in the *soft* TD-target:

$$y_i = \mathbb{E}_{s_{t+1}\sim\mathbb{P}(\bullet|s_t,a_t)}\big[r_t + \gamma\big(Q_{\theta_{i-1}}(s_{t+1}, a_{t+1}) - \alpha \log \pi_\phi(a_{t+1}|s_{t+1})\big)\big], \quad a_{t+1} \sim \pi_\phi(\bullet|s_t) \tag{16}$$

Similarily to DDPG, SAC also maintains an explicit policy, trained under the same MaxEnt framework for the maximization of 15, and updated using:

$$\pi_{k+1} \leftarrow \arg\min_{\pi'\in\Pi} \mathrm{D}_{\mathrm{KL}}\left(\pi'(\bullet|s_t)\middle\|\frac{\exp(Q_{\pi_k}(s_t, \bullet))}{Z_{\pi_k}(s_t)}\right) \tag{17}$$

The update rule provided in 17 optimizes the policy while projecting it on a set Π of tractable distributions (e.g., Gaussians, Haarnoja et al. (2017)).

Sample-efficient, data-driven RL Importantly, sampling (s_t, a_t, r_t, s_{t+1}) from the replay buffer D conveniently allows to approximate the previously introduced expectations for TD-target and TD-error through Monte-Carlo (MC) estimates. The replay buffer D also proves extremely useful in maintaining a history of previous transitions and using it for training, improving on sample efficiency. Furthermore, it also naturally provides an entry point to inject offline trajectories recorded, for instance, by a human demonstrator, into the training process.

Reinforcement Learning with Prior Data (RLPD) (Ball et al., 2023) is an Offline-to-Online RL algorithm leveraging prior data to effectively accelerate the training of a SAC agent. Unlike previous works on Offline-to-Online RL, RLPD avoids any pre-training and instead uses the available offline data D_{offline} to improve online-learning from scratch. During each training step, transitions from both the offline and online replay buffers are sampled in equal proportion, and used in the underlying SAC routine.

Sample-efficient, data-driven, real-world RL Despite the possibility to leverage offline data for learning, the effectiveness of real-world RL training is still limited by the need to define a task-specific, hard-to-define reward function. Further, even assuming to have access to a well-defined reward function, typical robotics pipelines rely mostly on propioperceptive inputs augmented by camera streams of the environment. As such, even well-defined rewards would need to be derived from processed representations of unstructured observations, introducing brittleness. In their technical report, Luo et al. (2025) empirically address the needs (1) to define a reward function and (2) to use it on image observations, by introducing a series of tools to allow for streamlined training of *reward classifiers* c, as well as jointly learn forward-backward controllers to speed up real-world RL. Reward classifiers are particularly useful in treating complex tasks—e.g., folding a t-shirt—for which a precise reward formulation is arbitrarily complex to obtain, or that do require significant shaping and are more easily learned directly from demonstrations of success (e^+) or failure (e^-) states, $s \in \mathcal{S}$, with a natural choice for the state-conditioned reward function being $r\mathcal{S} \mapsto \mathbb{R}$ being $r(s) = \log c(e^+ \; verts)$. Further, Luo et al. (2025) demonstrate the benefits of learning *forward* (executing the task from initial state to completion) and *backward* (resetting the environment to the initial state from completion) controllers, parametrized by separate policies.

Lastly, in order to improve on the robustness of their approach to different goals while maintaing practical scalability, Luo et al. (2025) introduced a modified state and action space, expressing proprioperceptive configurations q and actions $\dot{q}$ in the frame of end-effector pose at $t = 0$. Randomizing the initial pose of the end-effector (s_0),Luo et al. (2025) achieved a similar result to that of having to manually randomize the environment at every timestep, but with the benefit of maintaining the environment in the same condition across multiple training episodes, achieving higher scalability of their method thanks to the increased practicality of their approach.

Building on off-policy deep Q-learning with replay buffers, entropy regularization for better exploration and performance, expert demonstrations to guide learning, and a series of tools and recommendations for real-world training using reward classifiers (Figure 16), Luo et al. (2024) introduce human interactions during training, learning near-optimal policies in challenging real-world manipulation tasks in 1-2 hours.

Human in the Loop Sample Efficient Robot reinforcement Learning (HIL-SERL) (Luo et al., 2024) augments offline-to-online RL with targeted human corrections during training, and employs prior data to (1) train a reward classifier and (2) bootstrap RL training on expert trajectories. While demonstrations provide the initial dataset seeding learning and constraining early exploration, interactive corrections allow a human supervisor to intervene on failure modes and supply targeted interventions to aid the learning process. Crucially, human interventions are stored in both the offline and online replay buffers, differently from the autonomous transitions generated at training time and stored in the online buffer only. Consequently, given an intervention timestep $k \in (0, T)$, length-K human intervention data $\{s_k^{\text{human}}, a_k^{\text{human}}, r_k^{\text{human}}, s_{k+1}^{\text{human}}, \}_{k=1}^{K}$ is more likely to be sampled for off-policy learning than the data generated online during training, providing stronger supervision to the agent while still allowing for autonomous learning. Empirically, HIL-SERL attains near-perfect success rates on diverse manipulation tasks within 1-2 hours of training (Luo et al., 2024), underscoring how offline datasets with online RL can markedly improve stability and data efficiency, and ultimately even allow real-world RL-training.

3.2.1 Code Example: Real-world RL

TODO(fracapuano): work out rl training example

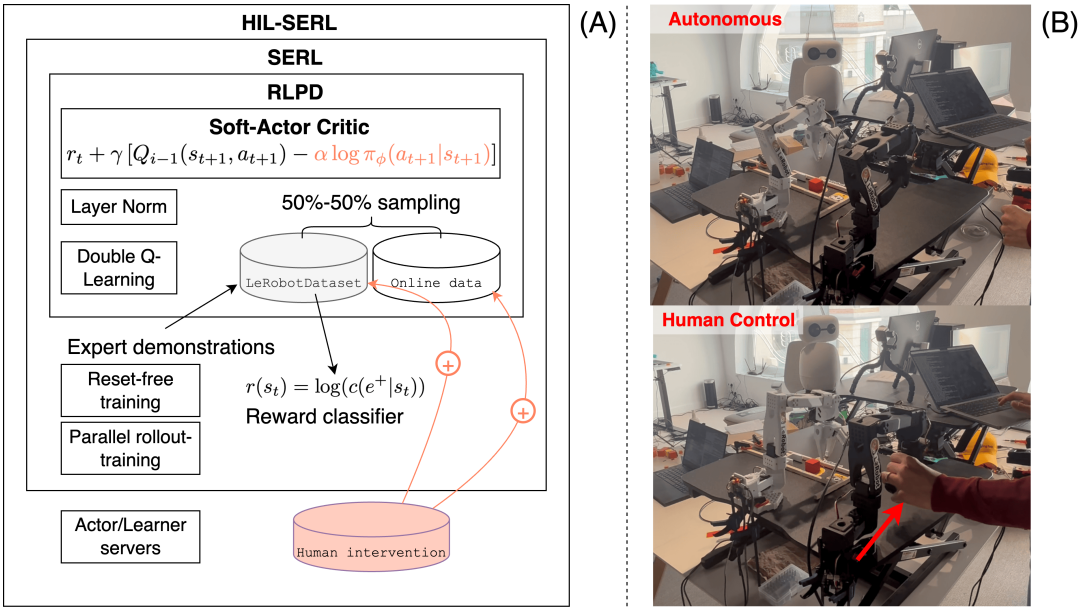


Figure 16 | (A) HIL-SERL allows for real-world training of high performance RL agents by building on top advancements presented by of SAC, RLPD and SERL. (B) Example of human intervention during a HIL-SERL training process on a SO-100.

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

Despite the advancements in real-world RL training, solving robotics training RL agents in the real world still suffers from the following limitations:

- In those instances where real-world training experience is prohibitively expensive to gather (Degrave et al., 2022; Bellemare et al., 2020), in-simulation training is often the only option. However, high-fidelity simulators for real-world problems can be difficult to build and maintain, especially for contact-rich manipulation and tasks involving deformable or soft materials.
- Reward design poses an additional source of brittleness. Dense shaping terms are often required to guide exploration in long-horizon problems, but poorly tuned terms can lead to specification gaming or local optima. Sparse rewards avoid shaping but exacerbate credit assignment and slow down learning. In practice, complex behaviors require efforts shaping rewards: a britlle and error prone process.

Advances in Behavioral Cloning (BC) from corpora of human demonstrations address both of these concerns. By learning in a supervised fashion to reproduce expert demonstrations, BC methods prove competitive while bypassing the need for simulated environments and hard-to-define reward functions.

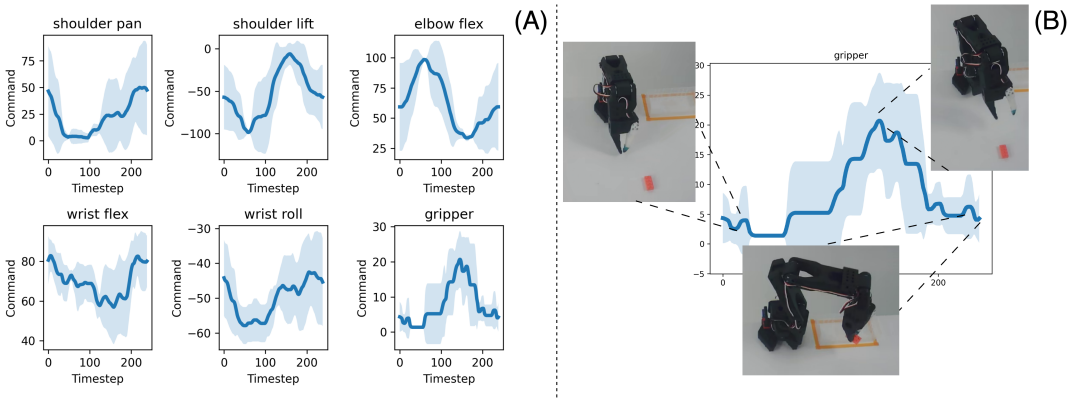


Figure 17 | (A) Average (with standard deviation) evolution of the actuation levels over the first 5 recorded episodes in `lerobot/svla_so101_pickplace`. Proprioperceptive state provide invaluable to determine the robot's state during an episode. (B) Camera frames are also recorded alongside measurements on the robot's state, capturing information about the robot's interaction with its environment.

4 Robot (Imitation) Learning

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

Norbert Wiener

TL;DR

Behavioral Cloning provides a natural platform to learn from real-world interactions without the need to design any reward function, and generative models prove more effective than point-wise policies at dealing with multimodal demonstration datasets.

Learning from human demonstrations provides a pragmatic alternative to the reinforcement-learning pipeline discussed in Section 3. Indeed, in real-world robotics online exploration is typically costly and potentially unsafe, and designing (dense) reward signals is a brittle and task-specific process. In general, success detection itself may often require bespoke instrumentation, while episodic training demands reliable resets—all factors complicating training RL algorithms on hardware at scale. Behavioral Cloning (BC) sidesteps these constraints by casting control an imitation learning problem, leveraging previously collected expert demonstrations. Most notably, by learning to imitate autonomous systems naturally adhere to the objectives, preferences, and success criteria implicitly encoded in the data, which obviates reduces early-stage exploratory failures and obviates hand-crafted reward shaping altogether.

Formally, let $\mathcal{D} = \{\tau^{(i)}\}_{i=1}^{N}$ be a set of expert trajectories, with $\tau^{(i)} = \{(o_t^{(i)}, a_t^{(i)})\}_{t=0}^{T_i}$ representing the i-th trajectory in $\mathcal{D}$, $o_t \in \mathcal{O}$ denoting observations (e.g., images and proprioception altogether), and $a_t \in \mathcal{A}$ the expert actions. Typically, observations $o \in \mathcal{O}$ consist of both image and proprioperceptive information, while actions $a \in \mathcal{A}$ represent control specifications for the robot to execute, e.g. a joint configuration. Note that differently from Section 3, in the imitation learning context $\mathcal{D}$ denotes an offline dataset collecting N length-T_i reward-free (expert) human trajectories $\tau^{(i)}$, and *not* the environment dynamics. Similarily, in this section $\tau^{(i)}$ represent a length-T_i trajectory of observation-action pairs, which crucially *omits entirely any reward* information. Figure 17 graphically shows trajectories in terms of the average evolution of the actuation on the 6 joints over a group of teleoperated episodes for the SO-100 manipulator. Notice how proprioperceptive states are captured jointly with camera frames over the course of the recorded episodes, providing a unified high-frame rate collection of teleoperation data. Figure 18 shows (o_t, a_t)-pairs for the same dataset, with the actions performed by the human expert illustrated just alongside the corresponding observation. In principle, (expert) trajectories $\tau^{(i)}$ can have different lengths since demonstrations might exhibit multi-modal strategies to attain the same goal, resulting in possibly multiple, different behaviors.

Behavioral Cloning (BC) (Pomerleau, 1988) aims at synthetizing synthetic behaviors by learning the mapping from observations to actions, and in its most natural formulation can be effectively tackled as a *supevised* learning problem,

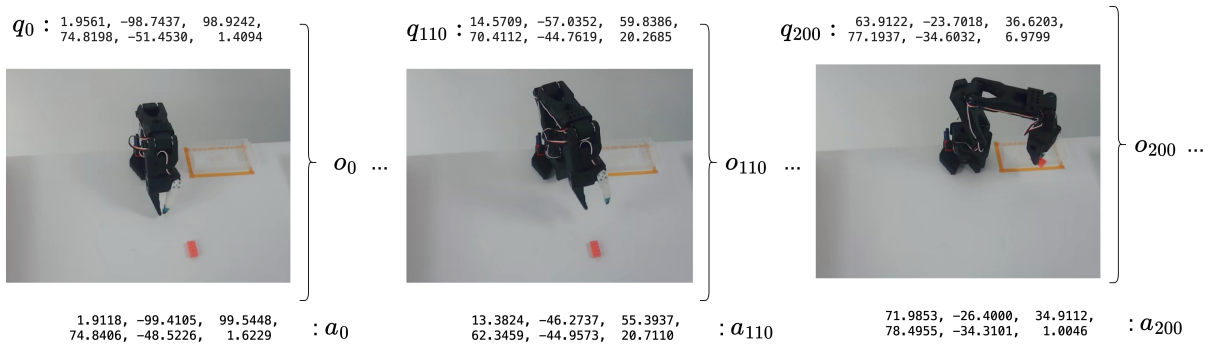


Figure 18 | Sample observations and action pairs over the course of a given trajectory recorded in `lerobot/svla_so101_pickplace`. Observations, comprising of both proprioperceptive and visual information, are recorded alongside the configuration of a second, leader robot controlled by a human expert, providing complete information for regressing actions given observation.

consisting of learning the (deterministic) mapping $f : \mathcal{O} \mapsto \mathcal{A},\ a_t = f(o_t)$ by solving

$$\min_f \mathbb{E}_{(o_t,a_t)\sim p(\bullet)} \mathcal{L}(a_t, f(o_t)), \tag{18}$$

for a given risk function $\mathcal{L} : \mathcal{A} \times \mathcal{A} \mapsto \mathbb{R},\ \mathcal{L}(a, a')$.

Typically, the expert's joint observation-action distribution $p : \mathcal{O} \times \mathcal{A} \mapsto [0, 1]$ such that $(o, a) \sim p(\bullet)$ is assumed to be unknown, in keeping with a classic Supervised Learning (SL) framework[2]. However, differently from standard SL's assumptions, the samples collected in $\mathcal{D}$, correspoding to observations of the underlying p are *not* i.i.d., as expert demonstrations are collected *sequentially* in trajectories. In practice, this aspect can be partially mitigated by considering pairs in a non-sequential order—*shuffling* the samples in $\mathcal{D}$—so that the expected risk under p can be approximated using MC estimates, although estimates may in general be less accurate. Another strategy to mitigate the impact of regressing over non-i.i.d. samples relies on the possibility of interleaving BC and data collection (Ross et al., 2011), aggregating multiple datasets iteratively. However, because we only consider the case where a single offline dataset $\mathcal{D}$ of (expert) trajectories is already available, dataset aggregation falls out of scope.

Despite the inherent challenges of learning on non-i.i.d. data, the BC formulation affords several operational advantages in robotics. First, training happens offline and typically uses expert human demonstration data, hereby severily limiting exploration risks by preventing the robot from performing dangerous actions altogether. Second, reward design is entirely unnecessary in BC, as demonstrations already reflect human intent and task completion. This also mitigates the risk of misalignment and specification gaming (*reward hacking*), otherwise inherent in purely reward-based RL (Heess et al., 2017). Third, because expert trajectories encode terminal conditions, success detection and resets are implicit in the dataset. Finally, BC scales naturally with growing corpora of demonstrations collected across tasks, embodiments, and environments. However, BC can in principle only learn behaviors that are, at most, as good as the one exhibited by the demonstrator, and thus critically provides no mitigation for the suboptimal decision making that might be enaced by humans. Still, while problematic in sequential-decision making problems for which expert demonstrations are not generally available—data migth be expensive to collect, or human performance may be inherently suboptimal—many robotics applications benefit from relative cheap pipelines to acquire high-quality trajectories generated by humans, thus justifying BC approaches.

While conceptually elegant, point-estimate policies $f : \mathcal{O} \mapsto \mathcal{A}$ learned by solving 18 have been observed to suffer from (1) compounding errors (Ross et al., 2011) and (2) poor fit to multimodal distributions (Florence et al., 2022; Ke et al., 2020). Figure 19 illustrates these two key issues related to learning *explicit policies* (Florence et al., 2022). Besides sequentiality in $\mathcal{D}$, compounding errors due to *covariate shift* may also prove catastrophic, as even small ϵ-prediction errors $0 < \|\mu(o_t) - a_t\| \leq \epsilon$ can quickly drive the policy into out-of-distribution states, incuring in less confident generations and thus errors compounding (Figure 19, left).Moreover, point-estimate policies typically fail to learn *multimodal* targets, which are very common in human demonstrations solving robotics problems, since multiple trajectories can be equally as good towards the accomplishment of a goal (e.g., symmetric grasps, Figure 19, right). In particular, unimodal regressors tend to average across modes, yielding indecisive or even unsafe commands (Florence et al., 2022). To address poor multimodal fitting, Florence et al. (2022) propose learning the generative model $p(o, a)$ underlying the samples in $\mathcal{D}$, rather than an explicitly learning a prediction function $f(o) = a$.

[2]Throughout, we will adopt the terminology and notation for SL introduced in Shalev-Shwartz and Ben-David (2014)

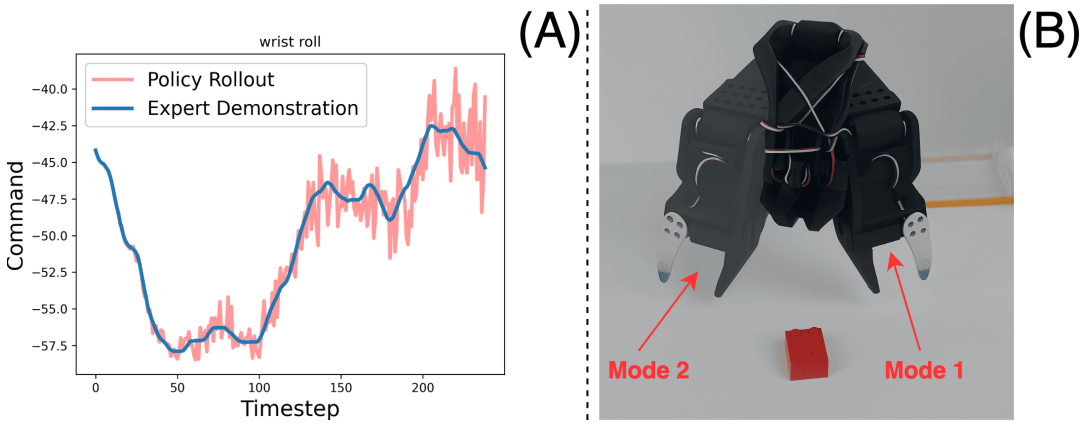


Figure 19 | Point-wise policies suffer from limitations due to (A) covariate shifts and poor approximation of (B) multimodal demonstrations. (A) Initially small errors may drive the policy out of distribution, incuring in a vicious circle ultimately resulting in failure. (B) Both modes of reaching for a target object in a scene, either left or right-first, are equally as good and thus equally as likely to be present in a dataset of human demonstrations, ultimately resulting in multimodal demonstrations.

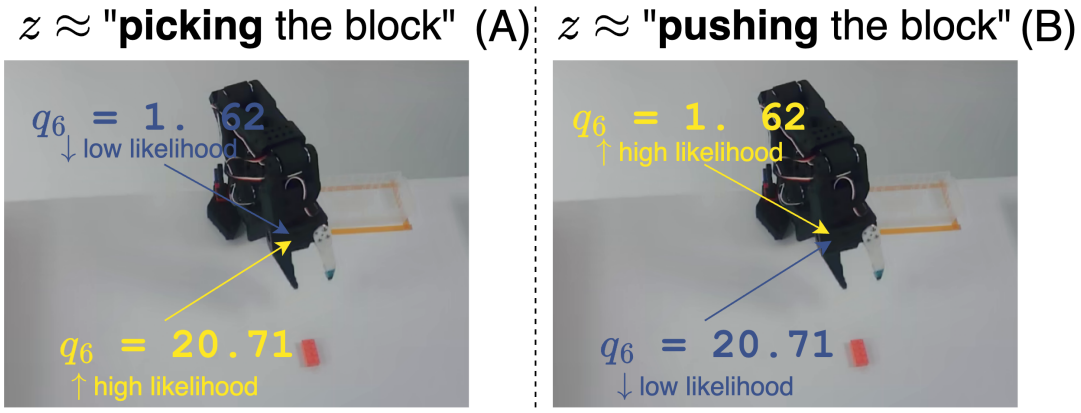


Figure 20 | Intuitively, latent variable in a single latent model may contain information regarding the task being performed, which directly results in the likelihood of the same observation-action pair being different for two different tasks. When (A) picking a block the likelihood of a wide gripper's opening should be higher than narrower one, while it should be the opposite when (B) pushing the block.

4.1 A (Concise) Introduction to Generative Models

Generative Models (GMs) aim to learn the stochastic process underlying the very generation of the data collected, and typically do so by fitting a probability distribution that approximates the unknown *data distribution*, p. In the case of BC, this unknown data distribution p represents the expert's joint distribution over (o, a)-pairs. Thus, given a finite set of N pairs $\mathcal{D} = \{(o, a)_i\}_{i=0}^{N}$ used as an imitation learning target (and thus assumed to be i.i.d.), GM seeks to learn a *parametric* distribution $p_\theta(o, a)$ such that (1) new samples $(o, a) \sim p_\theta(\bullet)$ resemble those stored in $\mathcal{D}$, and (2) high likelihood is assigned to the observed regions of the unobservable p. Likelihood-based learning provides a principled training objective to achieve both objectives, and it is thus extensively used in GM (Prince, 2023).

4.1.1 Variational Auto-Encoders

A common inductive bias used in GM posits samples (o, a) are influenced from an unobservable latent variable $z \in Z$, resulting in

$$p(o, a) = \int_{\text{supp}(Z)} p(o, a|z)p(z) \tag{19}$$

Intuitively, in the case of observation-action pairs (o, a) for a robotics application, z could be some high level representation of the underlying task being performed by the human demonstrator. In such case, treating $p(o, a)$ as a marginalization over supp(Z) of the complete joint distribution $p(o, a, z)$ natively captures the effect different tasks

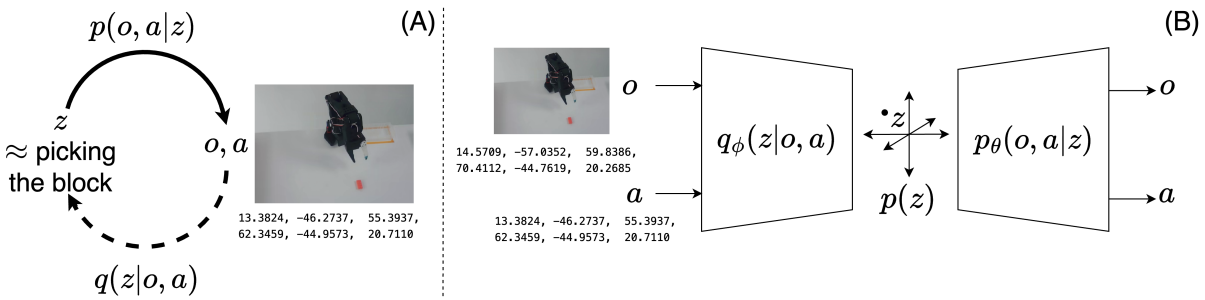


Figure 21 | (A) The latent variable model in a robotics application regulates influence between observed (o, a) variables and an unobservable latent variable. (B) VAEs approximate exact latent variable models by means of variational inference.

have on the likelihood of observation-action pairs. Figure 20 graphically illustrates this concept in the case of a (A) picking and (B) pushing task, for which, nearing the target object, the likelihood of actions resulting in opening the gripper—the higher q_6, the wider the gripper's opening—should intuitively be (A) high or (B) low, depending on the task performed. While the latent space Z typically has a much richer structure than the set of all actual tasks performed, 19 still provides a solid framework to learn joint distribution conditioned on unobservable yet relevant factors. Figure 21 represents this framework of latent-variable for a robotics application: the true, z-conditioned generative process on assigns *likelihood* $p((o, a)|z)$ to the single (o, a)-pair. Using Bayes' theorem, one can reconstruct the *posterior* distribution on $\text{supp}(Z)$, $q_\theta(z|o, a)$ from the likelihood $p_\theta(o, a|z)$, *prior* $p_\theta(z)$ and *evidence* $p_\theta(o, a)$. VAEs approximate the latent variable model presented in 19) using an *approximate posterior* $q_\phi(z|o, a)$ while regressing parameters for a parametric likelihood, $p_\theta(o, a|z)$ (Figure 21).

Given a dataset $\mathcal{D}$ consisting of N i.i.d. observation-action pairs, the log-likelihood of all datapoints under θ (in Bayesian terms, the *evidence* $p_\theta(\mathcal{D})$) can thus be written as:

$$\log p_\theta(\mathcal{D}) = \log \sum_{i=0}^{N} p_\theta((o, a)_i) \tag{20}$$

$$= \log \sum_{i=0}^{N} \int_{\text{supp}(Z)} p_\theta((o, a)_i|z)p(z) \tag{21}$$

$$= \log \sum_{i=0}^{N} \int_{\text{supp}(Z)} \frac{q_\theta(z|(o, a)_i)}{q_\theta(z|(o, a)_i)} \cdot p_\theta((o, a)_i|z)p(z) \tag{22}$$

$$= \log \sum_{i=0}^{N} \mathbb{E}_{z\sim p_\theta(\bullet|(o,a)_i)} \left[\frac{p(z)}{q_\theta(z|(o, a)_i)} \cdot p_\theta((o, a)_i|z) \right], \tag{23}$$

where we used 19 in 20, multiplied by $1 = \frac{q_\theta(z|(o,a)_i)}{q_\theta(z|(o,a)_i)}$ in 21, and used the definition of expected value in 23.

In the special case where one assumes distributions to be tractable, $p_\theta(\mathcal{D})$ is typically tractable too, and $\max_\theta \log p_\theta(\mathcal{D})$ provides a natural target for (point-wise) infering the unknown parameters θ of the generative model. Unfortunately, 23 is rarely tractable when the distribution p is modeled with approximators such as neural networks, especially for high-dimensional, unstructured data.

In their seminal work on Variational Auto-Encoders (VAEs), Kingma and Welling (2022) present two major contributions to learn complex latent-variable GMs on unstructured data, proposing (1) a tractable, variational lower-bound to 23 as an optimization target to jointly learn likelihood and posterior and (2) high-capacity function approximators to model the likelihood $p_\theta(o, a|z)$ and (approximate) posterior distribution $q_\phi(z|o, a) \approx q_\theta(z|o, a)$.

In particular, the lower bound on 23 (Evidence LOwer Bound, *ELBO*) can be derived from 23 applying Jensen's inequality—$\log \mathbb{E}[\bullet] \geq \mathbb{E}[\log(\bullet)]$—yielding:

$$\log p_\theta(\mathcal{D}) \geq \sum_{i=0}^{N} \left(\mathbb{E}_{z\sim p_\theta(\cdot|(o,a)_i)} \left[\log p_\theta((o, a)_i|z) \right] + \mathbb{E}_{z\sim p_\theta(\cdot|(o,a)_i)} \left[\log \left(\frac{p(z)}{q_\theta(z|(o, a)_i)} \right) \right] \right) \tag{24}$$

$$= \sum_{i=0}^{N} \left(\mathbb{E}_{z\sim p_\theta(\cdot|(o,a)_i)} \left[\log p_\theta((o, a)_i|z) \right] - \text{D}_{\text{KL}} \left[q_\theta(z|(o, a)_i) \| p(z) \right] \right) \tag{25}$$

The true, generally intractable posterior $p_\theta(z|o,a)$ prevents computing both the expectation and KL divergence terms in 25, and therefore Kingma and Welling (2022) propose deriving the ELBO using an *approximate* posterior $q_\phi(z|o,a)$, resulting in the final, tractable ELBO objective,

$$\text{ELBO}_{\mathcal{D}}(\theta,\phi) = \sum_{i=0}^{N}\left(\mathbb{E}_{z\sim q_\phi(\cdot|(o,a)_i)}\left[\log p_\theta((o,a)_i|z)\right] - \text{D}_{\text{KL}}\left[q_\phi(z|(o,a)_i)\|p(z)\right]\right) \tag{26}$$

From Jensen's inequality, maximizing ELBO results in maximizing the log-likelihood of the data too, thus providing a natural, tractable optimization target. Indeed, expectations can be estimated using MC estimates from the learned distributions in 26, while the KL-divergence term can typically be computed in closed-form (1) modeling q_ϕ as a Gaussian $q_\phi(z|o,a) = \mathcal{N}\big(\mu_\phi(o,a), \Sigma_\phi(o,a)\big)$ and (2) imposing a standard Gaussian prior on the latent space, $p(z) = \mathcal{N}(\mathbf{0},\mathbf{I})$.

An intuitive explanation of the learning dynamics of VAEs can be given considering the equivalent case of *minimizing the negative ELBO*, which admits a particularly interpretable factorization

$$\min_{\theta,\phi} -\text{ELBO}_{(o,a)\sim\mathcal{D}}(\theta,\phi) = \min_{\theta,\phi} \mathbf{L}^{\text{rec}}(\theta) + \mathbf{L}^{\text{reg}}(\phi) \tag{27}$$

$$\mathbf{L}^{\text{rec}}(\theta) = \mathbb{E}_{z\sim q_\phi(\cdot|o,a}\left[\log p_\theta(o,a|z)\right] \tag{28}$$

$$\mathbf{L}^{\text{reg}}(\phi) = \text{D}_{\text{KL}}\left[q_\phi(z|o,a)\|p(z)\right] \tag{29}$$

For any given (o,a) pair, the expected value term of 28 is typically computed via MC estimates, resulting in

$$-\mathbb{E}_{z\sim q_\phi(\bullet|o,a)}\left[\log p_\theta(o,a|z)\right] = \mathbf{L}^{\text{rec}} \approx -\frac{1}{n}\sum_{i=0}^{n}\log p_\theta(o,a|z_i).$$

Assuming $p_\theta(o,a|z)$ is parametrized as an isotropic Gaussian distribution with mean $\mu_\theta(z) \in \mathbb{R}^d$ and variance σ^2, the log-likelihood thus simplifies to:

$$\log p(o,a|z_i) = -\frac{1}{2\sigma^2}\left\|(o,a) - \mu_\theta(z_i)\right\|_2^2 - \frac{d}{2}\log(2\pi\sigma^2) \implies \mathbf{L}^{\text{rec}} \approx \frac{1}{n}\sum_{i=0}^{n}\left\|(o,a) - \mu_\theta(z_i)\right\|_2^2$$

Indeed, it is very common in practice to approximate from the learned likelihood $p_\theta(o,a|z)$ as a parametric distribution (e.g. Gaussians) parametrized by some learned vector of coefficients derived from $\mu_\theta(z),\ z\sim p(\bullet)$. In all such cases, learning a VAE corresponds to optimally *reconstructing* the examples in $\mathcal{D}$ by minimizing the L2-error—a very common *supervised learning* objective for regression targets—while regularizing the information compression into the latent, as under the common modeling choice $p(z) = \mathcal{N}(\mathbf{0},\mathbf{I})$ 29 regularizes the posterior limiting the expressivity of $q_\phi(z|o,a)$.

4.1.2 Diffusion Models

VAEs approximate probability distributions via a *single* latent variable model, assuming the underlying unknown distribution can be factored according to 19, and solve the variational inference problem of jointly learning the likelihood p_θ and (approximate) posterior q_ϕ for such model. In that, the unknown data distribution $p(o,a)$ is effectively approximated via $\int_Z p(z)p_\theta(o,a|z)$, and the underlying generative process reproduced by (1) sampling a latent variable and (2) learning to decode it into a (ideally) high-likelihood sample under the (unknown) $p(o,a)$. Diffusion Models (DMs) (Ho et al., 2020) are another class of GMs which treat the similar problem of approximating an underlying unknown data distribution—*variational inference*—by *partially* extending VAEs to the case where *multiple* latent variables influence each other and the generative process underlying o,a itself. In particular, DMs posit the generative process can be decomposed to a series of piece-wise (Markovian) interactions between (latent) variables (Figure 22), resulting in

$$p(\underbrace{o,a}_{=z_0}) = \int_{\text{supp}(Z_0)}\int_{\text{supp}(Z_1)}\dots\int_{\text{supp}(Z_T)} p(z_0,z_1,\dots z_T) \tag{30}$$

$$p(z_0,z_1,\dots z_T) = p(z_T)\prod_{t=0}^{T} p(z_{t-1}|z_t), \tag{31}$$

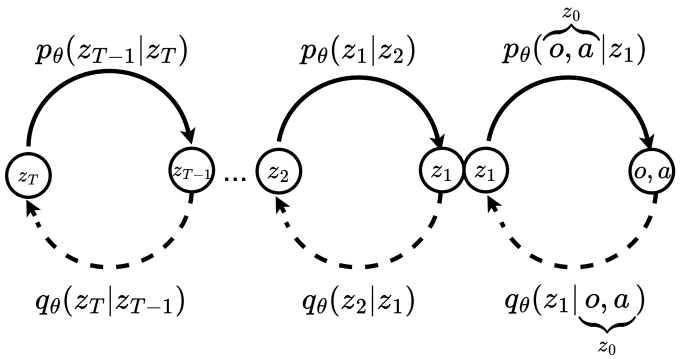


Figure 22 | HMLV models posit the data generation process is influenced by a stack of Markov-dependent latent variables, with samples from the posterior distribution being progressively higher up in the hierarchy.

where we explicitly showed the marginalization over the multiple latents in 30, and used the law of conditional probability and Markov property in 31.

Similarily to VAEs, providing an exact interpretation for the latent variables is typically not possible. Still, one fairly reasonable application-driven intuition is that, by providing a model of the hierarchical, decoupled interaction of latent variables, Hierarchical Markov Latent Variable (HMLV) models attempt to capture the different resolutions at which different conditioning factors intervene, so that in a robotics application for instance, one could naturally distinguish between early-stage trajectory planning ($t \to T$) and fine-grained adjustments ($t \to 0$). In that, HMLV models thus provide a framework to perform variational inference via multiple, sequential sampling steps from different higher level distributions instead of approximating the generative process with a single-latent variable model. DMs are a particular instantiation of HMLV models for which the posterior $q(z_t|z_{t-1}) = \mathcal{N}(z_t\sqrt{1-\beta_t}, \beta_t\mathbf{I})$ for a given $\beta_t \in \mathbb{R}^+$, thereby iteratively reducing the signal-to-noise ratio as β_t increases along the latents hierarchy.

Just like VAEs, DMs attemp to learn to reproduce an underlying data distribution $p(o,a)$ given a collection of i.i.d. samples approximating the model posited to have generated the data in the first place (30). Similarily to VAEs, DMs approximate the process of sampling from the unknown $p(o,a)$ (1) sampling from an easy-to-sample distribution (e.g., Gaussian) and (2) learning to reconstruct high-likelihood samples under the unknown distribution. However, in stark contrast with VAEs, the easy-to-sample distribution contains *no mutual information* regarding the data distribution $p(o,a)$. Crucially, as no information from the sample (o,a) (denoted as $z_0 \equiv (o,a)$ for the sake of notation) is assumed to be propagated throughout the chain of latents, the posterior $q(z_t|z_{t-1})$ assumes a relatively amicable structure in DMs, reducing complexity. The *true* likelihood $p(z_{t-1}|z_t)$ is instead typically approximated using the parametrization $p_\theta(z_{t-1}|z_t)$. In that, the information contained in the unknwon data distribution is *reconstructed* via a process in which samples from a fixed distribution are turned into (ideally) high-likelihood samples under $p(o,a)$—a process referred to as *denoising*.

Under such model, we can express the log-likelihood of an arbitrary sample as[3]

$$\begin{aligned}\log p_\theta(\underbrace{o,a}_{=z_0}) =& \mathbb{E}_{z_1\sim q(\bullet|z_0)} \log p_\theta(z_0|z_1) - \\ & \mathbb{E}_{z_{T-1}\sim q(\bullet|z_0)}\left[\mathrm{D}_{\mathrm{KL}}(q(z_T|z_{T-1})\|p(z_T))\right] - \\ & \sum_{t=1}^{T-1} \mathbb{E}_{(z_{t-1},z_{t+1})\sim q(\bullet|z_0)}\left[\mathrm{D}_{\mathrm{KL}}(q(z_t|z_{t-1})\|p_\theta(z_t|z_{t-1}))\right],\end{aligned} \tag{32}$$

providing an optimization target in the form of $\max_\theta \log p_\theta(\mathcal{D})$.

In their seminal work on using DMs for variational inference, Ho et al. (2020) introduce major contributions regarding solving $\min_\theta -\log p_\theta(o,a)$. In particular, Ho et al. (2020) exclusively adopt a fixed *Gaussian* posterior in the form of $q(z_t|z_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}z_{t-1}, \beta_t\mathbf{I})$. The choice of adopting Gaussians has profound implications on the generative process modeled. Indeed, under the (mild) assumption that the variance is sufficiently small $\beta_t \leq \eta, \eta \in \mathbb{R}^+$, Sohl-Dickstein et al. (2015) proved that the likelihood $p(z_{t-1}|z_t)$ is Gaussian as well, which allows for the particularly convenient parametrization of the approximate likelihood $p_\theta(x_{t-1}|x_t) = \mathcal{N}(\mu_\theta(x_t,t), \Sigma_\theta(x_t,t)),\ t \in [1,T]$, as well as for closed-form tractability of the KL-divergence terms in 32. Further, the posterior's structure also enables

[3] $o,a = z_0$ for the sake of notation. Steps omitted for brevity. See Section A in Ho et al. (2020) for a complete derivation.

an analytical description for the distribution of the t-th latent variable, $q(z_t|z_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}z_0, (1-\bar{\alpha}_t)\mathbf{I})$, with $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{k=1}^{t} \alpha_k$, which conveniently prevents iterative posterior sampling.

Finally, adopting Gaussian posteriors permits a particularly pleasing interpretation of the dynamics of training DMs (Permenter and Yuan, 2024). By using Gaussian posteriors, the hierarchical latent variables effectively lose increasingly more information circa the original (unknown) distribution's sample, z_0, increasingly distributing according to a standard Gaussian and thus containing no information at all (Figure 23). Figure 23 illustrates this procedure on a simplified, bidimensional observation-action distribution, where we considered $o = q_2$ and $a = q_2^h$, with q_2 representing the robot's *elbow flex* actuation and q_2^h the human teleoperator's robot elbow flex.

Because the recorded behavior is teleoperated, measurements mostly distribute along the line $a = o + \eta, \eta \sim N(0, 1)$, with η-variability accouting for minor control inconsistencies (Figure 24). Using Gaussian posteriors—i.e., adding Gaussian noise—effectively simulates a *Brownian motion* for the elements in the distribution's support (in Figure 23, $\mathcal{O} \times \mathcal{A}$), whereby information *diffuses away* from the samples, and comparing the diffused samples to the original data points one can derive an estimate of the total displacement induced by diffusion. Under the only assumption that the likelihood of the diffused samples is low under the original unknown data distribution, then one can effectively approximate the unkwown distribution by learning to *reverse* such displacement. This key intuition allows to write a simplified training objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{t, z_0, \epsilon}\left[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}z_0 + \epsilon\sqrt{1-\bar{\alpha}_t}, t)\|^2\right], \quad t \sim \mathcal{U}(\{1, \dots, T\}), \quad z_0 \sim \mathcal{D}, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{33}$$

In this simplified (minimization) objective, the optimization process differs from 32 in that, rather than maxizing p_θ directly, the parameters θ of the pairwise likelihood $p_\theta(z_{t-1}|z_t)$ are adjusted to *predict the total displacement* ϵ for a randomly long ($t \sim \mathcal{U}(\{1, \dots, T\}$)) diffusion process starting from a sample of the target distribution.

By learning the total displacement from a generally, uninformative corrupted sample obtained diffusing information and a sample from an unknown distribution—significant ($\|\epsilon\| > 0$) whenever input and target distribution are sufficiently different— Ho et al. (2020) show that one can approximate the underlying distribution reversing the displacement, *denoising* samples. Interestingly, under the hypothesis real-world data belongs to a single higher dimensional manifold (Manifold Hypothesis), Permenter and Yuan (2024) show that diffusion learns the gradient of a distance function from any off-point manifold (such as perturbed, uniformative samples), and the data manifold itself. Following this gradient—i.e., denoising a sample from an uninformative distribution—corresponds to projecting back into the manifold, yielding a procedure to sample from unknown distributions by means of Euclidean projection. Indeed, under the assumption that $p_\theta(z_{t-1}|z_t)$ is Gaussian, then sampling $z_{t-1} \sim p_\theta(\bullet|z_t)$ corresponds to computing

$$z_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(z_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(z_t, t)\right) + \sigma_t\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{34}$$

thus showing that the lower-level latent variables in a DM can be obtained by iteratively removing noise from the one-step higher order variable, using the noise regressor $\epsilon_\theta(z_t, t)$ learned minimizing 33.

4.1.3 Flow Matching

The posterior parametrization adopted by DMs proved traditionally effective, yet it raised concerns circa its efficiency at inference time, where a possibly large of compute-expensive denoising steps are needed in order to recover a sample from the target distribution. Flow Matching (FM) (Lipman et al., 2023) extends DMs to the general case of arbitrary, parametrized likelihood and posteriors, and in this defines a superseding class of GMs providing a unified framework for learning *continuous transformations* between distributions, encompassing and generalizing DMs. Instead of a *stochastic, discrete, multi-step* denoising process, FM aims to learn a *deterministic, continuous, differentiable flow* $\psi[0, 1] \times Z \mapsto Z$, formalized starting from possibly time-dependent vector field $v : [0, 1] \times Z \mapsto Z$ transporting samples from a simple prior distribution p_0—e.g., a standard Gaussian—to a more complex, potentially unknown data distribution p_1 over time. Note how FM models time $t \in [0, 1]$ to be varying continuously while moving away *from* an easy-to-sample distribution p_0 *towards* the unknown data-distribution, p_1. This results in a continuous and deterministic trajectory for each sample, which can be more efficient to generate compared to the stochastic paths of DMs. Formally, FM can be fully characterized by an ordinary differential equation (ODE) relating instantaneous variations of flows with the underlying vector field, and hence providing complete trajectories over the distributions' support when integrating over time,

$$\frac{d}{dt}\psi(z, t) = v(t, \psi(t, z)) \tag{35}$$

$$\psi(0, z) = z \tag{36}$$

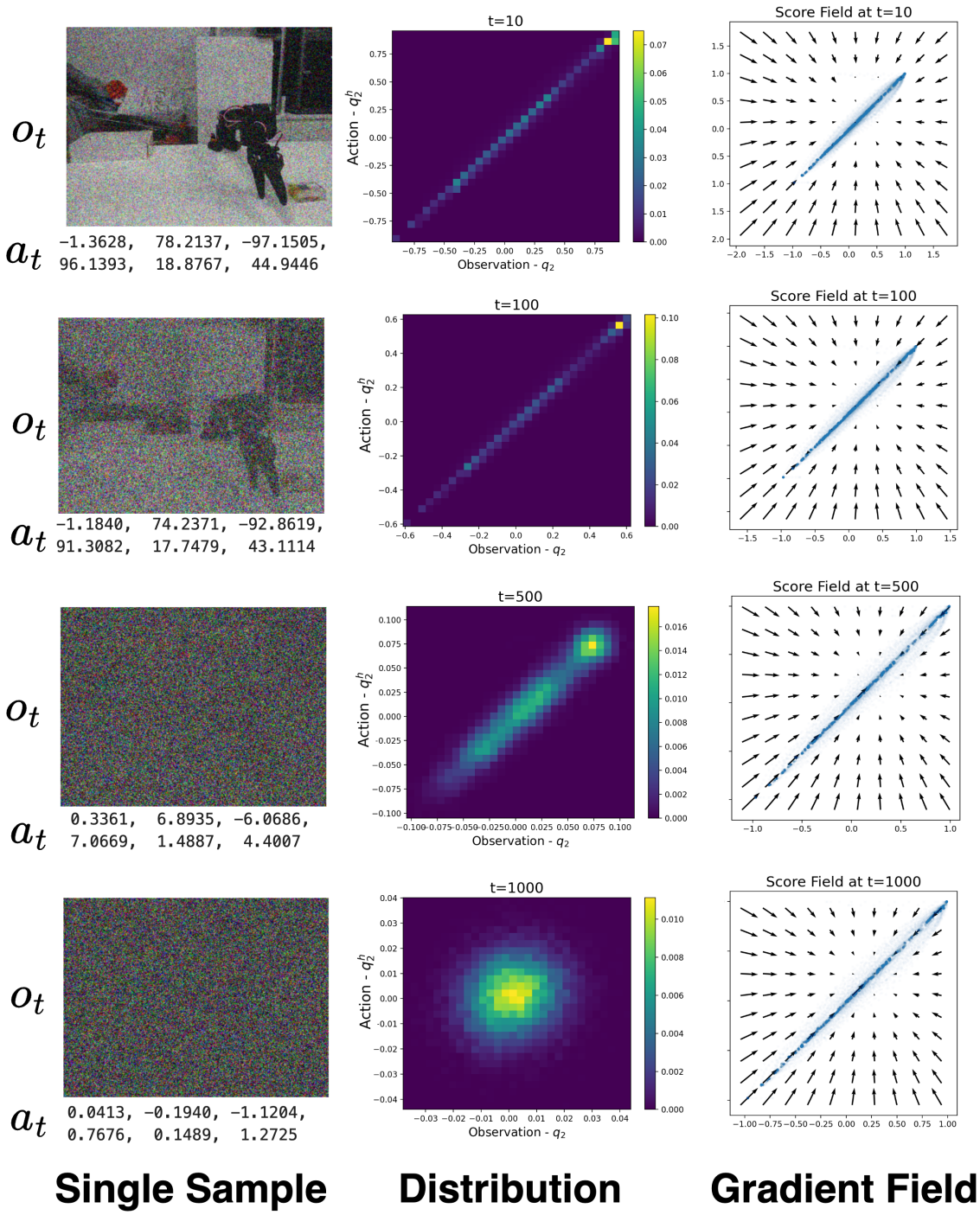


Figure 23 | DMs iteratively corrupt samples (left) from an unknown distribution into a quasi-standard Gaussian (center), learning the displacement field (right) that permits to reconstruct samples from the unknown target distribution by iteratively denoising samples of a tractable, easy-to-sample distribution.

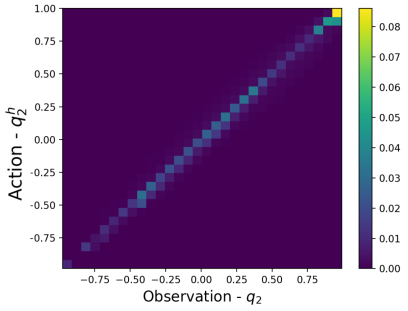


Figure 24 | A joint action-observation distribution, in the simplified case where the observation is the elbow-flex actuation in a SO-100, and the action is the recorded position for the same joint in the teleoperator arm. The motion recorded being teleoperated, the points distribute along a the diagonal.

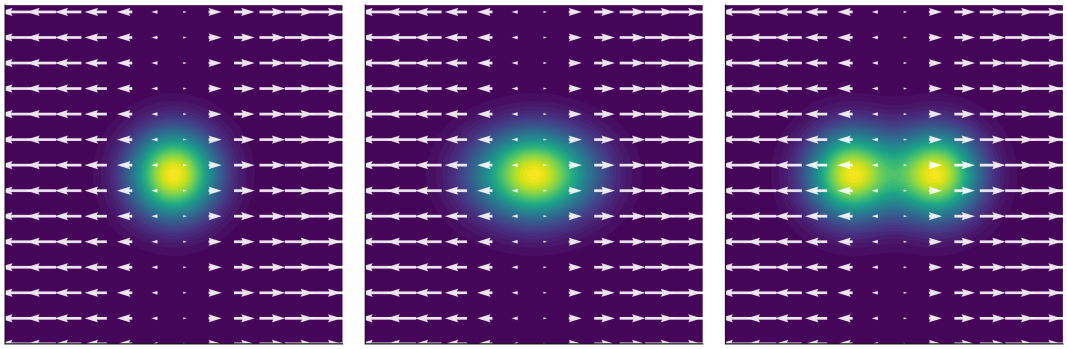


Figure 25 | Probability distributions can be modified applying vector fields resulting in a flow of mass in the support. When acting over time, vector fields can effectively change the distribution's structure.

FM proved very effective in a variety of applications, ranging from image (Esser et al., 2024) and video generation (Polyak et al., 2025) to robotics control (Black et al., 2024). Most notably, in their introductory work on FM for GM, Lipman et al. (2023) show how DMs can be seen as a specific instance of FM where the *conditional* target vector field u approximated by the noise regressor corresponds to

$$u(t, z|z_0) = \frac{\frac{d}{dt}\alpha(1-t)}{1-(\alpha(1-t))^2}(\alpha(1-t)z - z_0), \quad \alpha(t) = e^{-\frac{1}{2}\int_0^t \beta(s)ds}, \quad \forall z_0 \in \mathcal{D} \tag{37}$$

Note that the traditional discrete-time noise-scheduler $\beta_{t\,t=0}^{T}$ is now generalized to a continuous map $\beta : [0, 1] \mapsto \mathbb{R}^+$. Crucially, Lipman et al. (2023) prove that by exclusively optimizing the vector field for individual data points $z_0 \in \mathcal{D}$ individually, one also retrieves the optimal flow to morph the entire support of the initial distribution p_0 into p_1 s.t.$\mathcal{D} \sim p_1$.

While the noising schedule of DMs results in a stochastic process that resembles a random walk, FM allows for more general—potentially, deterministic—likelihood and posterior parametrization. In the FM literature the likelihood and posterior probabilty densities defined along a HMLV model are typically jointly referred to as a *probability path*, where the distributions for successive adjacent transitions in the HMLV model are related by the (normalized) flow between them (Figure 25). The inherent flexibility of FM is one of their key advantages over DMs, as it opens up the possibility of *learning* more efficient paths. For instance, one can design probability paths inspired by Optimal Transport (OT)—a subdiscipline studying the problem of finding the most efficient way to morph one probability distribution into another. Probability paths obtained through OT paths tend to be *straighter* than diffusion paths (Figure 26), which can lead to faster and more stable training, as well as higher-quality sample generation with fewer steps at inference time. By avoiding unnecessary backtracking associated with the inherent stochastic nature of both the noising and denoising process in DMs, test-time compute is typically significantly reduced, while retaining comparable results (Lipman et al., 2023).

In practice, FM can be applied to generative modeling by learning a vector field regressor $v_\theta(z, t)$ to approximate a given target vector field $u(t, z)$. In the particular case of DMs, $u(t, z)$ is defined as in 37, while in priciple the target vector field can be learned to induce a particular transportation, or fixed according to OT. Given a sample

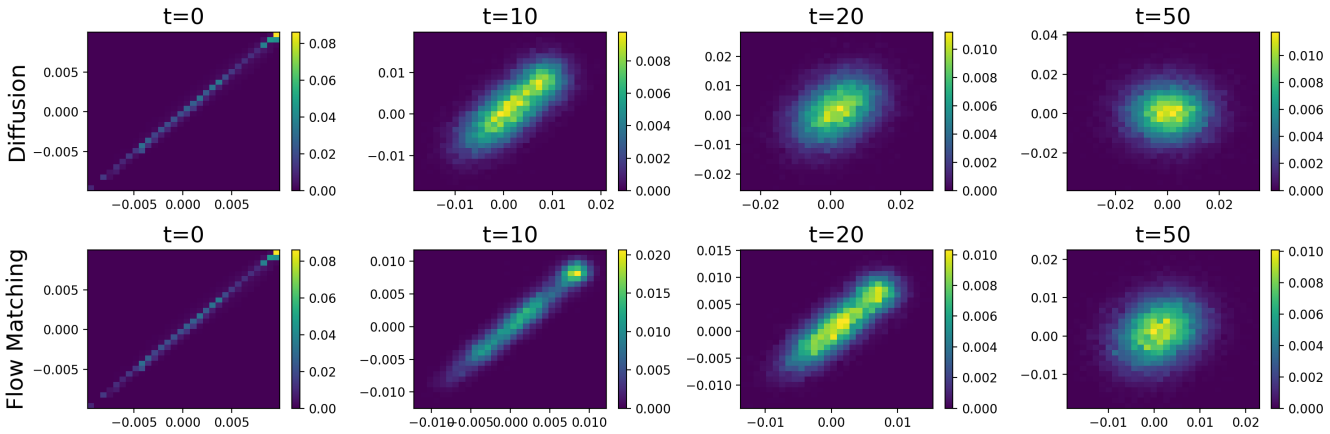


Figure 26 | Compared to diffusion, flow matching distorts distribution along a less randomic pattern, resulting in a clearer interpolation between soruce and target distribution. The visualization shows an example comparison between these two methods on joint distribution of robot observations and actions over $T = 50$ steps.

from the data distribution $z_1 \sim p_1$ and a sample from an easy-to-sample prior $z_0 \sim p_0$, CFM defines a simple path between them using *linear interpolation* between samples $z_t = (1-t)z_0 + tz_1$, resulting in the target vector field $u(t, z_t) = z_1 - z_0$. Then, a FM model can be trained with the simple regression objective defined as

$$\mathcal{L}(\theta) = \mathbb{E}_{t,z_0,z_1}\left[\|v_\theta((1-t)z_0 + tz_1, t) - (z_1 - z_0)\|^2\right], \quad t \sim \mathcal{U}([0,1]), \tag{38}$$

where $z_0 \sim p_0(\bullet)$ and $z_1 \sim p_1(\bullet)$. Note how in 38—differently from 33—time is assumed to be varying continuously $t \sim \mathcal{U}([0,1])$ rather than discretely $t \sim \mathcal{U}(\{0,1\})$, a key property of flow-based models. The objective in 38 directly regresses the learned vector field onto the simple, straight path connecting a point from the prior and a point from the data, providing a simulation-free training procedure that is both stable and efficient. At inference time, samples are generated by starting with $z_0 \sim p_0$ and iteratively refined according to $\frac{dz}{dt} = v_\theta(z_t, t)$ for $t \in [0,1]$—an operation that can be numerically carried out with standard ODE solvers.

4.2 Action Chunking with Transformers

While GMs prove useful in learning complex, high-dimensional multi-modal distributions, they do not natively address the compouding errors problem characteristic of online, sequential predictions. In Action Chunking with Transformers (ACT), Zhao et al. (2023) present an application of VAEs to the problem of learning purely from offline trajectories, introduce a simple, yet effective method to mitigate error compounding, learning high-fidelity autonomous behaviors. Drawing inspiration from how humans plan to enact atomically sequences of the kind $a_{t:t+k}$ instead of single actions a_t, Zhao et al. (2023) propose learning a GM on a dataset of input demonstrations by modeling *action chunks*. Besides contributions to learning high-performance autonomous behaviors, Zhao et al. (2023) also introduce hardware contributions in the form of a low-cost bimanual robot setup (ALOHA) capable of performing fine-grained manipulation tasks, such as opening a lid, slotting a battery in its allotment or even prepare tape for application.

On the robot learning side of their contributions, Zhao et al. (2023) adopt transformers as the architectural backbone to learn a *Conditional* VAE (Sohn et al., 2015). Conditional VAEs are a variation of the more standard VAE formulation introducing a conditioning variable on sampling from the latent prior, allowing the modeling of *one-to-many* relationships between latent and data samples. Further, in stark contrast with previous work (Florence et al., 2022; Janner et al., 2022), Zhao et al. (2023) do not learn a full joint $p_\theta(o, a)$ on observation and actions. While the *policy* distribution $p_\theta(a|o)$ can in principle be entirely described from its joint $p_\theta(o, a)$, it is often the case that the conditional distribution is intractable when using function approximators, as $p_\theta(a|o) = \frac{p_\theta(o,a)}{\int_{\mathcal{A}} p_\theta(o,a)}$ and the integral in the denominator is typically intractable. Instead of modeling the full joint using a vanilla VAE, Zhao et al. (2023) propose learning a *conditional* VAE (Sohn et al., 2015) modeling the policy distribution directly $p(a|o)$.

In practice, when learning from demonstrations adopting CVAEs results in a slight modification to the VAE objective

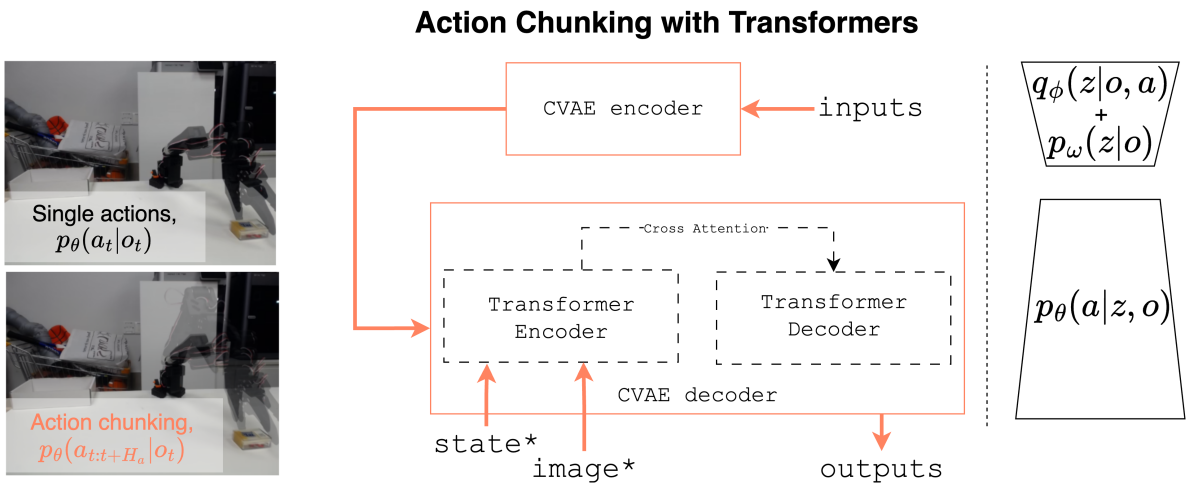


Figure 27 | Action Chunking with Transformer (ACT), as in Zhao et al. (2023). ACT introduces an action chunking paradigm to cope with high-dimensional multi-modal demonstration data, and a transformer-based CVAE architecture.

in 26, which is adapted to

$$\text{ELBO}_{\mathcal{D}}(\theta, \phi, \omega) = \sum_{i=0}^{N} \left(\mathbb{E}_{z \sim q_\phi(\cdot|o_i, a_i)} \left[\log p_\theta(a_i|z, o_i) \right] - \text{D}_{\text{KL}} \left[q_\phi(z|o_i, a_i) \| p_\omega(z|o_i) \right] \right) \tag{39}$$

Notice how in 39 we are now also learning a new set of parameters ω for the prior distribution in the latent space. Effectively, this enables conditioning latent-space sampling (and thus reconstruction) during training, and potentially inference, providing useful when learning inherently conditional distributions like policies. Further, ACT is trained as a β-CVAE (Higgins et al., 2017), using a weight of the KL regularization term in 39 as an hyperparameter regulating the information condensed in the latent space, where higher β results in a less expressive latent space.

In their work, Zhao et al. (2023) ablated using a GM to learn from human demonstrations compared to a simpler, supervised objective, $\mathcal{L}_1(a, a') = \|a - a'\|_1$. Interestingly, they found the performance of these two approaches to be comparable when learning from *scripted* demonstrations. That is, when learning from data collected rolling out a predetermined set of commands $[q_0^c, q_1^c, \dots]$, GM did *not* prove competitive compared to standard supervised learning. However, when learning from human demonstrations—i.e., from data collected executing commands coming from a human controller $[q_0^h, q_1^h, \dots]$—they found performance (success rate on a downstream task) to be severily (-33.3%) hindered from adopting a standard supervised learning objective compared to a richer, potentially more complex to learn variational objective, in keeping with the multimodal nature of human demonstrations data and findings presented in Florence et al. (2022). The authors also ablate the action chunking paradigm, reporting significant performance gains for performing action chunking (1% vs. 44% success rate). To avoid acting openloop, Zhao et al. (2023) design an inference process consisting in performing inference at every timestep t and then aggregate overlapping chunks using chunks' exponential moving average.

In ACT (Figure 27), inference for a given observation $o \in \mathcal{O}$ could be performed by (1) computing a prior $p_\omega(z|o)$ for the latent and (2) decoding an action chunk from a sampled latent $z \sim p_\omega(\bullet|o)$, similarily to how standard VAEs generate samples, with the exception that vanilla VAEs typically pose $p(z|o) \equiv p(z) \sim N(\mathbf{0}, \mathbf{I})$ and thus skip (1).

However, the authors claim using a deterministic procedure to derive z may benefit policy evaluation, and thus avoid sampling from the conditional prior at all. At test time, instead, they simply use $z = \mathbf{0}$, as the conditional prior on z used in training is set to be the unit Gaussian. At test time, conditioning on the observation o is instead achieved through explicitly feeding proprioperceptive and visual observations to the decoder, $p_\theta(a|z, o)$, while during training z is indeed sampled from the approximate posterior distribution $p_\phi(z|o, a)$, which, however, disregards image observations and exclusively uses proprioperceptive states to form o for efficiency reasons (as the posterior q_ϕ is completely disregarded at test time).

4.2.1 Code Example: Learning ACT

using act example

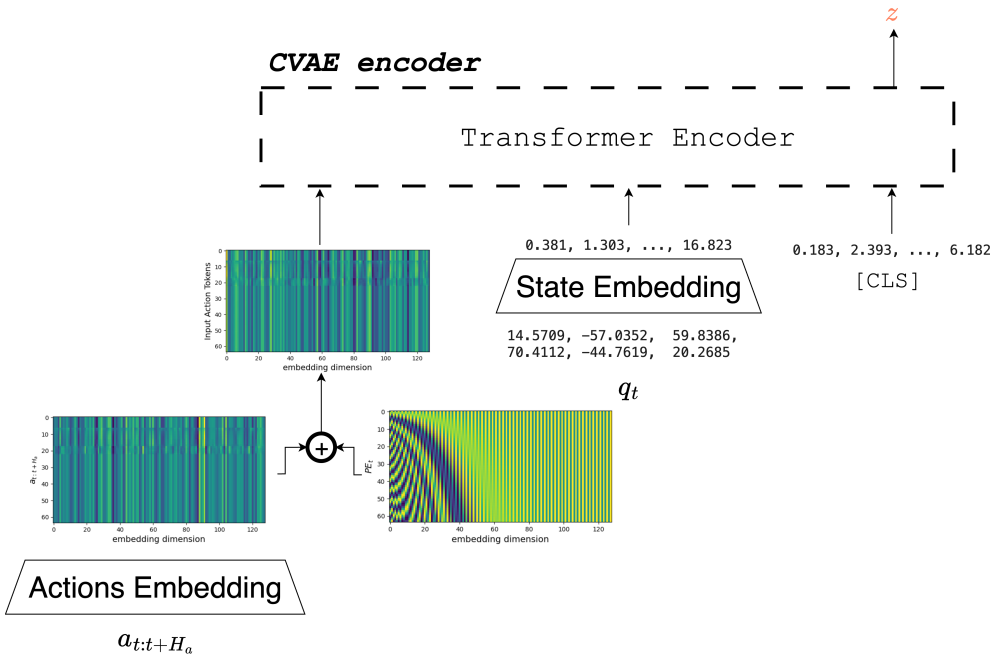


Figure 28 | The CVAE encoder used in ACT. Input action chunks are first embedded and aggregated with positional embeddings, before being processed alongside embedded proprioperceptive information, and a learned `[CLS]` token used to aggregate input level information, and predict the style variable z. The encoder is entirely disregarded at inference time.

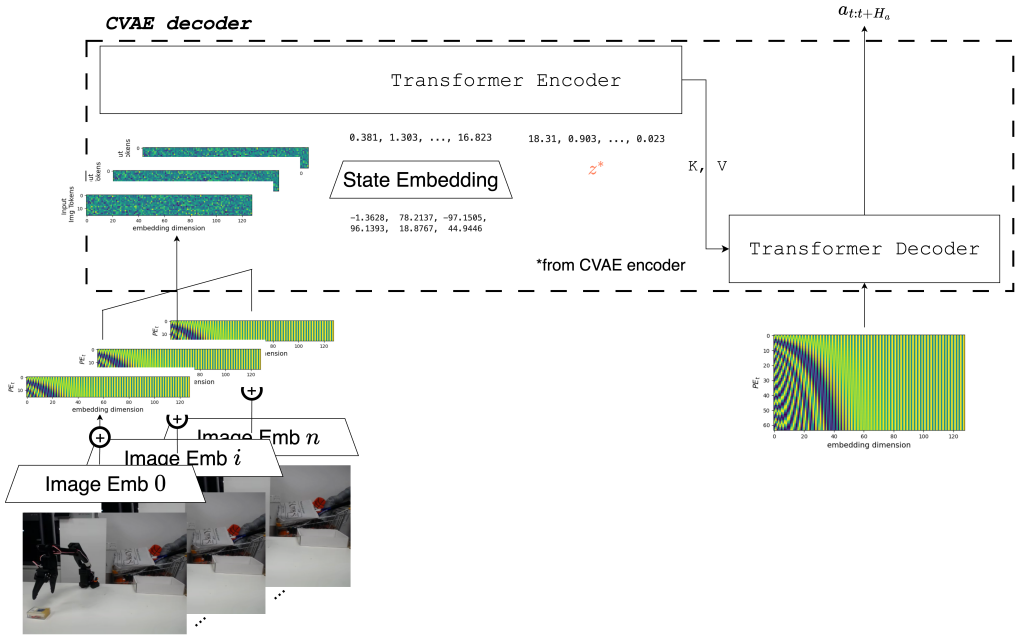


Figure 29 | The CVAE decoder used in ACT, comprising of a full encoder-decoder Transformer architecture. Camera observations from all n camera views are first embedded using pre-trained visual encoders, and then concatenated to the corresponding positional embeddings. Then, alongside embeddings for the proprioperceptive information available and the style variable z retrieved from the CVAE encoder, the Transformer encoder shares the matrices K, Q with the Transformer decoder, trained to decode fixed position embeddings into action valid chunks.

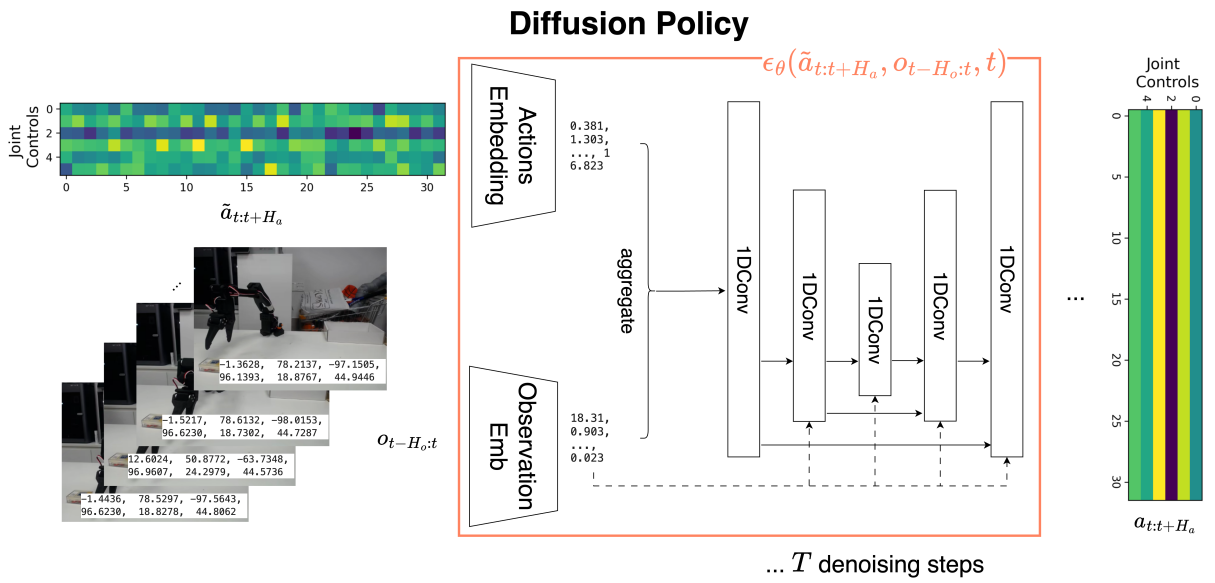


Figure 30 | The Diffusion Policy archicture, as in Chi et al. (2024). A stack of H_o previous observations is used as external conditioning to denoise a group of H_a actions. Conditioning is used at every layer of a U-Net block, and in practice allows to obtain fully-formed action chunks with as little as $T = 10$ denoising steps.

4.3 Diffusion Policy

DMs proved very effective in approximating complex highly dimensional distributions, such as distributions over images (Ho et al., 2020) or videos (Polyak et al., 2025), thanks to their inherent capability to deal with multimodal data and training stability. In Diffusion Policy (DP), Chi et al. (2024) present an application of DMs the field of robot learning, leveraging diffusion to model human expert demonstrations in a variety of simulated and real-world tasks. Similarily to Action Chunking with Transformer (Zhao et al., 2023), Chi et al. (2024) (1) adopt a modified *observation-conditioned target distribution* instead of the full joint $p(o, a)$ and (2) predict multiple actions into the future instead of a single action. Besides the intractability of the observations' marginal $p_\theta(o)$ given $p_\theta(o, a)$, DP's rationale for modeling the data distribution via $p_\theta(a|o)$ stems from the rather test-time compute intensive nature of diffusion, whereby generating actions *alongside* observations is likely to result in higher complexity and thus a likely larger number of denoising operations, which would prove ultimately pointless considering robotics applications rely on the capability to generate controls rather than reproducing observations.

In practice, conditioning on observation data is achieved conditioning the added noise regressor ϵ_θ introduced in 33 on a stack of T_o observations, resulting in the *conditional* simplified diffusion objective

$$\begin{aligned} \mathcal{L}(\theta) = \mathbb{E}_{t, a_{t:t+H_a}, \epsilon}\left[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} a_{t:t+T_a} + \epsilon\sqrt{1-\bar{\alpha}_t}, t, o_{t-T_o:t})\|^2\right], \\ t \sim \mathcal{U}(\{1, \dots, T\}), \quad a_{t:t+T_a}, o_{t-T_o:t} \sim \mathcal{D}, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \end{aligned} \tag{40}$$

Notice how in 40 the noise regressor is conditioned both on the latent variable rank t *and* on a stack of previous observations $o_{t-T_o:t}$. Chi et al. (2024) claim the combination of (1) conditioning on a horizon of previous observations and (2) predicting multiple actions into the future allows DP to *commit to specific modes* in the data at inference time, which proves essential for good performance and avoiding undecisiveness.

Figure 30 shows the convolution-based version of the architecture proposed by Chi et al. (2024), illustrating inference on a single sample from $\mathcal{D}$ for simplicity. An arbitrarily noisy chunk of H_a actions $\tilde{a}_{t:t+H_a}$ is mapped to a learned high-dimensional space. Similarily, both image observations and poses are embedded before being aggregated to the action embeddings. Then, a U-Net (Ronneberger et al., 2015) is trained to regress the noise added into $\tilde{a}_{t:t+H_a}$, using observation conditioning information at every layer and seeking to optimize 40. At inference time, the noise predictor is used to predict the quantity of noise at every $t \in [T, \dots, 0]$ and iteratively subtract it from $\tilde{a}_{t:t+T_a}$, reversing the diffusion process simulated in training conditioned on $o_{t-T_o:t}$ to predict $a_{t:t+T_a}$.

Training using 50-150 demos (15-60 minutes of teleoperation data) DP achieves strong performance on a variety of simulated and real-world tasks, including dexterous and deformable manipulation tasks such as sauce pouring and mat unrolling. Notably, the authors ablated the relevance of using RGB camera streams as input to their

policy, and observed how high frame-rate visual observations can be used to attain performance (measured as success rate) comparable to that of state-based policies, typically trained in simulation with priviledged information not directly available in real-world deployments. As high-frame rate RGB inputs naturally accomodate for dynamic, fast changing environments, Chi et al. (2024)'s conclusion offers significant evidence for learning streamlined control policies directly from pixels. In their work, Chi et al. (2024) also ablate the performance of DP against their baseline against the size of the dataset collected, showing that DP outperforms the considered baseline for every benchmark size considered. Further, to accelerate inference, Chi et al. (2024) employ Denoising Diffusion Implicit Models (Song et al., 2022), a variant of Denoising Diffusion Probabilistic Models (Ho et al., 2020) (DDPM) adopting a strictly deterministic denoising paradigm (differently from DDPM's natively stochastic one) inducing the same final distribution's as DDPM's, and yet resulting in 10 times less denoising steps at inference time (Chi et al., 2024). Across a range of simulated and real-world tasks, Chi et al. (2024) find DPs particularly performant when implementing a transformer-based network as ϵ_θ, although the authors note the increased sensitivity of transformer networks to hyperparameters and thus explicitly recommend starting out with a simpler, convolution-based architecture for diffusion (Figure 30), which are however reported to be biased towards learning low-frequency components (Tancik et al., 2020) and thus may prove more challenging to train with non-smooth action sequences.

4.3.1 Code Example: Learning Diffusion Policies

using diff
sion poli
example

4.4 Optimized Inference

Modern visuomotor policies output *action chunks*–sequences $\pi(o_t) = \mathbf{A}_t$ with $\mathbf{A}_t = \left(a_t, a_{t+1}, \dots, a_{t+H_a}\right)$ being a sequence of $H_a \gg 1$ low-level commands enqueued in an action queue, originating from an environment observation, o_t. Predicting series of actions instead of single commands proved essential in learning complex, multi-modal behavior (Zhao et al., 2023; Chi et al., 2024).

Typically, the robot executes the entire action chunk $\mathbf{A}_t$, before a new observation o_{t+H_a} is passed to the policy π to predict the next chunk. This results in open-loop inference in between observations captured every H_a timesteps. Zhao et al. (2023) adopts a different strategy whereby the robot controller interleaves chunk prediction $\mathbf{A}_t \leftarrow \pi(o_t)$ and chunk consumption $a_t \leftarrow \textsc{PopFront}(\mathbf{A}_t)$, computing a new chunk of actions at every timestep t and aggregating the predicted chunks on overlapping sections. While adaptive—every observation at every timestep o_t is processed—such approaches rely on running inference continuously, which can be prohibitive in resource-constrained scenarios, such as edge deployments.

A less resource-intensive approach is to entirely exhaust the chunk $\mathbf{A}$ before predicting a new chunk of actions, a strategy we refer to as *synchronous* (sync) inference. Sync inference efficiently allocates computation every H_a timesteps, resulting in a reduced average computational burden at control time. In contrast, it inherently hinders the responsiveness of robot systems, introducing blind lags due to the robot being *idle* while computing $\mathbf{A}$.

We directly assess the lack of adaptiveness of robot systems due to acting open-loop, and the presence of lags at runtime by decoupling action chunk prediction $\mathbf{A}$ from action execution $a_t \leftarrow \textsc{PopFront}(\mathbf{A}_t)$, developing an *asynchronous* (async) inference stack (1), whereby a RobotClient sends an observation o_t to a PolicyServer, receiving an action chunk $\mathbf{A}_t$ once inference is complete (31). In this, we avoid execution lags by triggering chunk prediction while the control loop is still consuming a previously available queue, aggregating it with the newly incoming queue whenever available. In turn, async-inference tightens the loop between action prediction and action execution, by increasing the frequency at which observations are processed for chunk prediction. Crucially, decoupling action prediction from action execution also directly allows to allocate more computational resources on a remote policy server sending actions to the robot client over networks, something which may prove very effective in resource-constrained scenarios such as low-power robots.

Implementation details *Async* inference (1) tightens the control loop by capturing observations more often, directly eliminates idle gaps at runtime, and (2) directly allows to run inference on more powerful computational resources than the ones typically available onboard autonomous robotic platforms.

Algorithmically, we attain (1) on the RobotClient-side by consuming actions from a readily available queue until a threshold condition on the number of remaining actions in the queue ($|\mathbf{A}_t|/H_a < g$) is met. When this condition is triggered, a new observation of the environment is captured and sent to the (possibly remote) PolicyServer. To avoid redundant server calls and erratic behavior at runtime observations are compared in joint-space, and near-duplicates are dropped. Two observations are considered near-duplicates if their distance in joint-space is under

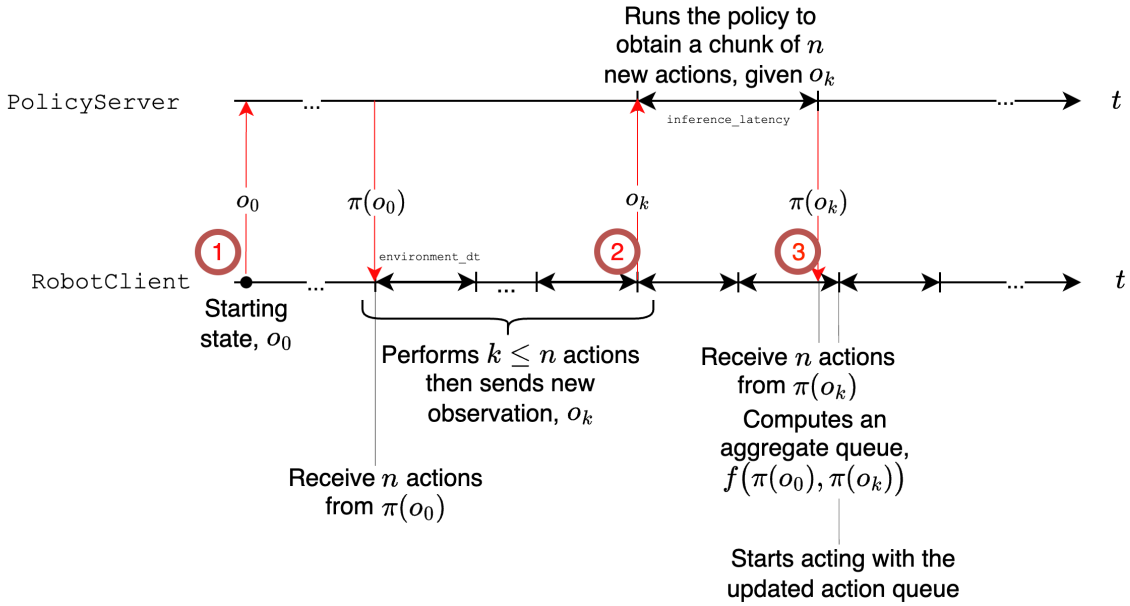


Figure 31 | **Asynchronous inference**. Illustration of the asynchronous inference stack. Note that the policy can be run on a remote server, possibly with GPUs.

Algorithm 1 Asynchronous inference control-loop

Input: horizon T, chunk size H_a, threshold $g \in [0, 1]$
Init: capture o_0; send o_0 to PolicyServer; receive $\mathbf{A}_0 \leftarrow \pi(o_0)$
for t **to** H_a **do**
  $a_t \leftarrow \text{PopFront}(\mathbf{A}_t)$
  $\text{Execute}(a_t)$ ▷ execute action at step t
  if $\frac{|\mathbf{A}_t|}{H_a} < g$ **then** ▷ queue below threshold
    capture new observation, o_{t+1}
    if $\text{NeedsProcessing}\ (o_{t+1})$ **then** ▷ similarity filter, or triggers direct processing
      `async_handle` $\leftarrow \text{AsyncInfer}(o_{t+1})$ ▷ Trigger new chunk prediction (non blocking)
      $\tilde{\mathbf{A}}_{t+1} \leftarrow \pi(o_{t+1})$ ▷ New queue is predicted with the policy
      $\mathbf{A}_{t+1} \leftarrow f(\mathbf{A}_t, \tilde{\mathbf{A}}_{t+1})$ ▷ aggregate overlaps (if any)
    end if
  end if
  if NotCompleted(`async_handle`) **then**
    $\mathbf{A}_{t+1} \leftarrow \mathbf{A}_t$ ▷ No update on queue (inference is not over just yet)
  end if
end for

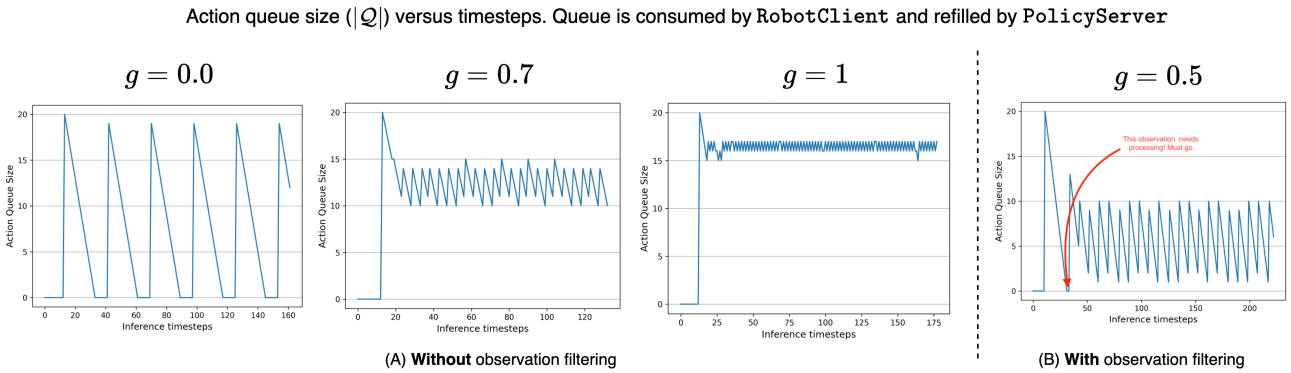


Figure 32 | Action queue size evolution at runtime for various levels of g when (A) not filtering out observation based on joint-space similarity and (B) filtering out near-duplicates observation, measuring their similarity in joint-space.

a predetermined threshold, $\epsilon \in \mathbb{R}_+$. Importantly, when the queue available to robot client eventually becomes empty, the most recent observation is processed regardless of similarity.

Interestingly, the behavior of async inference can be studied analytically. First, let ℓ be a random variable modeling the time needed to receive an action chunk $\mathbf{A}$ after sending an observation o, i.e. the sum of (1) the time to send across the observation o between the RobotClient and PolicyServer, $t_{C\to S}$ (2) the inference latency on the PolicyServer, ℓ_S and (3) the time to send $\mathbf{A}$ between the PolicyServer and RobotClient, $t_{S\to C}$. Assuming independence, $\mathbb{E}[\ell] = \mathbb{E}[t_{C\to S}] + \mathbb{E}[\ell_S] + \mathbb{E}[t_{S\to C}]$ which can be further simplified to $\mathbb{E}[\ell] \simeq \mathbb{E}[\ell_S]$, assuming communication time is (1) equal in both directions and (2) negligible with respect to the inference latency. Second, let Δt be the environment's control cycle. With a real-world frame-rate of 30 frames per second, $\Delta t = 33$ms. Consequently, exhausted queues at runtime–i.e. being idle awaiting for a new chunk–are avoided for $g \geq \frac{\mathbb{E}[\ell_S]/\Delta t}{H_a}$. In this, the queue threshold g plays a major role relatively to the availability of actions to the RobotClient.

32 illustrates how the size of the action chunk $|\mathbf{A}_t|$ evolves over time for three representative values of g, detailing the following key scenarios:

- **Sequential limit** ($g = 0$). The client drains the entire chunk before forwarding a new observation to the server. During the round-trip latency needed to compute the next chunk, the queue is empty, leaving the robot *incapable of acting*. This reproduces the behavior of a fully sequential deployment and results in an average of $\mathbb{E}[\ell_S]$ idle seconds.
- **Asynchronous inference** ($g \in (0, 1)$). Allowing the client to consume $1 - g$ of its available queue $\mathbf{A}_{t-1}$ before triggering inference for a new action queue $\mathbf{A}_t$, amortizing computation while keeping the queue from emptying. The overlap between successive chunks provides a buffer against modeling errors without the full cost of the $g = 1$ regime. The updated queue $\mathbf{A}_t$ is obtained aggregating queues on the overlapping timesteps between $\mathbf{A}_{t-1}$ and the incoming $\tilde{\mathbf{A}}_t$.
- **Compute-intensive limit** ($g = 1$). As an extreme case, and in keeping with Zhao et al. (2023), an observation is sent at *every* timestep. The queue is therefore almost always filled, with only a minor saw-tooth due to$\Delta t/\mathbb{E}[\ell_s] < 1$. While maximally reactive, this setting incurs one forward pass per control tick and can prove prohibitively expensive on limited hardware. Importantly, because the client is consuming actions while the server computes the next chunk, the available queue never gets filled again.

32 emphasizes the trade-off governed by g: small values place result in idle periods, whereas $g \approx 1$ assumes a highly accurate model and pays a significant compute price. In practice, choosing $g \in (0, 1)$ allows to strike a balance between reactivity against resource budgets. If not for the aforementioned similarity filter, the RobotClient would send observations for processing every $(1 - g)H_a \cdot \Delta t$ seconds, receiving a new chunk of actions every $(1 - g)H_a \cdot \Delta t + \mathbb{E}[\ell_S]$, on average. The presence of the observation similarity filter dilates this processing time, and serves the scope of avoiding the robot stalling due to the queue being constantly integrated with an incoming, nearly identical, action chunk. In particular, 32 results in a queue which is filled with incoming actions *unless* near-duplicate observations are filtered out from the processing pipeline. For clarity, the red arrow in 32 highlights a timestep where the observation similarity mechanism is bypassed, forcing a (nearly identical) observation to be processed as the queue results empty.

4.4.1 Code Example: Using Async Inference

async inference example

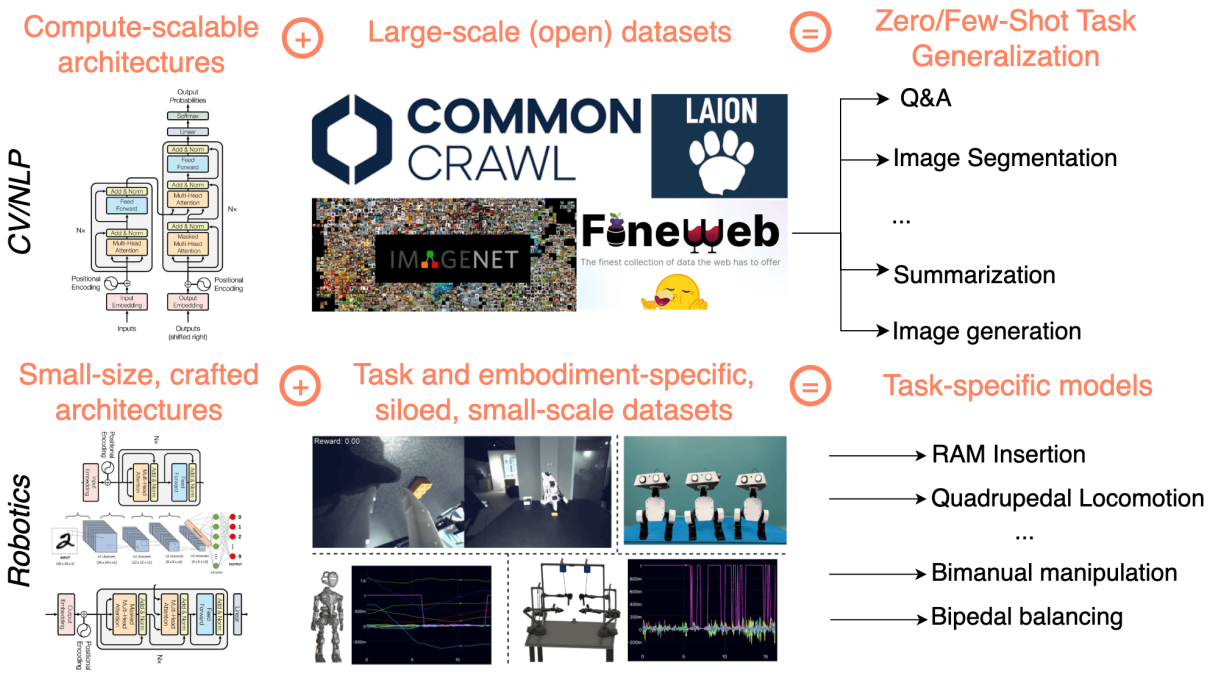


Figure 33 | Fields within ML such as Computer Vision and NLP converged on the development of foundation models, trained on a variety of large scale models and capable to perform multiple downstream tasks (top). Conversely, robotics suffered from limited standardization in terms of the architectures used, and siloed, task specific datasets, incurring in a high degree of fragmentation which traditionally hindered the development of generalist models for robotics in favour of task-specific models (bottom).

5 Generalist Robot Policies

Specialization is for insects

Robert A. Heinlein

TL;DR

Openly available large scale datasets and the development of stable, expressive and efficient architecture fostered research on the development of generalist robot policies that can operate across embodiment and tasks.

The advent of large models trained on internet-scale datasets has drastically influenced fields like Computer Vision (CV) and Natural Language Processing (NLP), shifting the paradigm towards combining (1) an initial, task-agnostic large-scale pre-training stage and a (2) task-specific, adjustment phase. The pre-training/adaptation paradigm has now largely replaced more classic approaches consisting of task-specific data collection, curation and model training in many subdomains within CV and NLP, motivated by the main drawback of limited scalability for *task-specific approaches*, traditionally labor intensive. Factors including (1) the advancements in generalist models learned with self-supervision for perception (Oquab et al., 2024) or semantic understanding (Devlin et al., 2019) and (2) the popularization collective efforts to aggregate large-scale openly available datasets (Collaboration et al., 2025; Khazatsky et al., 2025) are increasingly pushing the field of robot learning towards the pre-train-and-adapt paradigm. This shift taps into the long-standing challenge of developing generalist robot policies, and holds the premise to surpass traditionally siloed approaches to robotics problems and develop a *foundation robotics model*. While Section ?? introduced methods for learning *single-task policies* such as ACT or Diffusion Policy, in this section we present advancements in developing *generalist, multi-task, policies*, capable of performing a wide range of tasks across different environments and embodiments, and guided by unstructured instructions given via natural language.

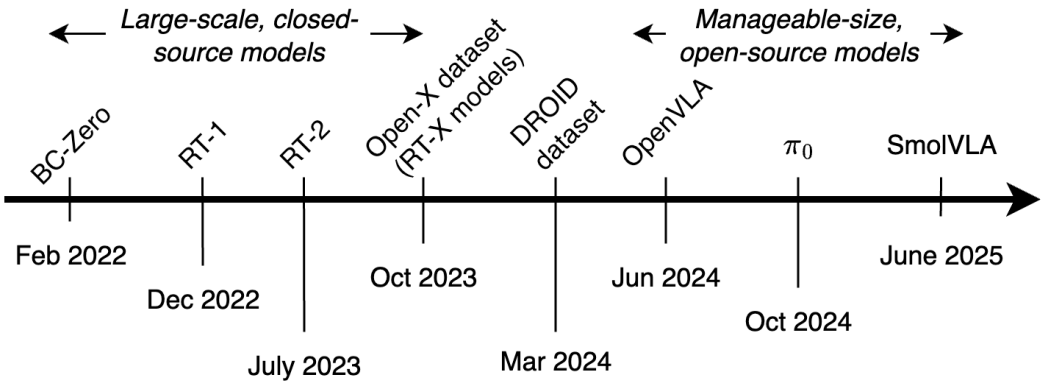


Figure 34 | Early efforts in the development of generalist models for robotics include BC-Zero (Jang et al., 2022), RT-1 (Brohan et al., 2023b), and RT-2 (Brohan et al., 2023a): large scale models trained on thousands of demonstrations. The open release of the Open-X (Collaboration et al., 2025) and DROID datasets (Khazatsky et al., 2025) fostered the development of open source models: OpenVLA (Kim et al., 2024), π_0 (Black et al., 2024) and SmolVLA (Shukor et al., 2025).

5.1 Preliminaries: Models and Data

The remarkable success of foundation models in NLP and CV is predicated on two core principles: architectural innovation and joint data-compute scaling. The transformer architecture proved instrumental in capturing long-range dependencies in sequential data such as text, and its stability and expressivity made it the *de facto* standard for modern large-scale models trained on internet-scale amounts of data. In stark contrast with popular NLP (Raffel et al., 2023) and CV (Deng et al., 2009) general-purpose datasets, the field of robotics has historically developed around task-specific datasets which hinders scalability across problems, resulting in a concrete data deficit for general-purpose robot learning. Unlike the wealth of relatively readily available text and images on the internet, robotics data is intrinsically embodied—datasets collected for a manipulation robot typically differ entirely from locomotion datasets. Further, datasets consisting of expert demonstrations are (1) intrinsically expensive to collect (2) and notoriously heterogeneous—different human experts may perform the same task optimally yet in very different ways. In particular, since each expert trajectory is tied to a specific robot platform and the operating conditions of its environment and task, data heterogeneity has long posed a *methodological* challenge for scaling robotics datasets via aggregation. Beyond this, heterogeneity also raises *conceptual* issues: naively mixing data across embodiments can induce negative transfer, as control strategies developed in isolation for different robot systems in different environments may even conflict when combined. Thus, the high degree of fragmentation of robotics datasets and tasks has traditionally led to the development of *specialist* policies, trained on small, task-specific datasets, and which excel at their designated task but fail to generalize to new situations (Figure 33).

Motivated by the pursuit of generalist robot policies, the research community started investigating what and how to integrate from other domains within ML. Figure 34 shows a timeline of some of the most popular contributions attempting at developing generalist policies. Starting from BC-Zero, a latent variable model trained on 25K+ demonstrations, the field has now evolved into π_0, a transformer-based model trained on 10M+ demonstrations and exhibiting strong few-shot capabilities across tasks and embodiments. For starters, Robotics Transformer 1 (RT-1) (Brohan et al., 2023b) represented a significant step in the direction of developing a generalist robot policies over prior work including (1) BC-Zero (Jang et al., 2022) and (2) Gato (Reed et al., 2022), in that Brohan et al. (2023b) uses a much larger and diverse set of training tasks compared to both BC-Zero and Gato. In particular, RT-1 uses a transformer architecture, and is trained on as many as 130k human-recorded trajectories collected over 13 robots in the span on 17 months. RT-1 learns to process a history of camera images and a natural language instruction, and feeds the resulting sequence of high-dimensional tokens to a transformer, trained using a *classification loss on a discretized actions space* consisting of 6 256 bins, each for each joint of a 6-dof robotic arm.

Perhaps motivated by the contemporary successes of the transformer architecture in both CV and NLP, the same group of authors investigated using a discrete output space to model—inherently continuous—quantities such as actions, leveraging a (1) more powerful architecture and (2) scaling up the dataset used (Brohan et al., 2023a, RT-2). In RT-2, Brohan et al. (2023a) propose inheriting internet-scale semantic knowledge from large-scale multi-modal datasets to learn a single, *unified model* for robotics control. Such a model, termed *Vision-Language-Action* (VLA) in the original RT-2 paper, effectively casts robot control as a language modeling problem, and in particular as a Visual Question-Answering (VQ&A) task, whereby the output token space used to represent *string* tokens is shared with the

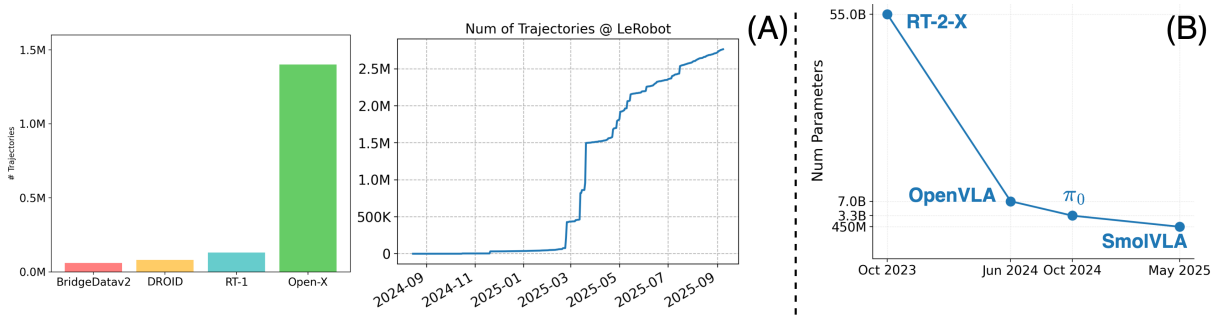


Figure 35 | Robot learning is undergoing a paradigmatic shift: centralized data collections (A, left) are increasingly larger, often comprising Ms of demonstrations, and (A, right) decentralized approaches to data collection are also rising as an alternative for large scale data collection. (B) Generalist models are also becoming increasingly smaller and easier to run on limited hardware.

8-bits tokens used to represent the 256 actuation levels of a 6-dof robot joint. In their work, Brohan et al. (2023a) propose co-fine-tuning then-leading large-scale VLMs such as PaLIX (Chen et al., 2023) or PaLM-E (Driess et al., 2023) on a mix of web and robotics data, thus complementing VQ&A training with robotics-specific signal, learning to directly output robot actions in a shared token space for visual and language inputs. Using large models trained on internet-scale data as backbones for VLAs allows models to tap into the rich semantic knowledge embedded in the VLM's parameters, interpret new commands as well as recognize unseen objects by connecting them to concepts acquired while pre-training. For instance, Brohan et al. (2023a) show that while RT-2 has never been explicitly trained to repurpose tools for a hammering task, it can still combine its semantic understanding of images, so that when asked which object between (1) a piece of paper, (2) a pair of headphones or (3) a rock may be used instead of a hammer, it answers correctly, (3).

Traditionally, research involved not only training the model but also collecting the underlying data, a costly and time-consuming process—for instance, Jang et al. (2022) gathered 25K+ trajectories before training, while RT-1 required 130K+. In turn, the data used in robot learning research efforts have traditionally proved rather fragmented, tailored to the specific task considered by the specific group of researchers who collected it, ultimately hindering integration. The Open X-Embodiment project (Collaboration et al., 2025) was a landmark effort to address the data fragmentation problem, curating the aggregation of 60 *existing* robotics datasets from 22 different robot embodiments and 21 institutions, resulting in a total 1.4M of cross-embodiments, cross-tasks, openly-available trajectories. Besides the contribution of an aggregate, large scale dataset, Collaboration et al. (2025) also demonstrated significant positive transfer *across tasks and embodiments*, showing that a single model trained on multi-embodiment data can outperform specialist models trained on their respective single-embodiment datasets. The Distributed Robot Interaction Dataset (DROID) (Khazatsky et al., 2025) represents another significant step towards addressing the problem of scarse and disaggregated data in robot learning, providing a unique dataset consisting of 75K+ human demonstrations collected in realistic (*in-the-wild*) manipulation settings, providing another cornerstone for building general-purpose robot policies. Recently, foundational datasets curated through large, centralized efforts, are increasingly complemented by decentralized, community-driven collection of robotics data. Software libraries as `lerobot` have been instrumental in enabling decentralized collection of large amounts of data, providing the infrastructure for researchers and practitioners to easily contribute trajectories from range of embodiments, democratizing data access via distributed collection.

The success of large, proprietary models like RT-1 and RT-2, highlighted a growing accessibility gap in robotics research, as training and deploying large-scale models requires computational resources simply unattainable for most research institutions. The OpenVLA project (Kim et al., 2024) emerged in direct contrast of closed-source counterparts, as a community-driven effort to create powerful, openly available VLAs. In particular, Kim et al. (2024) trained OpenVLA by exclusively leveraging openly available data (970K+ from the Open-X dataset), and share training recipes alongside the model weights. Architecturally, OpenVLA integrates a pre-trained vision encoder to project visual tokens into the embedding space of Llama2-7B (Touvron et al., 2023) language model backbone. The language model backbone is then used to predict *discrete action tokens* over 256 activation levels.

Figure 35 illustrates graphically the two most relevant trends in modern robot learning. As datasets collected via centralized, cross-institutions cooperation of increasing size are made available for the research community, decentralized datasets collected by individual researchers and practitioners have also gained traction recently, closing the gap with academic benchmarks thanks to community-contributed datasets. Further, models used across tasks and embodiments are also becoming much more compute-efficient, and as a result the models' size has been consistently

reducing over time, with consequent gains for autonomous robots in real-world, resource-constrained environments.

5.2 Modern VLAs

Modern recipes to train large scale VLAs extend early efforts to learn foundation models from large amounts of data via BC, introducing significant advancements concerning both architectural and procedural aspects. From an architectural perspective, modern VLAs such as π_0 (Black et al., 2024) leverage a *unified transformer model* for efficiency of computation, while maintaining specialized sub-components within the model for visual perception and action prediction, enabling cross-task performance via language conditioning. Crucially, modern VLAs including Black et al. (2024)[π_0] and Shukor et al. (2025)[SmolVLA] adopt *unified* transformer models employing disjoint set of weights (*experts*) for compute-efficient visual-semantic understanding and robotic control. Procedurally, modern VLAs complement advanced Vision-Language Model (VLM) backbones with action-specific modules (1) adopting mid-sized *action experts* to model continuous actions distributions $p(a_{t:t+H_a}|o_t)$—avoiding discrete action tokens entirely—and (2) relying on *action chunking* (Zhao et al., 2023, Section **??**) as a strategy to reduce error compounding when predicting multiple actions learning from inherently non-i.i.d. data, such as demonstration data.

These architectural and procedural innovations present three benefits. First, developing architectures that exploit internet-scale pre-trained backbones allows to fully capitalizes on the vast world knowledge and skills state-of-the-art VLMs exhibit, preventig models from needing to learn visual, linguistic and semantic concepts from scratch. Second, using generative models for continuous action distributions allows to learn rich, multimodal data distributions, a much more likely scenario in the big-data regime typically tackled while developing generalist policies. Further, introducing two separate components for perception and action planning could enable using Mixture of Experts (MoE) architectures (Fedus et al., 2022), more efficient to run and thus resulting in faster inference—a key features for models deployed in real-world scenarios. This new paradigm has been at the core of some of the most capable generalist policies developed to date, capable to few-shot adapt to novel tasks and to perform highly dexterous manipulation tasks, ranging from end-to-end folding laundry, to bussing tables.

5.2.1 VLMs for VLAs

VLMs are designed to process both visual and textual modalities—most commonly by taking both images and text as input and generating text conditioned on the visual context. Recent advances in VLMs have been driven by the success of LLMs, with many approaches building upon pretrained LLMs and adopting similar training paradigms to the ones used in language modeling. Typically, VLMs (Alayrac et al., 2022; Laurençon et al., 2024; Lin et al., 2024) are constructed by integrating a pretrained vision encoder (Radford et al., 2021; Zhai et al., 2023; Fini et al., 2024) with a pretrained LLM (Grattafiori et al., 2024; Jiang et al., 2023). Training then proceeds in multiple multimodal stages, beginning with a large-scale pretraining on datasets containing image-text pairs (Schuhmann et al., 2022; Byeon et al., 2022) and interleaved vision-language corpora (Laurençon et al., 2023; Zhu et al., 2023), all followed by a supervised fine-tuning stage on instruction-tuning datasets (Liu et al., 2023; Tong et al., 2024; Laurençon et al., 2024). The inherent multimodal nature of VLMs enables them to jointly reason over vision and language. Pre-training on vast internet-scale datasets allows these models to associate visual patterns with textual descriptions, thereby acquiring a rich semantic understanding of the world—knowledge about objects, their properties, and relationships—without explicit supervision for each concept. In turn, integrating a VLM as a perception backbone for a VLA allows the complete model to inherit rich world knowledge, sidestepping the need to learn visual and semantic representations from scratch. In principle, this allows the robot to ground high-level natural language instructions in its visual context, and possibly recognize unseen objects by connecting them to pre-trained concepts absorbed during pre-training, improving on the possibility to generalize to novel scenarios.

Recently, compute efficiency has also become a central focus in VLM research. Several works aim to reduce training costs by using smaller, more diverse datasets (Liu et al., 2023; Dai et al., 2023; Bai et al., 2025; Zhu et al., 2024; Tong et al., 2024), training smaller-scale models (Marafioti et al., 2025; Korrapati, 2024; Yao et al., 2024), or by adapting pretrained unimodal models by tuning only a small subset of parameters (Shukor et al., 2023; Vallaeys et al., 2024; Mañas et al., 2023; Koh et al., 2023; Tsimpoukelli et al., 2021; Li et al., 2023). While the majority of VLM research focuses on image and text modalities, recent work has demonstrated that similar techniques can be extended to integrate additional modalities, such as video and audio (Wang et al., 2025; Liu et al., 2024; Zhang et al., 2025; Kong et al., 2024)—a particularly promising direction of research for robotics applications, where multiple sensor modalities can be integrated effectively. This trend towards efficiency is paramount for robotics applications, where policies must operate under the stringent constraints of real-world deployment. Indeed, robots often possess limited on-board computational resources and must react in real-time to dynamic environments. Smaller and faster VLMs

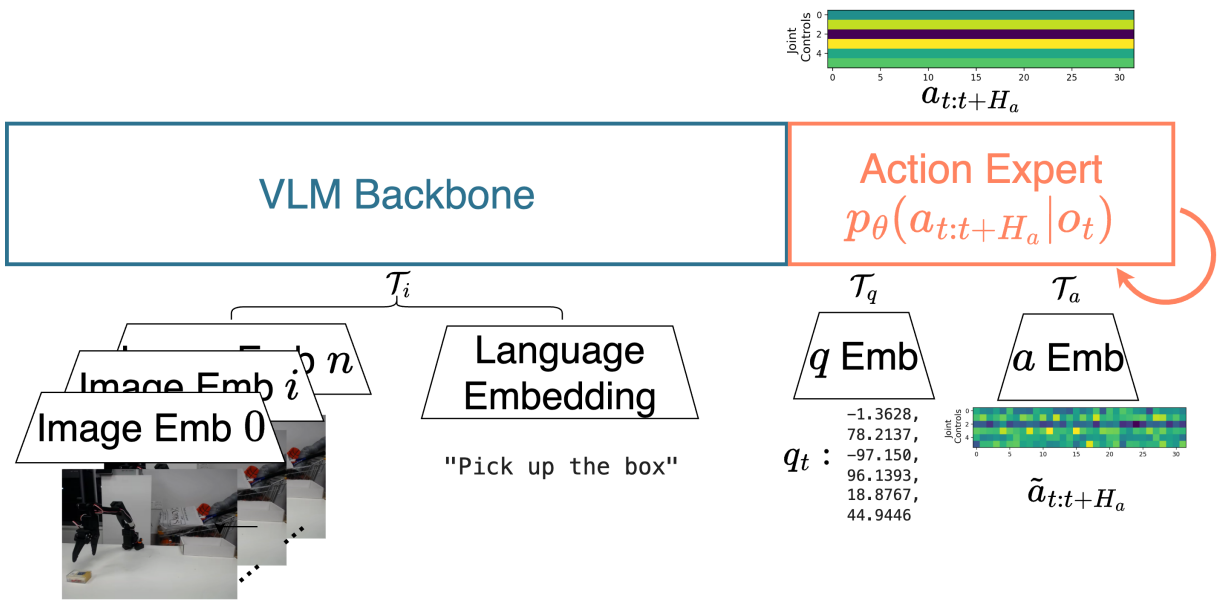


Figure 36 | The $\pi_0$architecture, as in Black et al. (2024). Vision and language tokens are routed to a VLM backbone which is prevented from attending robot proprioperceptive states and action tokens, which are instead routed to a smaller subset of weights within the architecture. The architecture is trained with Flow Matching on 10M+ trajectories from a mixture of closed and openly available datasets.

have thus become quintessential for developing responsive autonomous systems, enabling high-frequency control loops by reducing the latency between perception and action.

5.3 π_0

π_0 (Black et al., 2024) introduce a VLA consisting of a MoE architecture consisting of (1) a pre-trained VLM backbone (Gemma 2.6B (Team et al., 2024)) and (2) a dedicated action expert used to generate continuous actions via flow matching. Images and language are embedded with a late-fusion VLM (PaliGemma), while proprioceptive state and actions chunks are routed to a smaller action expert expert, initialized from scratch. The two separate experts communicate via self-attention layers, but maintain disjoint weights to obtain query, key and values matrices at each layer, maintaining specialization while efficiently allocating computation.

Concretely, π_0 is a unified transformer with two disjoint sets of weights ϕ, θ. A larger VLM backbone p_ϕ initialized from Gemma 2.6B processes multiple image frames obtained from multiple cameras points $[\{I_t\}_{t=1}^n]$, as well as a language instruction $[\ell_t]$ used to describe the task considered. Concurrently, a 300M-parameter *action expert* based on a similar transformer architecture is used processes the robot proprioperceptive state q_t and an action chunk $a_{t:t+H_a}$ (Figure 36). The different expert networks operate separately in processing the respective inputs and turning them into query, key and value matrices, and only share information between each other via self-attention layers. The outputs from the VLM backbone are disregarded, while the vector field regressed by the action expert is used to iteratively refine the action process. In particular, $\pi_0$uses a *blockwise causal attention mask* over tokens belonging to three separate blocks: (1) image and language tokens $\mathcal{T}_i$ obtained from $[\{I_t\}_{t=1}^n, \ell_t]$, (2) proprioperceptive tokens $\mathcal{T}_q$ obtained from q_t, and (3) the action tokens $\mathcal{T}_a$ for items in the chunk $a^\tau_{t:t+H_a}$ at time τ in the flow-matching process. Notably, *within* each block the attention operations are bidirectional, while across blocks, future blocks are masked out. Formally, this corresponds to using the attention mask

$$\mathbf{A} = \begin{array}{c} \\ \mathcal{T}_i \\ \mathcal{T}_q \\ \mathcal{T}_a \end{array} \begin{array}{c} \begin{array}{ccc} \mathcal{T}_i & \mathcal{T}_q & \mathcal{T}_a \end{array} \\ \begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{pmatrix} \end{array}, \quad \mathbf{1}: \text{Bidirectional Attention}, \ \mathbf{0}: \text{Masked Attention}$$

Note how *intra*-block directional attention allows tokens to communicate freely, while *inter*-block communication is mediated by the attention mask $\mathbf{A}$. *Blockwise causal masking* effectively prevents the pre-trained perception-language tokens from attending to robotics-tokens, likely out of distribution for VLM backbones traditionally trained on large

corpora of internet, non-robotics, data. Crucially, because communication is obstructed between image-language tokens, proprioperceptive and action tokens, one can cache keys and values across denoising steps at runtime time, incuring in a reduced computational footprint and faster inference.

In π_0, both the VLM backbone and action expert are update using a *flow matching* loss, and in particular are updated minimizing:

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{\tau, \epsilon, o_t, a_{t:t+H_a}} \left[\left\| v_\theta(\underbrace{\tau a_{t:t+H_a} + (1-\tau)\epsilon}_{\tilde{a}_{t:t+H_a}}, o_t, \tau) - (\epsilon - a_{t:t+H_a}) \right\|^2 \right], \tag{41}$$

$$\tau \sim \text{Beta}_{[0,s]}(1.5, 1), \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad o_t, a_{t:t+H_a} \sim \mathcal{D}$$

Where the experts parametrized by the separate weights ϕ, θ interact with each other via self-attention layers only, so that the action expert v_θ internal computations also depend on the VLM backbone's parameters ϕ. Importantly, Black et al. (2024) minimize 41 over both the multimodal backbone and action expert parameters, thus updating the internal representations of the VLM using BC-specific gradients. In contrast, Driess et al. (2025) later show that failing to insulate the VLM knowledge from the flow matching gradients actually harms performance. Inference is performed iteratively refining action chunks while numerically forward-integrating the vector field predicted by the action expert,

$$a^{\tau+\delta}_{t:t+H_a} = a^{\tau}_{t:t+H_a} + \delta v_\theta(a^{\tau}_{t:t+H_a}, o_t) \tag{42}$$

Flow matching (Lipman et al., 2023, Section4.1.3) can be seen as a continuous time, detetrministic generalization of Diffusion and has proven effective in modeling highly complex multi-modal distributions, including those over images and video. In turn, its application to large-scale data collections of multiple human behaviors across tasks and embodiments appears rather consequential, particularly considering how it can enable faster inference via a reduced number of denoising steps—as few as 10, in π_0. In particular, the action expert is model as a conditional flow matching model. Each action token embeds a noisy action $a^\tau_i \in a^\tau_{t:t+H_a}$, alongside a sinusoidal encoding of the *flow process* timestep τ. The action expert then leverages full bidirectional attention across the H_a action tokens provided, as well as attends to previous proprioperceptive and image-language tokens as well. Interestingly, differently from a standard flow matching pipeline Lipman et al. (2023), τ is *not* sampled from a uniform distribution $\tau \sim \mathcal{U}([0,1])$, but rather obtained from $\tau \sim \text{Beta}(1.5, 1)$ defined on the $[0, s], s < 1$ support (Figure 37).

Using such Beta distribution emphasizes higher noise levels during training, a choice Black et al. (2024) argue allows $\pi_0$to focus on learning the mean of the data distribution $\mathbb{E}[a_{t:t+H_a}|o_t]$ during training, in keeping with Esser et al. (2024). To further optimize performance and reduce inference time, Black et al. (2024) propose reducing the support of the timestep distribution to $[0, s],\ s < 1$, as for any forward-integration step size $\delta = 1 - s$ timesteps above s are never sampled at inference time.

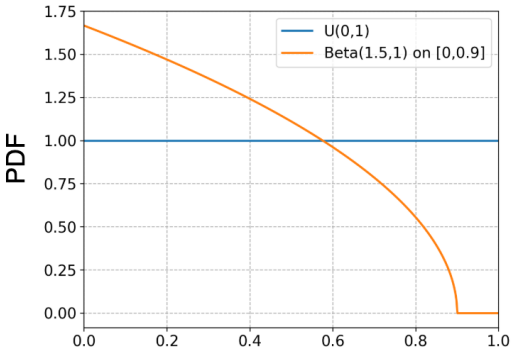


Figure 37 | Unlike more traditional flow-matching algorithms, $\pi_0$uses a modified distribution for the timestep τ used during training and inference, favouring earlier timestamps corresponding to noisier chunks.

Besides adopting a MoE architecture with a VLM backbone initialized from a pre-trained model and trained jointly with an action expert via flow matching, $\pi_0$also relies on a unique pre-training corpus mixes open data of 10M+ trajectories, which Black et al. (2024) claim to be the largest dataset used in building a foundational model in robotics to date. The dataset used to train π_0—referred to as π dataset—comprises a private, undisclosed portion obtained via teleoperation aggregated to openly available datasets including Open-X and DROID, with $\approx 9.1\%$ of the π being openly available. Open datasets such as DROID and Open-X are complemeneted with expert trajectories with of dexterous demonstrations tasks spanning 7 robot configurations and 68 different tasks. Black et al. (2024) show that pre-training on the π dataset yields a broadly capable base model, which can be adapted via post-training on narrower high-quality task data, inducing fluent multi-stage behavior while retaining robustness. In particular, Black et al. (2024) report that, across a variety of benchmarks, $\pi_0$pretrained on the π dataset and post-trained on extra high-quality data demonstrations *consistently outperform* $\pi_0$trained from scratch (i.e., without pretraining on the π dataset), further scoring the relevance of pretraining. Black et al. (2024) offer an intuition behind this finding: high-quality demonstrations of a given task typically do not contain mistakes, and how human demonstrator may recover from them. In turn, robot trained on high-quality data exclusively with BC may be incapable to recover from failure. Conversely, large scale

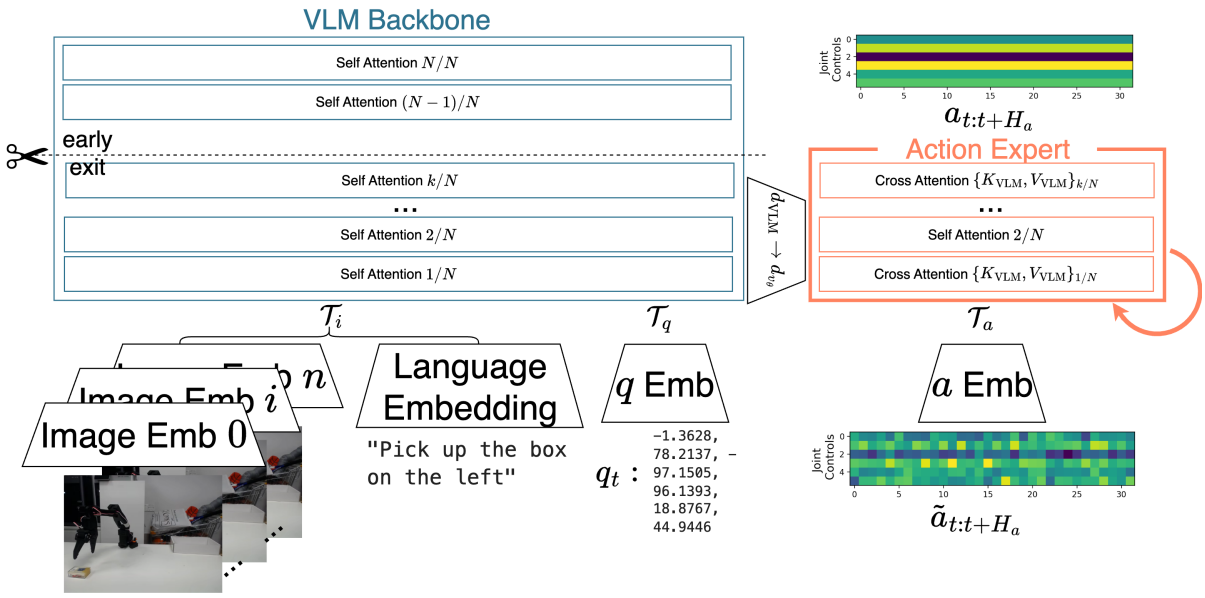


Figure 38 | The SmolVLA architecture, as in Shukor et al. (2025). SmolVLA is a compact MoE model trained with flow matching to denoise action chunks. Vision and language tokens are fed to a VLM backbone, and share information with the proprioperceptive and action tokens via the attention mechanism. The attention expert interleaves SA and CA layers for further conditioning on the visual features from the VLM backbone. SmolVLA skips computations and reduces the visual tokens, resulting in 6x less memory usage than π_0.

collections of human demonstrations are typically much more diverse (if anything, for their sheer scale), and therefore typically contain rich and diverse information, which may prove suboptimal for any given task when considered in isolation but that proves invaluable in coupling with a small, narrower set of demonstrations.

Lastly, Black et al. (2024) present cross-embodiment experiments where they demonstrate π_0's ability to control both mobile and static manipulator robots with varying arm embodiments. The emergence of cross-embodiment capabilities is largely to be attributed to the presence of large scale cross-embodiment data in the data mixture, handled by $\pi_0$defaulting to the maximal configuration size across the π dataset, and zero-padding robots with fewer dof. In that $\pi_0$constantly processes 18 DoFs robots (two 6-DoF arms, two grippers, base, vertical torso), regardless of the kind of robot, and robots with fewer dofs are zero-padded. $\pi_0$also relies on three camera views, and uses masked image slots for training and deployment scenarios with fewer cameras.

5.3.1 Code Example: Using π_0

add code example

5.4 SmolVLA

VLAs remain in an early stage of development and are not yet as mature or widely adopted as LLMs and VLMs. Further, much of the impactful VLA progress remains proprietary, with many models sharing only weights while withholding full training details and essential methodological components. SmolVLA (Shukor et al., 2025) is an entirely open-source research effort, aiming to democratize the developments of robotics foundation models by open sourcing model, training recipes and data used.

While encouraging efforts like π_0 (Black et al., 2024) demonstrate the feasibility of open VLA systems, they remain (1) large and compute-intensive and (2) dependent on closed datasets collected via centralized efforts on costly robotic platforms, ultimately hindering accessibility. SmolVLA mitigates both these accessibility issues by (1) prioritizing a compact, compute-efficient VLA design and (2) targeting community-contributed datasets on accessible robotic platforms such as the SO-100 and SO-101 arms. Similarly to π_0, SmolVLA (Figure 38) employs a MoE architecture combining a pretrained VLM backbone with a dedicated action expert, and trains with flow matching. To ensure efficiency and accessibility, SmolVLA adopts SmolVLM-2 (Marafioti et al., 2025) as its VLM backbone, considering SmolVLM-2's reduced size and capability to process multiple image inputs alongside text items. SmolVLM-2 uses SigLIP (Zhai et al., 2023) as vision encoder, producing visual features for a SmolLM2 language decoder (Allal et al., 2025). Further, SmolVLA adopts a smaller action expert consisting of ∼100M parameters and an interleaved stack

of self and cross-attention layers. To improve efficiency, the action expert adopts a reduced embedding dimension compared to the VLM backbone, resulting in $d_{v_\theta} = 0.75 d_{\text{VLM}}$. (Shukor et al., 2025)'s design choices thus result in a much smaller size model compared to π_0, consisting of around 450M parameters versus π_0's 3.3B parameters.

Effectively, SmolVLA consumes multi-view RGB images, a natural-language instruction, and a projected sensorimotor state token as inputs, together with the noised *action chunk* $a_{t:t+H_a}^{\sim}$ the action expert v_θ is trained to denoise. In particular, robot proprioperceptive states are projected into a shared token space with the VLM to match d_{VLM}, and successively projected into the expert's token space. Similarily to π_0, SmolVLA adopts separate experts communicating exclusively through self-attention layers, which do not employ the same blockwise causal masking in favour of a simple causal masking, resulting in a lower triangular attention mask.

In contrast with π_0, the action expert interleaves *cross-attention* (CA) and *self-attention* (SA) layers, a choice shown to yield higher success and smoother action chunks in practice. While in the expert SA layers, tokens are used to obtain queries, keys and values, CA layers use action tokens only as queries, and instead project visual, language and proprioperceptive tokens in a shared action space to obtain keys and values. Notably, keys and values can be cached as well, resulting in performance gains at inference time.

SmolVLA trims both token and layer compute. First, it *reduces visual tokens* via pixel shuffle to a fixed budget of 64 tokens per frame, foregoing tiling used during VLM pretraining for runtime efficiency. Second, it *skips upper VLM layers*: the action expert consumes features from the first N decoder layers, with $N = L/2$ providing a good speed-performance trade-off and effectively halving downstream compute for the larger part of SmolVLA. Beyond model compactness, SmolVLA also contributes an inference stack that decouples action prediction from execution for responsiveness on modest hardware (Section 4.4).

Departing from reliance on proprietary datasets, SmolVLA pretrains exclusively on 450+ *community datasets*, totaling 20K+ trajectories. Because instructions in community contributed dataset can be noisy or missing, the authors re-annotate tasks with a small off-the-shelf VLM using frames sampled from the dataset, and standardize camera viewpoints by mapping sources to a consistent top/wrist/side ordering. At inference, similarily to π_0, SmolVLA integrates flow over 10 steps, resulting in fast inference. SmolVLA proves effective across a range of both real-world and simulated environments, rivaling $\pi_0$while being close to 40% faster and consuming 6x less memory.

5.4.1 Code Example: Using SmolVLA

add code example

6 Conclusions

This tutorial has chronicled the paradigmatic shift transforming robotics, from the structured, model-based methods of its classical era to the dynamic, data-driven approaches that define modern robot learning. We began by examining the limitations of traditional dynamics-based control, highlighting the brittleness and the significant engineering overhead required by traditional approaches, which in turn motivates more flexible, less model-intensive learning approaches.

Our exploration of learning-based techniques revealed a clear trajectory of progress. We began with Reinforcement Learning, acknowledging its power to learn through interaction but also its real-world challenges, particularly sample inefficiency and the complexities of reward design. We saw how modern, data-driven approaches like HIL-SERL can make real-world RL feasible by incorporating human guidance and prior data. The inherent difficulties of RL, however, naturally motivated a deeper dive into imitation learning. This led us to single-task policies, where Behavioral Cloning, powered by advanced generative models like Action Chunking with Transformers and Diffusion Policy, demonstrated the ability to learn complex, multimodal behaviors directly from expert demonstrations. This laid the groundwork for the current frontier: the development of generalist, language-conditioned Vision-Language-Action models. Architectures like π_0 and SmolVLA—leveraging powerful pre-trained backbones and sophisticated generative modeling techniques like flow matching—represent a significant leap towards building foundational models for robotics that can generalize across varied tasks and embodiments.

A central theme throughout this work has been the critical role of openness in accelerating this progress. The recent explosion in capability is inseparable from the advent of large-scale, openly available datasets, the standardization of powerful and efficient model architectures, and the development of accessible, open-source software like `lerobot`. We argue the convergence towards an open approach to robotics is not merely a trend but a fundamental enabler, democratizing access to cutting-edge research in a traditionally siloed field like robotics.

We believe the path ahead for robot learning to be overly exciting, and filled with fundamental challenges we yet have to even scratch the surface of. The journey detailed in this tutorial, from the first principles to the state-of-the-art, equips researchers and practitioners alike with the context and the tools to chart their own journey in the future of robotics.

References

Open X-Embodiment: Robotic Learning Datasets and RT-X Models. https://robotics-transformer-x.github.io/.

Joshua Achiam. Spinning up in deep reinforcement learning. 2018.

AgiBot-World-Contributors, Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, Shu Jiang, Yuxin Jiang, Cheng Jing, Hongyang Li, Jialu Li, Chiming Liu, Yi Liu, Yuxiang Lu, Jianlan Luo, Ping Luo, Yao Mu, Yuehan Niu, Yixuan Pan, Jiangmiao Pang, Yu Qiao, Guanghui Ren, Cheng Ruan, Jiaqi Shan, Yongjian Shen, Chengshi Shi, Mingkang Shi, Modi Shi, Chonghao Sima, Jianheng Song, Huijie Wang, Wenhao Wang, Dafeng Wei, Chengen Xie, Guo Xu, Junchi Yan, Cunbiao Yang, Lei Yang, Shukai Yang, Maoqing Yao, Jia Zeng, Chi Zhang, Qinglin Zhang, Bin Zhao, Chengyue Zhao, Jiaqi Zhao, and Jianchao Zhu. AgiBot World Colosseo: A Large-scale Manipulation Platform for Scalable and Intelligent Embodied Systems, August 2025.

Pulkit Agrawal. Computational Sensorimotor Learning.

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.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: A Visual Language Model for Few-Shot Learning, November 2022.

Jorge Aldaco, Travis Armstrong, Robert Baruch, Jeff Bingham, Sanky Chan, Debidatta Dwibedi, Chelsea Finn, Pete Florence, Spencer Goodrich, Wayne Gramlich, Alexander Herzog, Jonathan Hoech, Thinh Nguyen, Ian Storz, Baruch Tabanpour, Jonathan Tompson, Ayzaan Wahid, Ted Wahrburg, Sichun Xu, Sergey Yaroshenko, and Tony Z Zhao. ALOHA 2: An Enhanced Low-Cost Hardware for Bimanual Teleoperation.

Mohammad Alizadeh and Zheng H. Zhu. A comprehensive survey of space robotic manipulators for on-orbit servicing. *Frontiers in Robotics and AI*, 11, October 2024. ISSN 2296-9144. doi: 10.3389/frobt.2024.1470950.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When Smol Goes Big – Data-Centric Training of a Small Language Model, February 2025.

Rika Antonova, Silvia Cruciani, Christian Smith, and Danica Kragic. Reinforcement Learning for Pivoting Task, March 2017.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report, 2025.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient Online Reinforcement Learning with Offline Data, May 2023.

Kostas E. Bekris, Joe Doerr, Patrick Meng, and Sumanth Tangirala. The State of Robot Motion Generation, October 2024.

Marc G. Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C. Machado, Subhodeep Moitra, Sameera S. Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836): 77–82, December 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-2939-8.

Richard Bellman. A Markovian Decision Process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957. ISSN 0095-9057.

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. π_0: A Vision-Language-Action Flow Model for General Robot Control, October 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control, July 2023a.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale, August 2023b.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners, July 2020.

Minwoo Byeon, Beomhee Park, Haecheon Kim, Sungjun Lee, Woonhyuk Baek, and Saehoon Kim. COYO-700M: Image-text pair dataset, 2022.

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, Siamak Shakeri, Mostafa Dehghani, Daniel Salz, Mario Lucic, Michael Tschannen, Arsha Nagrani, Hexiang Hu, Mandar Joshi, Bo Pang, Ceslee Montgomery, Paulina Pietrzyk, Marvin Ritter, A. J. Piergiovanni, Matthias Minderer, Filip Pavetic, Austin Waters, Gang Li, Ibrahim Alabdulmohsin, Lucas Beyer, Julien Amelot, Kenton Lee, Andreas Peter Steiner, Yang Li, Daniel Keysers, Anurag Arnab, Yuanzhong Xu, Keran Rong, Alexander Kolesnikov, Mojtaba

Seyedhosseini, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. PaLI-X: On Scaling up a Multilingual Vision and Language Model, May 2023.

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.

Open X.-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I. Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Muhammad Zubair Irshad, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J. Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R. Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Vitor Guizilini, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic Learning Datasets and RT-X Models, May 2025.

Jonathan H. Connell and Sridhar Mahadevan, editors. *Robot Learning*. Springer US, Boston, MA, 1993. ISBN 978-1-4613-6396-5 978-1-4615-3184-5. doi: 10.1007/978-1-4615-3184-5.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. In *Thirty-Seventh Conference on Neural Information Processing Systems*, 2023.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602 (7897):414–419, February 2022. ISSN 1476-4687. doi: 10.1038/s41586-021-04301-9.

J. Deng, K. Li, M. Do, H. Su, and L. Fei-Fei. Construction and analysis of a large scale image ontology. Vision Sciences Society, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An Embodied Multimodal Language Model, March 2023.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge Insulating Vision-Language-Action Models: Train Fast, Run Fast, Generalize Better, May 2025.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling Rectified Flow Transformers for High-Resolution Image Synthesis, March 2024.

William Fedus, Jeff Dean, and Barret Zoph. A Review of Sparse Expert Models in Deep Learning, September 2022.

Enrico Fini, Mustafa Shukor, Xiujun Li, Philipp Dufter, Michal Klein, David Haldimann, Sai Aitharaju, Victor Guilherme Turrisi da Costa, Louis Béthune, Zhe Gan, Alexander T. Toshev, Marcin Eichner, Moin Nabi, Yinfei Yang, Joshua M. Susskind, and Alaaeldin El-Nouby. Multimodal Autoregressive Pre-training of Large Vision Encoders, November 2024.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A. Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit Behavioral Cloning. In *Proceedings of the 5th Conference on Robot Learning*, pages 158–168. PMLR, January 2022.

Jun Fujita, Daisuke Soda, Chotaro Murata, and Hiroyuki Tsuhari. Development of Robots for Nuclear Power Plants and Their Application to New Fields. 57(4), 2020.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia

Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The Llama 3 Herd of Models, November 2024.

Robert J. Griffin, Georg Wiedebach, Sylvain Bertrand, Alexander Leonessa, and Jerry Pratt. Walking Stabilization Using Step Timing and Location Adjustment on the Humanoid Robot, Atlas. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 667–673, September 2017. doi: 10.1109/IROS.2017.8202223.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement Learning with Deep Energy-Based Policies, July 2017.

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.

Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin Riedmiller, and David Silver. Emergence of Locomotion Behaviours in Rich Environments, July 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. Beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, December 2020.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning, February 2022.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with Diffusion for Flexible Behavior Synthesis, December 2022.

Yandong Ji, Gabriel B. Margolis, and Pulkit Agrawal. DribbleBot: Dynamic Legged Manipulation in the Wild, April 2023.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian

Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B, October 2023.

Liyiming Ke, Jingqiang Wang, Tapomayukh Bhattacharjee, Byron Boots, and Siddhartha Srinivasa. Grasping with Chopsticks: Combating Covariate Shift in Model-free Imitation Learning for Fine Manipulation, November 2020.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R. Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J. Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. DROID: A Large-Scale In-The-Wild Robot Manipulation Dataset, April 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model, September 2024.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes, December 2022.

Rob Knight, Pepijn Kooijmans, Thomas Wolf, Simon Alibert, Michel Aractingi, Dana Aubakirova, Adil Zouitine, Russi Martino, Steven Palma, Caroline Pascal, and Remi Cadene. Standard Open SO-100 & SO-101 Arms.

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

Jing Yu Koh, Ruslan Salakhutdinov, and Daniel Fried. Grounding language models to images for multimodal inputs and outputs, 2023.

Zhifeng Kong, Arushi Goel, Rohan Badlani, Wei Ping, Rafael Valle, and Bryan Catanzaro. Audio flamingo: A novel audio language model with few-shot learning and dialogue abilities. In *International Conference on Machine Learning*, pages 25125–25148. PMLR, 2024.

Vik Korrapati. Moondream. Online, 2024.

Hugo Laurençon, Lucile Saulnier, Leo Tronchon, Stas Bekman, Amanpreet Singh, Anton Lozhkov, Thomas Wang, Siddharth Karamcheti, Alexander M Rush, Douwe Kiela, Matthieu Cord, and Victor Sanh. OBELICS: An open web-scale filtered dataset of interleaved image-text documents. In *Thirty-Seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. What matters when building vision-language models?, May 2024.

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.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, , Honolulu, Hawaii, USA,, 2023. JMLR.org.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, July 2019.

Ji Lin, Hongxu Yin, Wei Ping, Yao Lu, Pavlo Molchanov, Andrew Tao, Huizi Mao, Jan Kautz, Mohammad Shoeybi, and Song Han. VILA: On Pre-training for Visual Language Models, May 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling, February 2023.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow Matching Guide and Code, December 2024.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

Jiajun Liu, Yibing Wang, Hanghang Ma, Xiaoping Wu, Xiaoqi Ma, Xiaoming Wei, Jianbin Jiao, Enhua Wu, and Jie Hu. Kangaroo: A powerful video-language model supporting long-context video input. *arXiv preprint arXiv:2408.15542*, 2024.

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

Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. SERL: A Software Suite for Sample-Efficient Robotic Reinforcement Learning, March 2025.

Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 1 edition, May 2017. ISBN 978-1-316-66123-9 978-1-107-15630-2 978-1-316-60984-2. doi: 10.1017/9781316661239.

Oscar Mañas, Pau Rodriguez Lopez, Saba Ahmadi, Aida Nematzadeh, Yash Goyal, and Aishwarya Agrawal. MAPL: Parameter-efficient adaptation of unimodal pre-trained models for vision-language few-shot prompting. In Andreas Vlachos and Isabelle Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2523–2548, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.185.

Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Vaibhav Srivastav, Joshua Lochner, Hugo Larcher, Mathieu Morlon, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. SmolVLM: Redefining small and efficient multimodal models, April 2025.

Gabriel B. Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid Locomotion via Reinforcement Learning, May 2022.

John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks, September 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, December 2013.

Preetum Nakkiran, Arwen Bradley, Hattie Zhou, and Madhu Advani. Step-by-Step Diffusion: An Elementary Tutorial, June 2024.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision, February 2024.

Frank Permenter and Chenyang Yuan. Interpreting and Improving Diffusion Models from an Optimization Perspective, June 2024.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, David Yan, Dhruv Choudhary, Dingkang Wang, Geet Sethi, Guan Pang, Haoyu Ma, Ishan Misra, Ji Hou, Jialiang Wang, Kiran Jagadeesh, Kunpeng Li, Luxin Zhang, Mannat Singh, Mary Williamson, Matt Le, Matthew Yu, Mitesh Kumar Singh, Peizhao Zhang, Peter Vajda, Quentin Duval, Rohit Girdhar, Roshan Sumbaly, Sai Saketh Rambhatla, Sam Tsai, Samaneh Azadi, Samyak Datta, Sanyuan Chen, Sean Bell, Sharadh Ramaswamy, Shelly Sheynin, Siddharth Bhattacharya, Simran Motwani, Tao Xu, Tianhe Li, Tingbo Hou, Wei-Ning Hsu, Xi Yin, Xiaoliang Dai, Yaniv Taigman, Yaqiao Luo, Yen-Cheng Liu, Yi-Chiao Wu, Yue Zhao, Yuval Kirstain, Zecheng He, Zijian He, Albert Pumarola, Ali Thabet, Artsiom Sanakoyeu, Arun Mallya, Baishan Guo, Boris Araya, Breena Kerr, Carleigh Wood, Ce Liu, Cen Peng, Dimitry Vengertsev, Edgar Schonfeld, Elliot Blanchard, Felix Juefei-Xu, Fraylie Nord, Jeff Liang, John Hoffman, Jonas Kohler, Kaolin Fire, Karthik Sivakumar, Lawrence Chen, Licheng Yu, Luya Gao, Markos Georgopoulos, Rashel Moritz, Sara K. Sampson, Shikai Li, Simone Parmeggiani, Steve Fine, Tara Fowler, Vladan Petrovic, and Yuming Du. Movie Gen: A Cast of Media Foundation Models, February 2025.

Dean A. Pomerleau. ALVINN: An Autonomous Land Vehicle in a Neural Network. In *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

Simon J.D. Prince. *Understanding Deep Learning*. The MIT Press, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision, February 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, September 2023.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A Generalist Agent, November 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, March 2011.

Lindsay Sanneman, Christopher Fourie, and Julie A. Shah. The State of Industrial Robotics: Emerging Technologies, Challenges, and Key Research Directions, October 2020.

C Schuhmann, A Köpf, R Vencu, T Coombes, and R Beaumont. Laion coco: 600m synthetic captions from laion2b-en. *URL https://laion.ai/blog/laion-coco*, 2022.

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.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 1 edition, May 2014. ISBN 978-1-107-05713-5 978-1-107-29801-9. doi: 10.1017/CBO9781107298019.

Mustafa Shukor, Corentin Dancette, and Matthieu Cord. Ep-alm: Efficient perceptual augmentation of language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22056–22069, 2023.

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.

Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer Handbooks. Springer International Publishing, Cham, 2016. ISBN 978-3-319-32550-7 978-3-319-32552-1. doi: 10.1007/978-3-319-32552-1.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic Policy Gradient Algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, pages 387–395. PMLR, January 2014.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics, November 2015.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning Structured Output Representation using Deep Conditional Generative Models. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models, October 2022.

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.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains, June 2020.

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, September 2024.

Yang Tang, Chaoqiang Zhao, Jianrui Wang, Chongzhen Zhang, Qiyu Sun, Weixing Zheng, Wenli Du, Feng Qian, and Juergen Kurths. Perception and Navigation in Autonomous Systems in the Era of Learning: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):9604–9624, December 2023. ISSN 2162-237X, 2162-2388. doi: 10.1109/TNNLS.2022.3167688.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna

Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju-yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving Open Language Models at a Practical Size, August 2024.

Russ Tedrake. Robotic Manipulation. Perception, Planning and Control., a.

Russ Tedrake. Underactuated Robotics. Algorithms for Walking, Running, Swimming, Flying, and Manipulation, b.

Gabriele Tiboni, Karol Arndt, and Ville Kyrki. DROPO: Sim-to-Real Transfer with Offline Domain Randomization, January 2023.

Gabriele Tiboni, Pascal Klink, Jan Peters, Tatiana Tommasi, Carlo D'Eramo, and Georgia Chalvatzaki. Domain Randomization via Entropy Maximization, March 2024.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World, March 2017.

Peter Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Adithya Jairam Vedagiri IYER, Sai Charitha Akula, Shusheng Yang, Jihan Yang, Manoj Middepogu, Ziteng Wang, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Advances in Neural Information Processing Systems*, 37:87310–87356, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models, July 2023.

Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021.

Théophane Vallaeys, Mustafa Shukor, Matthieu Cord, and Jakob Verbeek. Improved baselines for data-efficient perceptual augmentation of llms. *arXiv preprint arXiv:2403.13499*, 2024.

Yi Wang, Xinhao Li, Ziang Yan, Yinan He, Jiashuo Yu, Xiangyu Zeng, Chenting Wang, Changlian Ma, Haian Huang, Jianfei Gao, et al. InternVideo2. 5: Empowering video mllms with long and rich context modeling. *arXiv preprint arXiv:2501.12386*, 2025.

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, Qianyu Chen, Huarong Zhou, Zhensheng Zou, Haoye Zhang, Shengding Hu, Zhi Zheng, Jie Zhou, Jie Cai, Xu Han, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. MiniCPM-v: A GPT-4V level MLLM on your phone, 2024.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid Loss for Language Image Pre-Training, September 2023.

Boqiang Zhang, Kehan Li, Zesen Cheng, Zhiqiang Hu, Yuqian Yuan, Guanzheng Chen, Sicong Leng, Yuming Jiang, Hang Zhang, Xin Li, et al. VideoLLaMA 3: Frontier multimodal foundation models for image and video understanding. *arXiv preprint arXiv:2501.13106*, 2025.

Chong Zhang, Wenli Xiao, Tairan He, and Guanya Shi. WoCoCo: Learning Whole-Body Humanoid Control with Sequential Contacts, November 2024.

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

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. MiniGPT-4: Enhancing vision-language understanding with advanced large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Wanrong Zhu, Jack Hessel, Anas Awadalla, Samir Yitzhak Gadre, Jesse Dodge, Alex Fang, Youngjae Yu, Ludwig Schmidt, William Yang Wang, and Yejin Choi. Multimodal C4: An open, billion-scale corpus of images interleaved with text. In *Thirty-Seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.