

THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

Optimal Transport Q-Learning for flow policy steering and acceleration

Citation for published version:
Sochopoulos, A, Whitammer, ES, Tsagkas, N, Moura, J, Gienger, M & Vijayakumar, S 2026, Optimal Transport Q-Learning for flow policy steering and acceleration. in *Proceedings of the 10th Conference on Robot Learning.* Proceedings of Machine Learning Research, PMLR, pp. 1-20, 10th Conference on Robot Learning, Austin, Texas, United States, 9/11/26. https://doi.org/10.48550/arXiv.2607.06262

Digital Object Identifier (DOI):
10.48550/arXiv.2607.06262

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Early version, also known as pre-print

Published In:
Proceedings of the 10th Conference on Robot Learning

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

OPEN ACCESS

Download date: 06. Nov. 2026

arXiv:2607.06262v1 [cs.RO] 7 Jul 2026

Optimal Transport Q-Learning for Flow Policy Steering and Acceleration

Andreas Sochopoulos[1,2], **Esmeralda S. Whitammer**[1], **Nikolaos Tsagkas**[1], **João Moura**[1], **Michael Gienger**[2], **Sethu Vijayakumar**[1]
[1]University of Edinburgh [2]Honda Research Institute Europe
ansocho.github.io/otql-flow

Abstract: Diffusion and flow policies have recently demonstrated remarkable performance in robotic applications by accurately capturing multimodal robot trajectory distributions, especially in the context of vision language action (VLA) models. However, high quality policy performance also requires fast inference and high quality demonstrations, which are often hard to get. Lack of these leads to suboptimal policy behaviors and failure under distribution shifts. In this work we address the problem of fine-tuning and accelerating suboptimal flow-based policies using the robot's experience through RL post-training. We introduce *Optimal Transport Q-Learning* (OTQL), a new methodology for finetuning flow policies using advantage weighted conditional optimal transport flow matching. OTQL can finetune and accelerate flows with an interaction budget of 50-60 episodes while avoiding computationally expensive distillation in simulation and real-world robot tasks. Our results show that OTQL post-trains flow policies using the robot's own experience, increasing average success percentage of single-task policies from 36% to 86% and of a pre-trained VLA from 38% to 76% while reducing the number of inference steps per action generation by 70%.

Keywords: Flow Matching, Reinforcement Learning, Optimal Transport

1 Introduction

Vision language action models (VLAs) and, more recently, world action models (WAMs) have brought the field of robotics one step closer to generalist policies. These systems use as backbone a large, pre-trained vision language model (VLM) [1, 2] or world model (WM), often in the form of a video model [3, 4], that provides features to an action head. This module is most commonly a flow or diffusion model [4, 3, 5, 6, 7, 8]. Flows trained with flow matching (FM) [9, 10] and diffusion models [11, 12] have proven highly effective for action generation, as they can capture complex, multimodal action distributions [13, 14], such as those demonstrated by humans during teleoperation or kinesthetic teaching.

Despite undergoing large-scale pretraining, VLAs and WAMs often struggle with zero-shot generalization to tasks unseen during training. Moreover, single and multi-task flow and diffusion policies frequently fail to handle out-of-distribution environment states, leading to task failures. The standard remedy for these issues is collecting additional demonstrations for the target tasks or states, and performing supervised fine-tuning (SFT) on the newly acquired dataset. However, collecting data through teleoperation is tedious and requires significant human effort, time, and expertise.

Reinforcement learning (RL) provides the tools for policies to adapt using the robot's own experience, eliminating the need for human demonstrations [15, 16, 17, 18, 19]. A large body of work has attempted to train flow and diffusion policies in *offline RL* settings using a static set of rollouts [20, 21, 22, 23, 24, 25]. Although successful in simple to moderate robotic tasks, most of these methods suffer from *slow inference* because they require integrating an SDE or ODE over multiple steps [23]. Furthermore, since critic training typically involves generating actions from the learned policy, this slow inference propagates to every training step, resulting in *slow training*. While there

have been attempts to train single-step policies [25, 21], they rely on computationally expensive distillation. In the context of behavior cloning, there has been extensive work on diffusion/flow acceleration [26, 27, 28, 29, 30, 31], but methods that address both acceleration and RL post-training remain scarce.

In this paper, we present *Optimal Transport Q-Learning (OTQL)*, an FM methodology that *steers* flow-based policies to achieve *few-step inference* without expensive training. Our method uses conditional optimal transport (COT) couplings [28, 32, 33] between noise and action samples to train flows with straight integration paths, avoiding the computational complexity of common inference acceleration techniques [34, 35, 36, 25, 37, 38]. Furthermore, we train a critic function and use it to inform the marginal probabilities of the action distribution while solving the COT problem. This results in couplings that favor high-value samples during training, leading to flows that learn straight paths between the noise distribution and these optimal samples.

Flows steered with OTQL demonstrate strong empirical performance in both offline and online fine-tuning scenarios, requiring only two to three integration steps for inference. More specifically, we show that OTQL achieves fine-tuned performance comparable to state-of-the-art steering methods [16, 23, 22] in both real and simulated tasks, but at a fraction of the training and inference time. We also demonstrate OTQL's ability to steer VLAs (*e.g.*, SmolVLA [39]) on novel tasks using the robot's own experience. In real-world tasks, our approach increases SmolVLA's performance from 38% to 76% while reducing the neural function evaluations (NFEs) required for inference by 70%.

2 Background

Offline and online RL. We consider a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \rho_0, \gamma, P)$, where $\mathcal{S}$ is the state space, $\mathcal{A} \in \mathbb{R}^d$ the action space, $r(s_t, a_t)$ the reward function, $\rho_0(s_0) \in \Delta(\mathcal{S})$ the initial state distribution, $\gamma \in [0, 1)$ the discount factor, and $P(s_{t+1} \mid s_t, a_t)$ the transition dynamics. By $\Delta(\mathcal{X})$ we denote the set of probability distributions defined over the set $\mathcal{X}$. Assuming a policy $\pi(a \mid s) : \mathcal{S} \to \Delta(\mathcal{A})$ that maps states to actions we define the critic over π as $Q_\pi(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s_t, a_0 = a_t\right]$. The goal of online RL is to find a policy π that maximizes the expected discounted return $\mathbb{E}_{s_{t+1} \sim P(\cdot|s_t, a_t), a_t \sim \pi(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$ with environment interactions, while the goal of offline RL is to find a policy that maximizes the expected discounted return using only an offline dataset $\mathcal{D}$ of transitions (s, a, r, s').

Imitation learning as conditional generative modeling. In imitation learning (IL), we typically assume access to a dataset of expert demonstrations $\mathcal{D} = \{(s^{(i)}, a^{(i)})\}_{i=1}^n$, which contains observation-action sequence pairs. Given the dataset $\mathcal{D}$ we treat the learning of actions a as a conditional generative modeling problem conditioned on s. We treat the $a^{(i)}$ as samples from a target distribution $p_a(a \mid s)$ with conditions $s = s^{(i)}$. In the context of generative modeling with continuous normalizing flows, or neural ODEs [40], we are searching for an ODE that transforms an initial noise distribution p_z over $\mathbb{R}^d$ into the conditional action distribution $p_a(a \mid s)$. Given an ODE $dx = v_\theta(t, x \mid s)\, dt$, where $v_\theta : \mathbb{R} \times \mathbb{R}^d \times \mathcal{S} \to \mathbb{R}^d$ is a vector field parametrized by a neural network with parameters θ taking x, s and t as input, one aims to find θ such that if $x(0) = z \sim p_z$, then the distribution over $x(1)$ induced by integration of the ODE from $t = 0$ to $t = 1$ with initial conditions $x(0)$ matches p_a.

RL post-training. Assuming we are given a policy that samples from a conditional distribution $\pi(a \mid s)$ approximating a distribution $p_a(a \mid s)$, we are interested in sampling from $\tilde{\pi}(a \mid s) \propto \pi(a \mid s) e^{\lambda Q(s,a)}$ or equivalently $\tilde{\pi}(a \mid s) \propto \pi(a \mid s) e^{\lambda A(s,a)}$, where A is the advantage function $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$ and $V(s_t) = \mathbb{E}_{s_{t+1} \sim P(s_t, a_t), a_t \sim \pi}[Q(s_t, a_t)]$ is the state-value function. Our motivation for attempting to sample $\tilde{\pi}$ is grounded on the fact that it is the closed-form solution to the KL regularized problem as defined in [41, 42, 43]. Intuitively, we are trying to *steer* the base policy towards behaviors that it already models and that maximize the Q value.

Conditional flow matching. In this paper we focus on policies instantiated as neural ODE-based generative models which are trained with CFM. To implement CFM, one must specify a *coupling* distribution $q(z, a \mid s)$ whose marginals equal p_z and p_a and an interpolating trajectory that links

each pair (z, a). This trajectory is governed by an ordinary differential equation (ODE), $dx = u(t, x \mid z, a)\, dt$, constrained such that the state flows from $x(0) = z$ to $x(1) = a$. Throughout this work, we restrict our focus to a linear interpolant, $u(t, x \mid z, a) = a - z$. Consequently, the state evolves as $x(t) = ta + (1-t)z,\ t \in [0, 1]$. Under this linear formulation, the neural network v_θ is optimized via the following stochastic regression loss:

$$\mathcal{L}_{\text{CFM}}(\theta) := \mathop{\mathbb{E}}_{\substack{t\sim\mathcal{U}([0,1]) \\ z,a\sim q(z,a\mid s)}} \Big\| \underbrace{v_\theta(t, \overbrace{ta + (1-t)z}^{\text{interpolant}} \mid s)}_{\text{learned vector field}} - \underbrace{(a - z)}_{\text{target}} \Big\|^2. \tag{1}$$

The primary theoretical justification for this objective is that the optimal time-dependent vector field v_θ minimizing (1) successfully pushes the base distribution p_z to the target distribution p_a at $t = 1$. This effectively resolves the generative modeling task defined earlier.

Conditional optimal transport flow matching. Different choices exist for the coupling $q(z, a)$. In Independent Coupling CFM $q(z, a) = q(z)q(a)$. Another option, which was shown in [10, 44] to reduce objective variance and straighten integration curves in the unconditional case, takes q to be a minibatch OT plan computed from batches of samples z, a. Such a coupling approximates the OT plan between p_z and p_a. A similar idea has also been explored in the case of conditional distributions [32, 28, 33] using minibatch OT to approximate a conditional OT (COT) plan. For more details on the basic definitions of OT and COT see [10] and §A.

Describing the OT plan as the solution of an ODE that moves particles from z to a over time yields the dynamic formulation of the OT problem, commonly known as the Benamou–Brenier formulation [45]. The dynamic OT plan transports particles along straight-line trajectories from z to a and when $q(z, a)$ denotes the 2-Wasserstein OT coupling (see §A) , the vector field that minimizes the unconditional version of the CFM objective in (1) coincides exactly with the dynamic OT ODE [10]. For a more detailed discussion, see Tong et al. [10]. Kerrigan et al. [32] have extended these results to the dynamic formulation of COT.

3 Optimal Transport Q-Learning

In this section we present OTQL, our method for learning flow policies that are able to sample actions from $\tilde{\pi}(a \mid s)$. In §3.1 we present the CFM training loss that enables OTQL and in §3.2 we detail the full algorithm.

3.1 Conditional optimal transport for flow acceleration and steering

Motivated by the observations made in §2 connecting OT and CFM, our goal is to learn a flow that approximates the dynamic COT ODE between the noise distribution and $\tilde{\pi}$. We initially consider a general version of the form $\tilde{\pi}(a \mid c) \propto \pi(a \mid c)e^{-\lambda\mathcal{E}(a,c)}$, where c is a general condition vector and $\mathcal{E} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a positive energy function. To learn a flow that samples from $\tilde{\pi}$, we employ an approximate COT plan as the coupling $q(z, \tilde{a} \mid c)$, where $\tilde{a} \sim \tilde{\pi}$, between the two distributions. The resulting flow is intended to approximate the COT ODE linking the noise distribution and samples from $\tilde{\pi}$, thereby producing straighter trajectories towards low-energy samples.

Unconditional OT couplings In practice, given m samples from the dataset $\{(a_j, c_j)\}_{j=1}^m \sim \mathcal{D}$, where c_j are conditions, and m noise samples $\{z_i\}_{i=1}^m \sim \mathcal{N}(0, I_d)$ we compute a mini-batch approximation of the unconditional OT plan by solving the following optimization problem between the measures $\mathbf{z} = \sum_{i=0}^m k_i \delta_{z_i}$ and $\mathbf{a} = \sum_{j=0}^m p_i \delta_{a_j}$ (where typically $k_i = p_i = \frac{1}{m}$):

$$T_\varepsilon(\mathbf{z}, \mathbf{a}) \ = \ \min_{\Gamma \in \Gamma(\mathbf{z}, \mathbf{a})} \sum_{i,j} C_{ij}\, \Gamma_{ij} - \varepsilon\, H(\Gamma), \quad s.t. \ \sum_j \Gamma_{ij} = k_i, \ \sum_i \Gamma_{ij} = p_j \tag{2}$$

where $C_{ij} = \|x_i - x_j\|^2$ is the squared-Euclidean cost, $H(\Gamma) = -\sum_{i,j} \Gamma_{ij} \log(\Gamma_{ij} - 1)$ is the entropy regulariser, $\varepsilon > 0$ is the regularisation strength, and $\Gamma(\mathbf{z}, \mathbf{a})$ is the set of joint distributions (transport plans) with marginals $\mathbf{z}$ and $\mathbf{a}$.

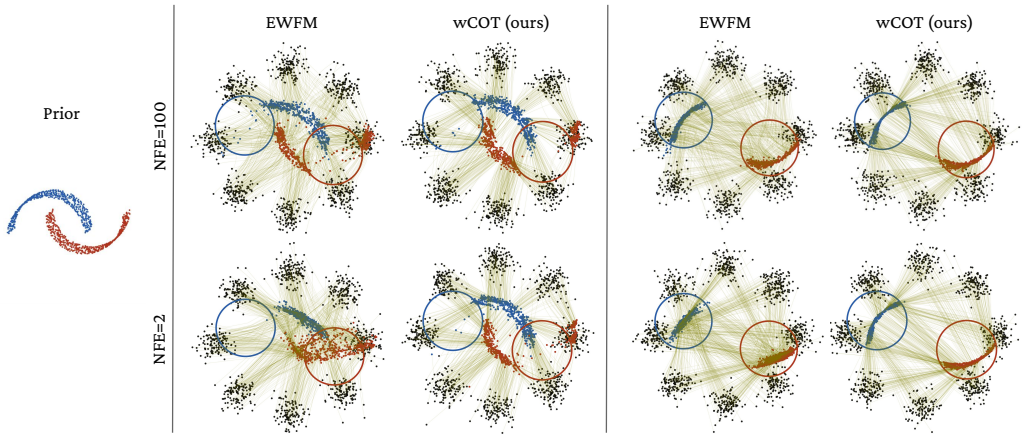


Figure 1: Energy-weighted Flow Matching (EWFM) [22] and weighted-marginal COT flow matching (wCOT) flows trained to generate the tilted conditional two moons distribution from the 8 Gaussians distribution. Generation with 100 (top row) and 2 (bottom row) `euler` integration steps is shown. The samples generated from both methods given an energy function that is 1 inside the drawn circles and 0 everywhere else is shown in the **left** and given an energy function that is 1 outside and 0 inside the circles in the **right**.

Conditioning via cost augmentation. Since we target *conditional* action distributions (*i.e.*action distributions conditioned on robot observations), we also wish the coupling to respect the condition structure: source and target points with the same condition should be preferentially paired, so the triangular constraint of COT [32] is satisfied. Similar to [28], we incorporate the condition directly into the cost by concatenating it with the data vector before solving (2). Concretely, we form augmented vectors

$$\bar{z}_i = \left[z_i \parallel \alpha \bar{c}_i\right], \quad \bar{a}_j = \left[a_j \parallel \alpha c_j\right], \tag{3}$$

where $\alpha > 0$ is a scaling hyperparameter controlling the relative influence of the condition on the coupling, and $\bar{c}_i$ are the elements of the source condition vector, the choice of which will be explained in the next paragraph. The augmented cost becomes:

$$C_{ij} = \left\|\bar{z}_i - \bar{a}_j\right\|^2 = \|z_i - a_j\|^2 + \alpha^2 \|\bar{c}_i - c_j\|^2. \tag{4}$$

The second term penalizes pairings whose conditions differ, steering the solver toward condition-consistent couplings.

Energy-weighted target marginal and source condition sampling. To obtain an approximate COT coupling between the noise and target $\tilde{\pi}$, we re-weight the target measure using the energy function $\mathcal{E}$. For each target sample we define unnormalized weights $w_j = e^{-\lambda \mathcal{E}(a_j, c_j)}$, which we use to transform the target probability mass function used in the COT problem to

$$\bar{\mathbf{a}} = \sum_{j=0}^{m} p_j \delta_{(a_j, c_j)}, \quad p_j = \frac{w_j}{\sum_{k=1}^{m} w_k}, \tag{5}$$

placing more mass on low-energy (high-reward) samples. To approximate the COT plan, the source and target measures should share the same condition marginals [32]. Since the weighted target has condition marginal $\sum_j p_j \delta_{c_j}$, the source conditions $\bar{c}_i$ used in (3) must be drawn from this same tilted distribution. We therefore sample each source condition independently as $\bar{c}_i \sim \sum_{j=1}^{m} p_j \delta_{c_j}$. Combining the source conditions with (3) and (5) we get the following measures $\bar{\mathbf{z}} = \sum_{i=0}^{m} \frac{1}{m} \delta_{\bar{z}_i}$ and $\bar{\mathbf{a}} = \sum_{j=0}^{m} p_j \delta_{\bar{a}_j}$.

Putting everything together, the coupling used for training is obtained by solving the entropic OT problem $T_\varepsilon(\bar{\mathbf{z}}, \bar{\mathbf{a}})$ at each mini-batch. Eventually, at each training step we sample noise-action pairs from the optimal coupling Γ^* that minimizes $T_\varepsilon(\bar{\mathbf{z}}, \bar{\mathbf{a}})$ and use (1) to train a flow. We term this loss *wCOT-CFM*. In Fig. 1 we can see that flows trained using wCOT-CFM successfully sample high energy samples with near straight lines and good low-NFE accuracy compared to an energy weighted flow matching approach [22].

3.2 Policy post-training using wCOT-CFM

We next apply wCOT-CFM to the problem of RL post-training. We assume access to a pre-trained flow policy $v_{\bar{\theta}}$ that samples from a distribution $\pi_{\bar{\theta}}$ and we train it using wCOT-CFM to extract a new policy that generates samples from $\pi_\theta(a \mid s) \propto \pi_{\bar{\theta}}(a \mid s) e^{\lambda A(a,s)}$.

Training the critic. We assume access to a buffer $\mathcal{B}$ that may contain transitions from an expert or semi-expert offline dataset $\mathcal{D}_{\text{off}}$ that was used to train the base flow $v_{\bar{\theta}}$ and policy rollouts using v_θ. We train a critic $Q_\phi(s,a)$ using a Temporal Difference (TD) loss:

$$L_Q(\phi) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{B},\ a_{t+1}\sim\pi_\theta(\cdot|s_{t+1})}\left[\left(Q_\phi(s_t,a_t) - (r_t + \gamma Q_{\bar{\phi}}(s_{t+1},a_{t+1}))\right)^2\right] \tag{6}$$

where $a_{t+1} \sim \pi_\theta(\cdot \mid s_{t+1})$ implies $a_{t+1} \leftarrow \text{ODE}(v_\theta, s_{t+1}, z)$ which is the `euler` integration of the flow v_θ with condition s_{t+1} and source noise sample $z \sim \mathcal{N}(0, I_d)$. The target $y = r + \gamma Q_{\bar{\phi}}(s_{t+1}, a_{t+1})$ is the Bellman target commonly used in RL [25, 21, 16] and $Q_{\bar{\phi}}$ is a target network whose parameters are updated using an exponential moving average filter.

The actions a_t are typically single-step actions; however they can also be thought of as action chunks $a_t = (a_0^t, a_1^t, \ldots, a_{H-1}^t)$ of horizon H where a_i^t is the action at $t = t + i$ generated as part of an action chunk at time t [23]. Then we can simply substitute the instantaneous reward r_t in (6) with the discounted sum of rewards over the chunk $\sum_{t'=t}^{t+H-1} \gamma^{t'-t} r_{t'}$ making the bellman target an unbiased value function approximator over action chunks. This is important as most pretrained diffusion and flow policies usually operate on chunked actions [13, 46, 8, 39, 4] and we therefore adopt the chunked Q function and TD loss throughout this work.

Algorithm 1 Optimal Transport Q-Learning (OTQL)

Input: Base policy $v_{\bar{\theta}}$, Buffer $\mathcal{B}$, Energy scale $\lambda > 0$, Target EMA $\tau \in (0,1)$
Initialize: $v_\theta \leftarrow v_{\bar{\theta}}, \quad Q_\phi, \quad Q_{\bar{\phi}} \leftarrow Q_\phi$
Populate $\mathcal{B}$ via initial environment rollouts using $v_{\bar{\theta}}$
for each training iteration **do**
  Sample mini-batch $\{(s_j, a_j, r_j, s'_j)\}_{j=1}^m \sim \mathcal{B}$
  ▷ Critic Q_ϕ update
  $z \sim \mathcal{N}(0, I_d), \quad a'_j \leftarrow \text{ODE}(v_\theta, s'_j, z)$
  Update ϕ: $\sum_{j=1}^m \left(Q_\phi(s_j,a_j) - r_j + \gamma Q_{\bar{\phi}}(s'_j, a'_j)\right)^2$
  $\bar{\phi} \leftarrow \tau\phi + (1-\tau)\bar{\phi}$
  ▷ Flow v_θ update using wCOT-CFM and $A(s,a)$
  $z_k \sim \mathcal{N}(0, I_d), \quad \hat{a}_j^k \leftarrow \text{ODE}(v_\theta, s'_j, z_k)$
  $A_j(s_j,a_j) \leftarrow Q_\phi(s_j,a_j) - \sum_k Q_\phi(s_j, \hat{a}_j^k)$
  $p_j \leftarrow \frac{e^{\lambda A_j}}{\sum_k e^{\lambda A_k}}$
  $\bar{\mathbf{a}} \leftarrow \sum_{j=1}^m p_j \delta_{\bar{a}_j}, \quad$ with $\bar{a}_j = [a_j \parallel \alpha s_j]$
  Sample $z_i \sim \mathcal{N}(0, I_d)$ and $\bar{c}_i \sim \sum_{j=1}^m p_j \delta_{c_j}$
  $\bar{\mathbf{z}} \leftarrow \frac{1}{m}\sum_{i=1}^m \delta_{\bar{z}_i}, \quad$ with $\bar{z}_i = [z_i \parallel \alpha \bar{c}_i]$
  Sample $(z,a) \sim \Gamma^*$, where $\Gamma^* \leftarrow \arg\min T_\varepsilon(\bar{\mathbf{z}}, \bar{\mathbf{a}})$
  Update θ: $\mathcal{L}_{\text{CFM}}(\theta)$
  ▷ Online Data Collection
  if online environment is available **then**
    Rollout using v_θ and add new transitions to the buffer $(s,a,r,s') \rightarrow \mathcal{B}$

Training the flow policy. Given the base policy $v_{\bar{\theta}}$ we first initialize v_θ with $\theta = \bar{\theta}$. Before starting training of the policy and the critic we rollout the base policy in the environment for a small number of episodes and populate the buffer $\mathcal{B}$ (or augment it if it contains $\mathcal{D}_{\text{off}}$) with the transitions collected. We then sample a batch of state-action-reward transitions and compute the advantage function $A(s_t,a_t) = Q(s_t,a_t) - \hat{V}(s_t)$, where $\hat{V}(s_t)$ is a value function approximation calculated as $\hat{V}(s_t) = \mathbb{E}_{a\sim\pi_\theta(\cdot|s_t)}\left[Q(s_t,a)\right]$ [43]. Lastly, we use the advantage values as a negative energy and calculate the wCOT coupling which is used in $\mathcal{L}_{\text{CFM}}$ to update θ. The full training pipeline can be seen in Algorithm 1.

Practical implementations. wCOT-CFM trains an energy-informed flow by modifying the coupling used in (1). Sampling noise-action pairs from the optimal coupling Γ^* will possibly lead to some actions in the batch to be dropped and others to be duplicated. When the Q function has sharp peaks, sampling from the optimal coupling may significantly reduce the effective batch size and increase the loss variance. Therefore, we make two practical modifications to overcome this challenge: 1) we clamp the weights $w_j = \min(e^{\lambda\mathcal{E}}, w_{max})$ in (5) and 2) at inference time and for the Bellman target computation, we generate N candidate actions from the flow and select the one with the highest Q-value [47, 23]. This is equivalent to approximately sampling from a sharpened version of the policy π_θ, and can be seen as a complementary mechanism to the OT-based steering: while wCOT-CFM

	NFE ≥ 10							NFE = 1		NFE = 2
Environment	FAWAC	DSRL	QSM	CGQL	FEdit	QAM	IFQL	FQL	QAM-F	OTQL
antmaze-large	0.17	0.61	**0.9**	0.76	0.58	0.81	0.36	0.76	0.83	0.88
antmaze-giant	0	0.03	**0.24**	0	0.02	0.18	0.01	0	0.12	0.19
humanoidmaze-medium	0.24	0.53	0.82	0.6	0.22	0.67	**0.86**	0.68	0.65	0.82
humanoidmaze-large	0	0.03	0.06	0.05	0.03	0.11	**0.24**	0.09	0.12	0.19
cube-double	0.02	**0.74**	0.33	0.38	0.4	0.64	0.11	0.46	0.65	0.68
cube-triple	0	0.01	0.06	**0.08**	0.02	0.03	0	0.03	0.03	0.04
scene	0.38	**0.99**	0.78	0.38	0.62	0.97	0.84	0.78	0.95	**0.99**
puzzle-3x3	0.03	0.87	0.57	0.48	**0.99**	**1**	**1**	0.7	**0.99**	**0.99**
Average	0.11	0.48	0.47	0.34	0.36	0.55	0.43	0.44	0.54	**0.59**

Table 1: Offline RL results on OGBench. Each environment has 5 tasks associated with it.

shapes the flow's trajectory distribution toward high-advantage regions, rejection sampling provides a lightweight refinement that further concentrates samples near the mode of the advantage-weighted distribution. We find that N=5 candidates suffice to obtain meaningful gains, adding only negligible computational overhead relative to the ODE integration cost. Although we employ weight clamping for all tasks, we only use rejection sampling in scenarios with relatively large chunk sizes and peaked Q functions. See §C for an ablation of the most important OTQL hyper-parameters.

4 Experiments

We evaluate the ability of OTQL to train flow policies in offline, online, and offline-online RL settings across a diverse set of 42 simulation tasks and 4 real-world tasks. For the simulations, we utilize the OGBench benchmark [48] for offline and offline-online experiments, alongside two MuJoCo tasks [49] from the LeRobot codebase [50] for online post-training. We compare OTQL against several baselines: DSRL [16], which has proven highly effective in robotics; Q-Chunking (QC) [23], which shares OTQL's chunking and rejection sampling mechanisms; and the fast flow-based methods FQL [25] and QAM-F [24]. We also evaluate Energy Weighted Flow Matching (EWFM) [22], a method that, like OTQL, connects to the importance sampling framework. Some offline RL baselines are diffusion-based (QSM [51]) or flow adaptations of diffusion-based methods (IFQL [52]). FAWAC, CGQL, and FEdit are baselines considered and detailed in [24].

4.1 Offline RL in simulation

We first test the ability of OTQL to train flow policies in offline settings, on the OGBench benchmark [48]. For all tasks, a dataset of suboptimal and mixed-quality demonstrations is provided along with sparse rewards (0 if an action solves the tasks, -1 otherwise) and the goal is to extract meaningful behaviors for solving the task from it. The main results across 40 tasks and 9 baselines are reported in Table 1, where the methods are grouped based on the NFE required to sample one action from the flow or diffusion policy. We follow the same experiment setup as in [24] and utilize action chunking for the tasks `cube-double, cube-triple, scene` and `puzzle`. For OTQL we also employ rejection sampling only in the aforementioned tasks with $N = 5$.

From Table 1 we observe that OTQL performs equally well and in some tasks even outperforms state-of-the-art methods that require multiple NFE for inference. Overall, OTQL achieves the highest success rate on average and outperforms fast inference methods like FQL and QAM-F by at least 5%. This showcases the ability of OTQL to train policies from scratch given suboptimal demonstrations, while maintaining competitive inference speed. Offline pretraining can be very effective when training policies from data sources that contain demonstrations of mixed quality [53] or when using the robot's failed rollouts [46]. We utilize OTQL in an offline setting extensively in the real-robot tasks (§4.3).

4.2 Offline-online and online RL in simulation

We further explore the ability of OTQL to pre-train a policy using offline RL and then finetune it using online RL. Additionally, we test the online adaption capabilities of OTQL when provided with a pre-trained policy that has been trained on a limited amount of demonstrations.

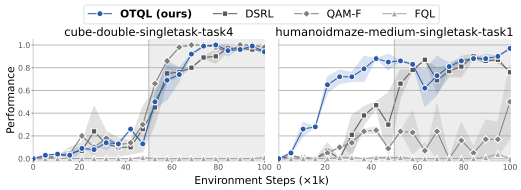


Figure 2: OTQL is effective in offline-online RL settings

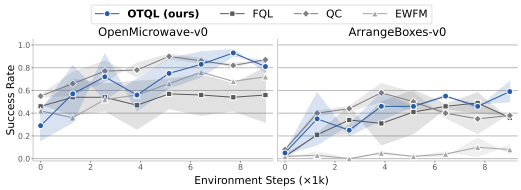


Figure 3: OTQL enables adaptation and acceleration in online RL settings

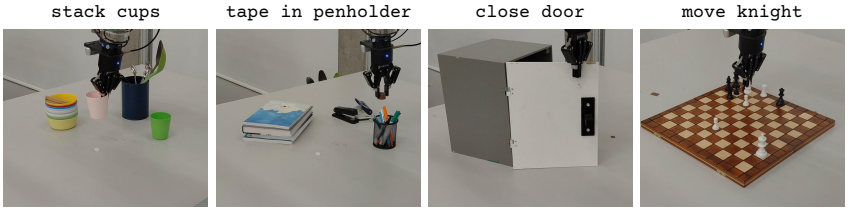


Figure 4: real-wrold tasks used for the evaluation of OTQL

Offline-to-online adaptation. We test offline-to-online adaptation in two OGBench environments, one in which we use action chunking (`cube-double`) and one in which no action chunking is employed (`humanoidmaze-medium`). DSRL, FQL and QAM-F are used as baselines as they are among the best performing methods in the offline RL experiments and FQL and QAM-F explicitly accelerate inference. All policies are pre-trained for 500k steps and then finetuned with online RL for 500k steps with an update-to-data ratio (UTD) of 1. The training curves are shown in Fig. 2. We observe than in the `cube` task where action chunking is employed all methods have similar performance, with a clear performance gain when switching from offline to online RL. However, in the `humanoidmaze` task we observe that OTQL is the only method reaching near 90% with offline pre-training, while DSRL is the only baseline matching OTQL's performance during online training. FQL and QAM-F, which have similar inference time to OTQL, fail to surpass a success rate of 50%.

Online finetuning. We further test if OTQL can steer flow policies pre-trained on a limited amount of demonstrations. More specifically, we evaluate OTQL, QC, FQL and EWFM on two Mujoco tasks, namely `OpenMicrowave-v0` and `ArrangeBoxes-v0` (see §D). We use action chunking for all methods with a chunk size of 10. The pre-trained policy is first rolled-out in the environment for 2000 steps and the transitions are added to the replay buffer $\mathcal{B}$ alongside the pre-training demonstrations $\mathcal{D}_{\text{off}}$, similar to [16]. We then train the policies for a total of 8000 steps using UTD = 10. For OTQL, we only use rejection sampling on `ArrangeBoxes-v0` with $N = 5$ candidates. Overall, OTQL provides the most effective online adaptation. Compared to high-NFE baselines, OTQL (NFE = 3, $N = 5$) outperforms EWFM (NFE = 10, $N = 1$) and performs as well as QC (NFE = 10, $N = 32$) while requiring only a fraction of the inference time. At the same time, it maintains a significant performance edge over other fast methods, achieving a maximum success rate that is on average 15% higher than FQL. While the randomized box locations in `ArrangeBoxes-v0` limit the absolute gains of all methods, OTQL consistently delivers the best combination of task success and inference speed.

4.3 Policy steering and acceleration in real-world tasks

We next evaluate OTQL's ability to fine-tune and accelerate both single-task and flow-based VLA policies under challenging real-world conditions. We evaluate our method across a suite of four tasks (Fig. 4), selected to represent varied manipulation paradigms. By spanning stacking, accurate manipulation, contact-rich pushing, and precise pick-and-place maneuvers, we aim to demonstrate OTQL's robustness across diverse skills and precision requirements. All evaluations operate under a maximum episode length of 150 steps with sparse rewards ($r_t = 0$ if the policy solves the task at timestep t, and $r_t = -1$ otherwise [46])

OTQL for real-world policies. To test how well OTQL can adapt single-task policies, we train a base flow policy on each of the first three tasks in Fig. 4, using CFM

with 10 demonstrations. The base policy typically achieves low performance ($\leq$ 50%) as the tasks require precise behaviors to be solved. Instead of doing online RL, we choose to adapt the policies by training in a semi-offline setting every 10 demonstrations. We start by collecting 10 rollouts from the base policy and train using offline RL on the expert demonstrations and the collected data for 20k steps. We then iteratively rollout the fine-tuned policy for 10 episodes and train with OTQL for 20k steps until the buffer contains a total of 50 episodes. We found that this approach significantly reduced the amount of time a human needed to supervise the robot, as no training was performed during the rollouts. Also this approach enables the policy to be deployed in higher frequencies, allowing adaptation in tasks that require fast motions and responsiveness. More details on the experiments setup can be found in §D. Overall, we observe from Fig. 5 that OTQL is able to adapt the base policies by increasing average success rates from 36% to 86%, while reducing the NFE needed for inference from 10 to only 3, offering a significant inference speed-up.

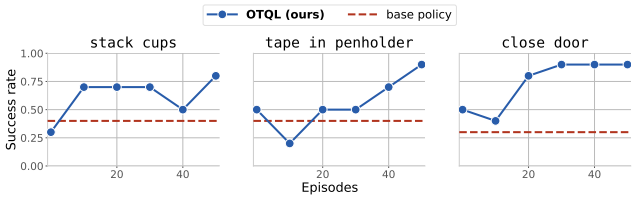


Figure 5: OTQL adaptation of base flow policies on 3 tasks.

OTQL for VLAs. Finally, we highlight the efficacy of our approach by fine-tuning a popular, state-of-the-art VLA [39] on the `tape in penholder` and `move knight` tasks. We first use SFT to fine-tune the SmolVLA action head with 10 demonstrations, as the publicly available pre-trained checkpoint failed zero-shot on both tasks. After training with SFT, the policy adapts to the tasks but performance is still suboptimal with success rates below 50%, as can be seen in Table 2. We utilize OTQL to adapt SmolVLA on a set of 30 rollouts collected form the fine-tuned VLA. From Table 2, we see that this is enough to adapt it to the two hardest out of the four tasks, increasing the average success rate by an average of 37% while bringing down its inference NFE to 3. We note that similarly to the single-task case, online finetuning could be more effective, however, given the time each training iteration takes, training online would result in intermittent robot motions and could possibly compromise success. Throughout these experiment, we kept the VLM of SmolVLA frozen as fully fine-tuning it in a very low-data regime could risk severe overfitting [54].

Task	SmolVLA	SmolVLA-OTQL
`tape in penholder`	14/30	**24/30**
`move knight`	9/30	**22/30**

Table 2: Comparison of SmolVLA performance on two real-world tasks trained with SFT and OTQL.

5 Conclusions

We present OTQL, a method that steers and accelerates flows. We employ weighted COT couplings and CFM to fine-tune policies, leading to an efficient and easy to implement training algorithm. We have shown that OTQL outperforms other post-training methods and that it can adapt *and* accelerate single-task flow policies and VLAs in the real-world with a limited interaction budget. OTQL can act as the bridge between large-scale pretraining and efficient real-time performance on new tasks that the pretrained policy solves only partially.

Limitations. OTQL is effective in adapting flow policies in both real and simulation environments using a single mechanism for steering and acceleration, however, there are certain limitations that need to be highlighted. First,the policy can't be treated as a black box, contrary to other methods [16, 18], since OTQL assumes the policy is a neural ODE-based generative model. The complexity of post-training large flow policies can be eliminated with techniques like LoRA [55], however not all VLAs or WAMs are compatible with our method as formulated in this paper. Second, the inference speed-up and the adaptation become more modest as the dimensionality of the problem increases, as minibatch OT is prone to larger errors in higher dimensions. Lastly, our method inherits many of the problems typically encountered in real-world RL methods such as manual episode labeling and manual environment resets after every episode.

References

[1] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[2] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[3] J. Pai, L. Achenbach, V. Montesinos, B. Forrai, O. Mees, and E. Nava. mimic-video: Video-action models for generalizable robot control beyond vlas. *arXiv preprint arXiv:2512.15692*, 2025.

[4] M. J. Kim, Y. Gao, T.-Y. Lin, Y.-C. Lin, Y. Ge, G. Lam, P. Liang, S. Song, M.-Y. Liu, C. Finn, and J. Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *International Conference on Learning Representations (ICLR)*, 2026.

[5] M. Reuss, H. Zhou, M. Rühle, Ö. E. Yağmurlu, F. Otto, and R. Lioutikov. FLOWER: Democratizing generalist robot policies with efficient vision-language-flow models. In *Conference on Robot Learning (CoRL)*, 2025.

[6] K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. R. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, b. ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Conference on Robot Learning (CoRL)*, 2025.

[7] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems (RSS)*, 2024.

[8] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[9] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

[10] A. Tong, K. Fatras, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, G. Wolf, and Y. Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research (TMLR)*, 2024.

[11] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems (NeurIPS)*, 2020.

[12] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

[13] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research (IJRR)*, 2023.

[14] M. Janner, Y. Du, J. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, 2022.

[15] J. Zhang, Y. Luo, A. Anwar, S. A. Sontakke, J. J. Lim, J. Thomason, E. Biyik, and J. Zhang. Rewind: Language-guided rewards teach robot policies without new demonstrations. In *Conference on Robot Learning (CoRL)*, 2025.

[16] A. Wagenmaker, Y. Zhang, M. Nakamoto, S. Park, W. Yagoub, A. Nagabandi, A. Gupta, and S. Levine. Steering your diffusion policy with latent space reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2025.

[17] M. Du and S. Song. Dynaguide: Steering diffusion polices with active dynamic guidance. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2026.

[18] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal. From imitation to refinement-residual rl for precise assembly. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2025.

[19] L. Ankile, Z. Jiang, R. Duan, G. Shi, P. Abbeel, and A. Nagabandi. Residual off-policy rl for finetuning behavior cloning policies. *arXiv preprint arXiv:2509.19301*, 2025.

[20] Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2023.

[21] F. N. Tiofack, T. L. Hellard, F. Schramm, N. Perrin-Gilbert, and J. Carpentier. Guided flow policy: Learning from high-value actions in offline reinforcement learning. *arXiv preprint arXiv:2512.03973*, 2025.

[22] S. Zhang, W. Zhang, and Q. Gu. Energy-weighted flow matching for offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025.

[23] Q. Li, Z. Zhou, and S. Levine. Reinforcement learning with action chunking. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2026.

[24] Q. Li and S. Levine. Q-learning with adjoint matching. *arXiv preprint arXiv:2601.14234*, 2026.

[25] S. Park, Q. Li, and S. Levine. Flow q-learning. In *International Conference on Machine Learning (ICML)*, 2025.

[26] Z. Wang, Z. Li, A. Mandlekar, Z. Xu, J. Fan, Y. Narang, L. Fan, Y. Zhu, Y. Balaji, M. Zhou, et al. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024.

[27] A. Prasad, K. Lin, J. Wu, L. Zhou, and J. Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. In *Robotics: Science and Systems (RSS)*, 2024.

[28] A. Sochopoulos, N. Malkin, N. Tsagkas, J. Moura, M. Gienger, and S. Vijayakumar. Fast flow-based visuomotor policies via conditional optimal transport couplings. In *Conference on Robot Learning (CoRL)*, 2025.

[29] X. Hu, qiang liu, X. Liu, and B. Liu. Adaflow: Imitation learning with variance-adaptive flow-based policies. In *Advances in neural information processing systems (NeurIPS)*, 2024.

[30] G. Lu, Z. Gao, T. Chen, W. Dai, Z. Wang, and Y. Tang. Manicm: Real-time 3d diffusion policy via consistency model for robotic manipulation. *arXiv preprint arXiv:2406.01586*, 2024.

[31] H. Ding, N. Jaquier, J. Peters, and L. Rozo. Fast and robust visuomotor riemannian flow matching policy. *arXiv preprint arXiv:2412.10855*, 2024.

[32] G. Kerrigan, G. Migliorini, and P. Smyth. Dynamic conditional optimal transport through simulation-free flows. In *Advances in neural information processing systems (NeurIPS)*, 2024.

[33] H. K. Cheng and A. Schwing. The curse of conditions: Analyzing and improving optimal transport for conditional flow-based generation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[34] T. Yin, M. Gharbi, R. Zhang, E. Shechtman, F. Durand, W. T. Freeman, and T. Park. One-step diffusion with distribution matching distillation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[35] T. Salimans and J. Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022.

[36] K. Frans, D. Hafner, S. Levine, and P. Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations (ICLR)*, 2025.

[37] D. Kim, C.-H. Lai, W.-H. Liao, N. Murata, Y. Takida, T. Uesaka, Y. He, Y. Mitsufuji, and S. Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. In *International Conference on Learning Representations (ICLR)*, 2024.

[38] L. Yang, Z. Zhang, Z. Zhang, X. Liu, M. Xu, W. Zhang, C. Meng, S. Ermon, and B. Cui. Consistency flow matching: Defining straight flows with velocity consistency. *arXiv preprint arXiv:2407.02398*, 2024.

[39] M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, A. Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[40] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems (NeurIPS)*, 2018.

[41] J. Peters and S. Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *International Conference on Machine learning (ICML)*, 2007.

[42] J. Peters, K. Mulling, and Y. Altun. Relative entropy policy search. In *AAAI Conference on Artificial Intelligence*, 2010.

[43] A. Nair, A. Gupta, M. Dalal, and S. Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[44] A.-A. Pooladian, H. Ben-Hamu, C. Domingo-Enrich, B. Amos, Y. Lipman, and R. T. Chen. Multisample flow matching: Straightening flows with minibatch couplings. In *International Conference on Machine Learning (ICML)*, 2023.

[45] J.-D. Benamou and Y. Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.

[46] P. Intelligence, A. Amin, R. Aniceto, A. Balakrishna, K. Black, K. Conley, G. Connors, J. Darpinian, K. Dhabalia, J. DiCarlo, D. Driess, M. Equi, A. Esmail, Y. Fang, C. Finn, C. Glossop, T. Godden, I. Goryachev, L. Groom, H. Hancock, K. Hausman, G. Hussein, B. Ichter, S. Jakubczak, R. Jen, T. Jones, B. Katz, L. Ke, C. Kuchi, M. Lamb, D. LeBlanc, S. Levine, A. Li-Bell, Y. Lu, V. Mano, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, C. Sharma, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, W. Stoeckle, A. Swerdlow, J. Tanner, M. Torne, Q. Vuong, A. Walling, H. Wang, B. Williams, S. Yoo, L. Yu, U. Zhilinsky, and Z. Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

[47] H. Chen, C. Lu, C. Ying, H. Su, and J. Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

[48] S. Park, K. Frans, B. Eysenbach, and S. Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, 2025.

[49] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[50] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec, A. Zouitine, S. Palma, P. Kooijmans, M. Aractingi, M. Shukor, D. Aubakirova, M. Russi, F. Capuano, C. Pascal, J. Choghari, J. Moss, and T. Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. https://github.com/huggingface/lerobot, 2024.

[51] M. Psenka, A. Escontrela, P. Abbeel, and Y. Ma. Learning a diffusion model policy from rewards via q-score matching. In *International Conference on Machine Learning (ICML)*, 2024.

[52] P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

[53] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[54] D. Driess, J. Springenberg, B. Ichter, L. Yu, A. Li-Bell, K. Pertsch, A. Ren, H. Walke, Q. Vuong, L. X. Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *Advances in Neural Information Processing Systems (NeurIPS)*, 2026.

[55] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICRL)*, 2022.

[56] Y. Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417, 1991.

[57] B. Hosseini, A. W. Hsu, and A. Taghvaei. Conditional optimal transport on function spaces. *arXiv preprint arXiv:2311.05672*, 2023.

[58] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

[59] N. Agarwal, A. Ali, M. Bala, Y. Balaji, E. Barker, T. Cai, P. Chattopadhyay, Y. Chen, Y. Cui, Y. Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

[60] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on computer vision and pattern recognition (CVPR)*, 2022.

[61] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35: 36479–36494, 2022.

[62] J. Urain, A. Mandlekar, Y. Du, M. Shafiullah, D. Xu, K. Fragkiadaki, G. Chalvatzaki, and J. Peters. Deep generative models in robotics: A survey on learning from multimodal demonstrations. *arXiv preprint arXiv:2408.04380*, 2024.

[63] J. Ho and T. Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[64] Y. Hou, Z. Liu, C. Chi, E. Cousineau, N. Kuppuswamy, S. Feng, B. Burchfiel, and S. Song. Adaptive compliance policy: Learning approximate compliance for diffusion guided control. In *International Conference on Robotics and Automation (ICRA)*, 2025.

[65] H. Xue, J. Ren, W. Chen, G. Zhang, Y. Fang, G. Gu, H. Xu, and C. Lu. Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation. *arXiv preprint arXiv:2503.02881*, 2025.

[66] E. Chisari, N. Heppert, M. Argus, T. Welschehold, T. Brox, and A. Valada. Learning robotic manipulation policies from point clouds with conditional flow matching. *Conference on Robot Learning (CoRL)*, 2024.

[67] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *Robotics: Science and Systems (RSS)*, 2024.

[68] S. Li, Y. Gao, D. Sadigh, and S. Song. Unified Video Action Model. In *Robotics: Science and Systems (RSS)*, 2025.

[69] C. Zhu, R. Yu, S. Feng, B. Burchfiel, P. Shah, and A. Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

[70] J. Zhang, K. Zheng, K. Jiang, H. Wang, I. Stoica, J. E. Gonzalez, J. Chen, and J. Zhu. Turbodiffusion: Accelerating video diffusion models by 100-200 times. *arXiv preprint arXiv:2512.16093*, 2025.

[71] Z. Geng, M. Deng, X. Bai, Z. Kolter, and K. He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems (NeurIPS)*, 2026.

[72] K. Black, M. Galliker, and S. Levine. Real-time execution of action chunking flow policies. *Advances in Neural Information Processing Systems (NeurIPS)*, 2026.

[73] Z. Li, R. Krohn, T. Chen, A. Ajay, P. Agrawal, and G. Chalvatzaki. Learning multimodal behaviors from scratch with diffusion policy gradient. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[74] S. Fujimoto and S. Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[75] X. Huang, X. Liu, E. Zhang, T. Yu, and S. Li. Offline-to-online reinforcement learning with classifier-free diffusion generation. *International Conference on Machine learning (ICML)*, 2025.

[76] S. Venkatraman, M. Jain, L. Scimeca, M. Kim, M. Sendera, M. Hasan, L. Rowe, S. Mittal, P. Lemos, E. Bengio, et al. Amortizing intractable inference in diffusion models for vision, language, and control. *Advances in neural information processing systems (NeurIPS)*, 2024.

[77] L. Fang, R. Liu, J. Zhang, W. Wang, and B. Jing. Diffusion actor-critic: Formulating constrained policy iteration as diffusion noise regression for offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025.

A Optimal Transport

The static OT problem aims to find a coupling of minimal cost between two distributions. Given a cost function $C(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, we search for a solution to the optimization problem:

$$\mathrm{OT}(p_z, p_a) = \inf_{\pi \in \Pi(p_z, p_a)} \mathbb{E}_{(z,a)\sim\pi}[C(z,a)], \tag{7}$$

where $\Pi(p_z, p_a)$ denotes the space of probability measures whose left and right marginals equal p_z and p_a, respectively. Intuitively, we seek a way to (stochastically) transport particles distributed according to p_z so as to make them distributed according to p_a, minimizing the total cost of transportation. The infimum in (7) is called the OT cost, and a minimizer is called an OT plan (under some conditions, *e.g.*, squared Euclidean cost and p_z absolutely continuous with finite variance), the infinimum is achieved and the minimizer is unique [56]). In the case where $C(z,a) = ||z-a||^2$ is the squared Euclidean distance, the square root of the distance in (7) is called the 2-Wasserstein distance and denoted $W_2(p_z, p_a)$.

The 2-Wasserstein distance can also take a dynamic form, called the Benamou-Brenier form [45], which involves a flow vector field $u(t,x)$, which transforms one probability distribution to the other:

$$W_2(p_z, p_a) = \inf_{p(t,x), u(t,x)} \int_{\mathbb{R}^d} \int_0^1 p(t,x)\|u(t,x)\|^2 \, dt \, dx, \tag{8}$$

, where $p(t,x)$ is the probability path generated by $u(t,x)$ and has marginal distributions $p(0,\cdot) = p_z,\ p(1,\cdot) = p_a$. The minimizer of (8) is a vector field that produces sample paths of minimum length, which subsequently forces them to be straight and thus easier to simulate with limited `euler` steps.

In the case of conditional distributions with conditions $c \in \mathcal{C}$ a relaxed version of the static OT problem is the conditional Kantorovich problem [57]:

$$\inf_{T \in \Pi_c} \int_{\mathbb{R}^d \times \mathbb{R}^d} c((z,\tilde{c}),(a,c)) \, dT((z,\tilde{c}),(a,c)), \tag{9}$$

where $\tilde{c}$ are source conditions and $\Pi_c := \{T \in \Pi \mid ((z,\tilde{c}),(a,c)) \sim T, \text{such that } \tilde{c} = c\}$ is the space of conditional distributions where the conditions are equal. Kerrigan et al. [32] introduced an equivalent Benamou-Brenier dynamic form for the conditional OT problem.

B Related Work

Diffusion/flow models for robot control. Diffusion [11, 12] and flow [9] models have demonstrated excellent multimodal and high-dimensional distribution learning capabilities [58, 59, 60, 61], which rendered them a state-of-the-art model choice for action generation in robotics [62]. Chi et al. [13] first introduced diffusion models for robot control via behavior cloning, by learning the conditional distribution of action chunks instead of single-step actions. Other early works have investigated the use of diffusion models as planners [14] that use rewards to steer the generation of actions using diffusion guidance [63]. Diffusion and flow models have also proven effective policy representations in scenarios with multimodal observations, such as forces [64, 65] and point-clouds [66, 67].

Diffusion and flow models, owing to their ability to sample from complex distributions with flexible conditioning, are very popular components for action generation in more complicated architectures used for generalist policies. More specifically, most VLA architectures [39, 7, 6, 8, 5], either export features from the VLM or generate next tokens autoregressively with the VLM which are used for conditioning a flow- or diffusion-based action head. The same principle has also been extended to WAMs [4, 3], which use a video or world model backbone instead of a VLM. Other architectures, train diffusion models which generate both future images and actions [68, 69] with the aim of improving scalability through large-scale video pretraining.

Inference acceleration of generative policies. Diffusion and flow model inference requires integration of an ODE or SDE and therefore multiple neural function evaluations for generating one sample from the target distribution. As a consequence, accelerating inference by reducing the amount of integration steps needed for the SDE/ODE integration is relevant across many fields [70, 33, 71]. Especially in robotics, high inference speeds are crucial for ensuring reactive and continuous robot motions [72]. Some methods have achieved one-step inference using consistency objectives [27, 26, 30] at the cost of more expensive training. Other methods avoid computationally complex training by using the flow variance to adapt the integration steps [29] or by using COT couplings to straighten the integration paths [28]. More efficient training, however, comes at the cost of one step inference quality, as they typically require 2-3 NFE in action generation problems. Instead of modifying the training objective of the generative policies, Black et al. [72] propose a practical post-processing alternative to overcoming intermittent robot motions, by generating the next action chunk while the current one is already being executed by the robot.

RL adaptation of flow policies. The same qualities that led to the success of diffusion and flow models as robot policies have proven useful in reinforcement learning too [51, 73, 52]. Flow models have unlocked new opportunities for policy and value learning, particularly in offline RL settings, where the policy and critic have access only to a static dataset of rollouts collected by a base policy [74]. The goal in offline RL is to sample from the tilted distribution that is the closed form solution of the KL regularized problem [41, 42]. Sampling from this distribution has been achieved using advantage and critic weighted flow matching objectives [22, 21], adjoint matching [24], rejection sampling [52, 23], classifier guidance [75] and by using the relative trajectory balance objective for diffusion models [76]. Since all these methods require multi-step integration for action generation, [25] distills the behavior of a flow policy to a single-step guassian policy while also optimizing a Q loss.

In robotics, access to a suboptimal pre-trained base policy is assumed and the goal is to extract a policy that samples high value actions and actions that also have a high density under the base policy density. Wagenmaker et al. [16] learn a policy that generates noise samples for the base policy, using RL. Other methods focus on guidance [17] or residual policies correcting the actions generated by the base policy [19, 18].

C Ablation

In this section we ablate the most important hyper-parameters of OTQL and provide some insight on how to tune them. In Fig. 6 we show the effects of the temperature parameter λ, the number of action candidates used for rejection sampling N and the NFE used for sampling actions.

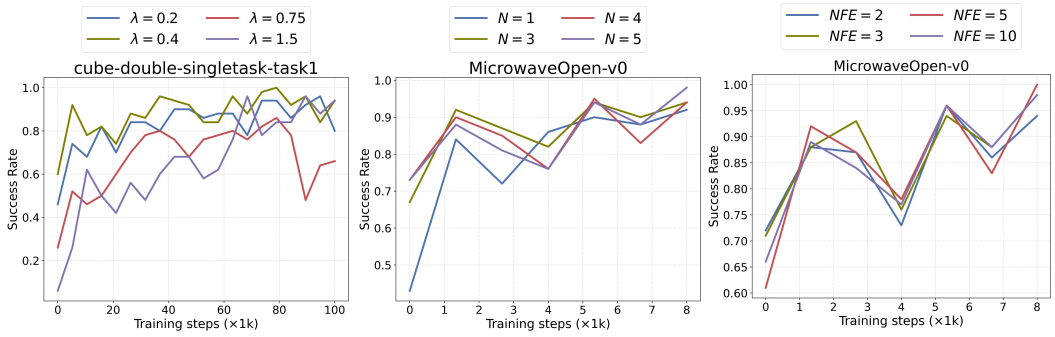


Figure 6: Ablation of OTQL for λ, N and inference steps NFE

Temperature λ. OTQL uses importance weights to assign more mass to low energy samples in the batch while calculating the empirical COT coupling. If the temperature λ is too high then there will be only a few samples in the batch that will be assigned a significant weight leading to a transport plan Γ^* that is concentrated around these few points. This means that when we sample from Γ^* most of the batch will be potentially discarded, leading to a significantly reduced effective batch size. This

can make the training slower to converge as depicted in Fig. 6 for $\lambda = 1.5$. The policy still ends up performing adequately, however many more training iterations are required until convergence.

On the other side, if λ is too small then potentially all samples in the batch could be assigned similar weights. This can lead to a COT coupling that does not prioritize high advantage samples and therefore fails to achieve high success rates. In practice, we choose λ to be between $[0.4, 1.0]$ in tasks where the chunk size is 10 and we have approximately 100-200 maximum episode steps. Other tasks and experiment choices can greatly impact the optimal value for λ.

Rejection sampling candidates N. As explained in §3.2 in certain tasks we employ rejection sampling with a relatively low number of candidates ($N = 5$). This decision is grounded on the fact that we have to choose a moderate λ value to avoid the problem described before, which might lead to difficulties in adapting policies with critics that have sharp peaks, or when high advantage actions are sparse. In Fig. 6 we show how rejection sampling assists our method. We use a moderate λ value which gives us a good approximation of the tilted distribution $\tilde{\pi}$ and then use rejection sampling to make it more concentrated around high value actions. Eventually, rejection sampling assists in earlier training iterations resulting in more successful rollouts in the buffer which in turn assists OTQL. We see that towards the middle and end of training rejection sampling does not further improve the performance compared to vanilla OTQL. This shows that OTQL does indeed take care of most of the adaptation and rejection sampling acts only as a lightweight filtering. This is also the reason why 3-5 candidates are enough compared to QC which uses 32 in many tasks on OGBench [23].

NFE for flow inference. OTQL uses COT coupling both for straightening the integration paths of the flow and for distribution steering. Apart from the competitive performance shown in §4 with 2 or 3 NFE during inference and training, we also show that flows trained with OTQL perform the same no matter the amount of steps used for inference and training. This can be seen in Fig. 6, where we see that the fine-tuned policy has the same performance overall throughout training for all NFE between 2 and 10. Many flow and diffusion RL methods [51, 77, 24] require at least 10 NFE for achieving competitive performance, while our method is competitive with NFE = 2 and without using distillation [25, 21].

D Implementation details

D.1 Offline RL

In all offline experiments involving OGBench we use the exact same evaluation setup as in [24]. We therefore also employ an ensemble of $K = 10$ critic functions $Q^k,\ k = 1, \ldots, 10$ and the pessimistic target value backup [77]:

$$L_{Q_k} = \mathbb{E}\big[(Q^k_\phi(s,a) - y)^2\big], \tag{10}$$

where $y = r + \frac{\gamma}{K}\big(\sum_k Q^k_{\bar{\phi}}(s,a) - \rho\sqrt{\sum_k (Q^k_{\bar{\phi}}(s',a') - \frac{1}{K}\sum_k Q^k_{\bar{\phi}}(s',a'))^2}\big)$ and ρ is the pessimistic coefficient.

All models used in the policy and the critic were MLPs with parameters shown in Table 3.

Parameter	Value
Batch size	256
Discount factor (γ)	0.99 for `puzzle,scene,cube,antmaze-large` 0.995 for `humanoidmaze,antmaze-giant`
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Target network update rate (λ)	5×10^{-3}
Critic ensemble size (K)	10
Critic target pessimistic coefficient (ρ)	0.5 for `puzzle,scene,cube,antmaze` 0 for `humanoidmaze`
UTD ratio	1
Number of offline training steps	10^6
Hidden dimensions	512
Network depth	4 hidden layers

Table 3: Parameters used in the offline RL experiments

Environment	OTQL			
	λ	w_{max}	N	NFE
`antmaze-large-singletask`	2	3	1	2
`antmaze-giant-singletask`	2	3	1	2
`humanoidmaze-medium-singletask`	5	1	1	2
`humanoidmaze-large-singletask`	5	1	1	2
`cube-double-singletask`	0.4	10	5	2
`cube-triple-singletask`	0.4	10	5	2
`scene-singletask`	0.4	10	5	2
`puzzle-3x3-singletask`	0.4	10	5	2

Table 4: Hyperparameters used for OTQL in the offline RL tasks.

D.2 Online RL in simulation

We ran online RL post-training for OTQL and all the baselines on two MuJoCo tasks visible in Fig. 7. The first (`OpenMicrowave-v0`) requires the robot to open the microwave door and the second (`ArrangeBoxes-v0`) requires the robot to arrange the two boxes on the correct colored planes. For both tasks we collect demonstrations using a gamepad by controlling only the position of the end-effector and the gripper. We again utilized an ensemble for the critic but not the pessimistic Bellman update. All networks used for the critic and the flow policies where MLPs.

We followed the pipeline presented in §3 and first collected transitions of the base policy interacting with the environment for 2k steps. We also included the offline demonstrations collected $\mathcal{D}_{off}$ in the buffer $\mathcal{B}$ and during training we sampled half of the batch from $\mathcal{D}_{off}$ and half from $\mathcal{B}$. We used again an ensemble of critics on all of the environments.

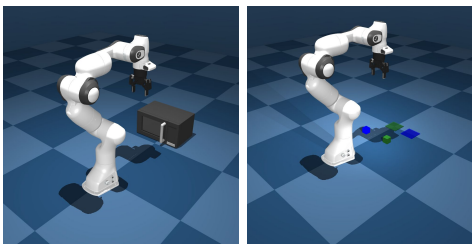


Figure 7: MuJoCo tasks for online RL post-training

Parameter	Value
Batch size	256
Discount factor (γ)	0.99
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Target network update rate (λ)	5×10^{-3}
Critic ensemble size (K)	2
Critic target pessimistic coefficient (ρ)	0.0
UTD ratio	10
Number of online training steps	8000
Number of online training steps before training	2000
Hidden dimensions	256
Network depth	4 hidden layers

Table 5: Parameters used in the online RL experiments

Environment	FQL		QC		EWFM			OTQL			
	α	NFE	N	NFE	λ	w_{max}	NFE	λ	w_{max}	N	NFE
`ArrangeBoxes-v0`	300	1	32	10	1.0	20	10	1.5	10	5	3
`MicrowaveOpen-v0`	300	1	32	10	1.0	20	10	1.5	10	1	3

Table 6: Hyperparameters used for OTQL and the baselines in the online RL tasks.

Baselines. The baselines considered in these experiments are QC [23], FQL [25] and EWFM, which we adapted from [22]. More specifically we used advantage weighting for the weighted flow matching objective instead of critic weighting and employed the same maximum weight clipping technique, like in OTQL. The details of all hyperparameters used in all the baselines can be found in Table 6 and the parameters used for training in Table 5.

D.3 RL in real-world

Similar to the offline and online simulation experiments we use an ensemble of critics for all real-world experiments, but without the pessimistic target update. the experimental protocols for both single-task policies and SmolVLA adaptation are detailed in §4.3. For single task policies we use 10

expert demonstrations per task and 50 total rollouts (60 episodes) and for the SmolVLA experiments we use 10 expert demonstrations and 30 rollouts (40 episodes). The details of all hyperparameters used in the single-task experiments can be found in Table 7 and Table 8 and for the SmolVLA experiments in Table 9 and Table 10. For all experiments we used LeRobot [50].

Parameter	Value
Batch size	256
Discount factor (γ)	0.99
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Target network update rate (λ)	5×10^{-3}
Critic ensemble size (K)	2
Critic target pessimistic coefficient (ρ)	0.0
Rollouts collected before training	10
Rollouts collected every 20k training steps	10

Table 7: Parameters used in the real-world tasks for single-tasks policies.

	OTQL			
Task	λ	w_{max}	N	NFE
`stack cups`	0.75	10	5	3
`pen in penholder`	0.75	10	5	3
`open door`	0.75	10	5	3

Table 8: Hyperparameters used for OTQL in the real-world tasks.

Parameter	Value
Batch size	64
Discount factor (γ)	0.99
Actor learning rate	1×10^{-4}
Critic learning rate	3×10^{-4}
Target network update rate (λ)	5×10^{-3}
Critic ensemble size (K)	2
Critic target pessimistic coefficient (ρ)	0.0

Table 9: Parameters used in the real-world tasks for SmolVLA.

	OTQL			
Task	λ	w_{max}	N	NFE
`pen in penholder`	1.0	10	1	3
`move knight`	1.0	10	1	3

Table 10: Hyperparameters used for OTQL in the real-world tasks for SmolVLA adaptation.

E Hardware

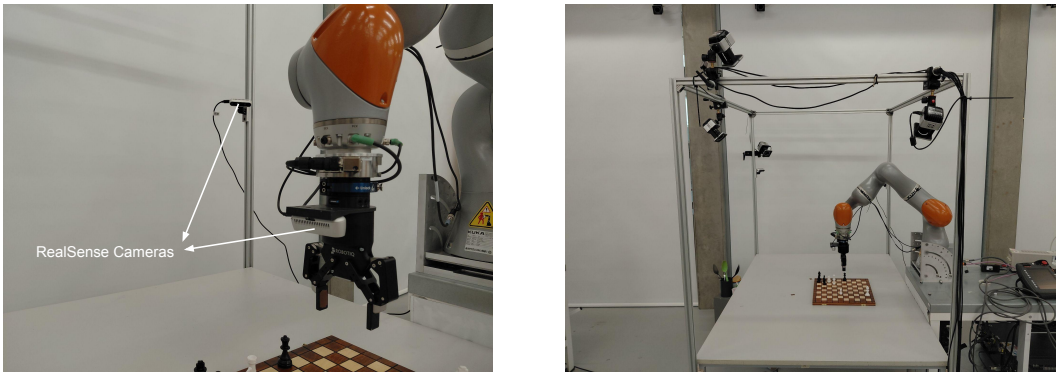


Figure 8: real-world robot setup

Experiment setup. The real-world experiments were performed on a KUKA IIWA 14 robot with a Robotiq 2F-85 gripper and we used 2 Realsense D415 cameras—one mounted on the end-effector and one providing an external view of the workspace. The entire setup can be seen in Fig. 8. The real-world inference was performed using a desktop PC with (CPU, RAM, GPU): Intel(R) i9-9900KF, 64GB, NVIDIA GeForce RTX 2080. Data collection was performed using a 6D Spacemouse device.

Training and simulation evaluation. Training and evaluation for the simulated tasks were performed using a workstation with (CPU, RAM, GPU): AMD Ryzen Threadripper PRO 5965WX 24-Cores, 128GB, 2× NVIDIA GeForce RTX 4090.