POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Cross-Embodiment Policy Learning for Robotic Manipulation

Supervisors

Prof. Giuseppe Bruno Averta

Dr.rer.nat Zhenshan Bing

Candidate

Federico Morro

December 2025

Abstract

The recent advancements in the machine learning field have demonstrated the potential of knowledge transfer and multi-task learning to enhance the performance and generalization capabilities of models across various domains. In the context of robotics, the ability to transfer skills between different embodiments is particularly appealing, as it can significantly reduce the time and resources required for training control agents, while also improving their adaptability and robustness. This thesis investigates how to leverage demonstrations and Reinforcement Learning (RL) to train agents capable of solving diverse manipulation tasks using multiple robotic arms and grippers.

The proposed method utilizes a contrastive supervised learning approach to construct a shared representation of different robotic configurations and tasks, aligning state and action spaces across diverse embodiments while preserving the critical distinctions necessary for accurate task execution. The approach employs a language-conditioned vision-based policy, which poses significant challenges but offers greater applicability to real-world scenarios. Additionally, to construct the dataset for the vision-based agent, the thesis introduces a novel methodology that leverages demonstrations from a single embodiment to accelerate and improve the learning of state-based RL policies for diverse embodiments. This is achieved by re-rolling aligned demonstrations and incorporating an advantage-weighted behavioral cloning term into the RL training process.

To assess the effectiveness of the proposed methods, extensive experiments are conducted in simulated environments using the MuJoCo physics engine. The state-based RL agents exhibit accelerated learning and improved performance, demonstrating the effectiveness of the knowledge transfer pipeline. The vision-based policy achieves significant task performance across diverse embodiments and shows promising generalization capabilities to unseen robot-gripper configurations. However, when addressing more complex tasks, certain limitations become evident, particularly in the vision-based policy. In these cases, performance tends to decrease for grippers that require precise object interaction or exhibit morphologies substantially different from those encountered during training. Overall, these findings provide valuable insights into the strengths and limitations of the proposed approaches and suggest potential directions for future research.

Acknowledgments

This thesis has been carried out at the Chair of Robotics, Artificial Intelligence and Real-time Systems at the Technical University of Munich (TUM), in the context of an exchange program from Politecnico di Torino (PoliTo). I would like to express my sincere gratitude to both institutions for providing me with the opportunity to conduct this research.

I am deeply grateful to my PoliTo supervisor, Prof. Giuseppe Bruno Averta, for his continuous support, encouragement, and valuable feedback during the development of this thesis.

I would also like to thank my TUM supervisor, Dr. rer. nat. Zhenshan Bing, for the possibility to work on this research topic and for his guidance throughout the project.

to my friends and family
who supported me along the way

Table of Contents

List of Tables VIII

List of Figures IX

Acronyms XI

1 Introduction 1

2 Preliminaries 4

2.1 Transformer Model 4

2.1.1 Vision Transformer 6

2.2 Diffusion Models 7

2.3 Reinforcement Learning 9

2.3.1 Taxonomy of RL Algorithms 11

2.3.2 Soft Actor-Critic Algorithms 12

2.4 Imitation Learning 16

2.5 Advanced Learning Paradigms 17

2.5.1 Representation Learning 17

2.5.2 Multi-Task Learning 17

3 State of the Art 19

3.1 Robotic Manipulation Learning 19

3.1.1 Diffusion Models for Robotic Manipulation 20

3.1.2 Grasping 20

3.2 Generalist Agents 21

3.3 Cross-Embodiment Learning 22

3.3.1 Knowledge Transfer 22

3.3.2 Skill Transfer 23

3.3.3 Dynamics Gap Bridging 24

3.3.4 Vision Gap Bridging 24

3.4 Leveraging Demonstrations for Policy Learning 25

3.4.1 Reinforcement Learning with Demonstrations 25
3.4.2 Weighted Imitation Learning 26

4 Methodology **27**
4.1 State-Based Agents 28
4.1.1 State and Action Spaces Alignment 29
4.1.2 SAC with Advantage-Weighted BC 30
4.2 Dataset Generation Procedure 33
4.3 Vision-Based Multi-Embodiment Policy 33
4.3.1 Fine-Tuning Configuration 35
4.3.2 Input Modalities 36
4.3.3 Contrastive Losses for Embodiment Generalization 38
4.3.4 Weighted Behavioral Cloning 42

5 Experiments **43**
5.1 Experimental Setup 43
5.2 State-Based Agent Training 44
5.2.1 Training Results vs Baseline SAC 46
5.2.2 Performance Evaluations 50
5.3 Dataset Generation 51
5.4 Vision-Based Policy Training 51
5.4.1 Representation Learning Settings and Results 52
5.4.2 Task Performance Evaluations 59
5.4.3 Ablation Studies 63

6 Conclusions **66**
6.1 Discussion 66
6.2 Future Directions 67

A Appendix **68**
A.1 Vision-Based Policy Additional Experiments 68
A.1.1 Multiple Action Heads 68
A.1.2 RL Finetuning 69
A.1.3 State Inputs 70
A.2 Additional Figures and Results 70

Bibliography **76**

List of Tables

5.1 State-Based Agents State Space 46
5.2 State-Based Agents Action Space 46
5.3 State-Based Agents Training Hyperparameters 47
5.4 State-Based Agents Performance Training Results 48
5.5 *Lift* Task State-Based Agents Success Rates (%) 50
5.6 *PickPlaceCan* Task State-Based Agents Success Rates (%) 50
5.7 Vision-Based Policy Tokens per Input Type 52
5.8 Vision-Based Policy Finetuning Hyperparameters 53
5.9 Vision-Based Policy Contrastive Losses Hyperparameters 54
5.10 Representation Learning Contrastive Losses Results: Hard Labels (k-NN: k=3,5,10). 58
5.11 Representation Learning Contrastive Losses Results: Timestep + Hard Task Labels (k-NN: k=3,5,10). 58
5.12 Vision-Based Policy Performance Evaluations 60
5.13 Vision-Based Policy Performance Evaluations Summary 60

A.1 Vision-Based Policy Performance With State Inputs VS Without State Inputs 70
A.2 Vision-Based Multi-Embodiment Per-Configuration Performance Evaluations 75

List of Figures

2.1 Overview of the attention mechanism (left) and of the multi-head configuration (right). Source: arXiv preprint of [1] (arXiv:1706.03762). 6

2.2 Overview of the ViT architecture, and of the transformer encoder model. Source: arXiv preprint of [3] (arXiv:2010.11929). 7

4.1 Overview of the Octo architecture. Source: arXiv preprint of [2] (arXiv:2405.12213). 34

4.2 Overview of the vision-based multi-embodiment policy architecture, including input modalities, model components, and training losses. In the figure, "embs" stands for "embeddings", while h, w, and a are the transformer hidden size, the window of input temporal observation size, and the action chunk size, respectively. The same figure is reported in horizontal format in the Appendix A.1 for better readability. 35

4.3 Examples of gripper point cloud representations (*Robotiq140Gripper*) used as input to the model in open (left), half-open (center), and closed (right) configurations. 38

5.1 *Lift* task progress in `robosuite` with *Panda* robot and *Panda* gripper. 44

5.2 *PickPlaceCan* task progress in `robosuite` with *Panda* robot and *Panda* gripper. 44

5.3 Robot arms used in the experiments: (a) *Panda*, (b) *IIWA*, (c) *Jaco* for training, (d) *Kinova3*, (e) *UR5e* for testing. 45

5.4 Grippers used in the experiments: (a) *Panda*, (b) *Rethink*, (c) *Robotiq140*, (d) *RobotiqThreeFinger* for training, (e) *Robotiq85*, (f) *JacoThreeFinger* for testing. 45

5.5 State-based agents average return during training for the *Lift* task. 49

5.6 State-based agents average return during training for the *PickPlaceCan* task. 49

5.7 Empirical Analysis for Contrastive Loss Hyperparameter Selection . 54

5.8 t-SNE Projections of Embeddings with Contrastive Losses. 56

5.9 Ablation studies for the *Lift* task 64
5.10 Ablation studies for the *PickPlaceCan* task 64

A.1 Overview of the vision-based multi-embodiment policy architecture in horizontal format for better readability, as an alternative to Figure 4.2. 71
A.2 Visualization of all robot-gripper configurations solved by the state-based agents for the *Lift* task. 72
A.3 Visualization of all robot-gripper configurations solved by the state-based agents for the *PickPlaceCan* task. 73
A.4 Visualization of all robot-gripper configurations (5 robots $\times$ 6 grippers) considered in the experiments with the vision-based policy. The unseen components during training are reported in blue in the titles of each image. 74

Acronyms

AWAC Advantage Weighted Actor-Critic

AWR Advantage Weighted Regression

BC Behavioral Cloning

BERT Bidirectional Encoder Representations from Transformers

CNN Convolutional Neural Network

DDPM Denoising Diffusion Probabilistic Model

DQN Deep Q-Learning

DRL Deep Reinforcement Learning

IK Inverse Kinematics

IL Imitation Learning

k-NN k-Nearest Neighbors

LfD Learning from Demonstrations

LLM Large Language Models

MDP Markov Decision Problem

MLP Multi Layer Perceptron

MTL Multi-Task Learning

NCE Noise Contrastive Estimation

NLP Natural Language Processing

PPO	Proximal Policy Optimization
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SAC	Soft Actor-Critic
SARSA	State-Action-Reward-State-Action
SNN	Soft Nearest Neighbors
t-SNE	t-Distributed Stochastic Neighbor Embedding
ViT	Vision Transformer

Chapter 1

Introduction

The recent advancements in the machine learning field have enabled significant progress in various domains, including computer vision, natural language processing, and robotics. These developments have enabled the creation of models that can learn to solve complex continuous control tasks, such as robotic manipulation ones, leveraging vision-based inputs and language conditioning. Nevertheless, these models often require substantial amounts of data and computational resources to train effectively; consequently, the possibility of reutilizing and transferring knowledge across different tasks or environments emerged as a promising direction to enhance their efficiency and generalization capabilities.

Traditional approaches to robotic learning often require training a separate policy for each embodiment-task combination, resulting in high computational and data collection costs. Moreover, the lack of shared representations across embodiments hinders the transferability of learned skills, limiting the scalability and applicability of these methods in real-world scenarios. Recent research has begun to address these limitations by exploring multi-task and multi-embodiment learning, aiming to develop models that can generalize across different robots and tasks.

This thesis focuses on improving the efficiency and generalization capabilities of robotic manipulation models across different embodiments. The main intuition lies in the observation that, even though different robotic arms and grippers may have varying kinematics and appearance, there exists underlying common patterns in the way they interact with objects and perform tasks. By identifying and leveraging these shared structures, it becomes feasible to transfer knowledge and train models that can generalize across multiple embodiments, reducing the need for extensive retraining and data collection for each new configuration.

The proposed approach found its foundation in generalist robotic agents, which aim to develop versatile models capable of performing a wide range of tasks across different environments and embodiments, and in representation learning techniques that facilitate the extraction of meaningful features from high-dimensional inputs. By combining these ideas, along with insights drawn from robotic manipulation learning literature, this thesis proposes to finetune a language-conditioned vision-based policy, with a contrastive supervised learning framework to achieve cross-embodiment generalization.

Additionally, to address the challenge of data scarcity for training such models, this thesis introduces a novel methodology for generating multi-embodiment robotic manipulation datasets. This approach involves training one agent per embodiment to then collect demonstrations for multiple robot-gripper configurations. This is achieved by training state-based RL agents that leverage adapted and re-rolled expert demonstrations collected with a different robot-gripper configuration to perform the same tasks using a particular robotic arm and gripper. By aligning state and action spaces across embodiments and incorporating an advantage-weighted behavioral cloning term into the RL training process, the method aims to accelerate and improve the learning of these agents.

To evaluate the effectiveness of both the state-based agents and the vision-based, multi-embodiment policy architecture, experiments are conducted in simulated environments, assessing task performance and generalization capabilities to unseen robots and grippers. The results demonstrate that both approaches enable effective learning while providing distinct advantages: the state-based method facilitates fast learning and exploration, while the vision-based approach shows promising cross-embodiment generalization. However, limitations are observed for more complex tasks and embodiment variations, providing insights into potential future research directions.

The main contributions of this thesis are summarized as follows:

- A novel methodology for training state-based RL agents leveraging adapted and re-rolled expert demonstrations collected with a different robot-gripper configuration.
- A contrastive supervised learning framework for finetuning a language-conditioned, vision-based policy architecture to achieve cross-embodiment generalization.
- Experimental validation of the proposed methods in simulated environments, with a detailed analysis of their strengths, limitations, and variations, providing insights for future research directions.

The remainder of this thesis is organized as follows. Chapter 2 introduces the foundational concepts and algorithms relevant to this work. Chapter 3 reviews the state of the art in robotic manipulation learning, generalist agents, and cross-embodiment knowledge transfer. Chapter 4 details the proposed methodology, including state-based RL agents, dataset generation, and vision-based policy architecture. Chapter 5 presents the experimental setup and results. Finally, Chapter 6 discusses the main findings, limitations, and future research directions.

Chapter 2

Preliminaries

In this chapter, the foundational concepts and algorithms relevant to this research are introduced.

It begins with the transformer model and the ViT architecture, followed by an overview of diffusion models and their generative principles. Reinforcement Learning is presented, with a focus on the Soft Actor-Critic algorithm, then Imitation Learning is discussed in the context of behavioral cloning. The chapter closes with relevant advanced learning paradigms, including representation learning, multi-task learning, and cross-embodiment knowledge transfer.

2.1 Transformer Model

The transformer model architecture [1] is a deep learning model that utilizes self-attention mechanisms to process sequential data. In the past few years, transformers have become the dominant architecture in several fields, mainly due to their ability to capture long-range dependencies in a parallelizable manner. Indeed, unlike traditional Recurrent Neural Network (RNN) architectures, which process temporal data sequentially requiring the computation of each time step to perform the next one, transformers leverage self-attention mechanisms to weigh the importance of different parts of the input data simultaneously, allowing for more efficient training and inference.

Another relevant feature of transformers is their ability to handle variable-length multimodal inputs, making them suitable for tasks that involve different types of data, such as proprioceptive and visual information.

In the latter, the main components of the transformer architecture are introduced:

- **Tokenization and Embedding**: The input data is preprocessed into a sequence of tokens. In Natural Language Processing (NLP), this often involves splitting text into words or syllables, then converting them into vectorized numerical representations, called embeddings, using feedforward layers or lookup tables. In robotics, this process can involve encoding multimodal data (e.g., images, joint states) into a sequence of feature vectors, by first splitting the data into fixed-size patches or segments, and then embedding them into a higher-dimensional space. The size of the embedding space is a hyperparameter of the model, typically denoted as d_{model} and referred to as the model dimension.

- **Positional Encoding**: Since transformers process input data in parallel, rather than sequentially like Recurrent Neural Networks (RNNs), they lack inherent information about the order of input tokens. To address this, positional encodings are added to the token embeddings to inform the model about the temporal or logical order of the input sequence. Common methods for positional encoding include sinusoidal functions, Equation (2.1), or learned embeddings.

$$PE_{(\text{pos},2i)} = sin\left(\frac{\text{pos}}{10000^{2i/d_{model}}}\right) \tag{2.1}$$

- **Self-Attention Mechanism**: The core of the transformer architecture is the self-attention mechanism, which allows the model to weigh the relative importance of different tokens in the input sequence when making predictions. This is achieved through the computation of attention scores, which determine how much focus each token should receive based on its relevance to other tokens. The self-attention mechanism enables the model to capture long-range dependencies and relationships within the data, and can be mathematically described as:

$$\text{Attention}(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.2}$$

  Where Q (queries), K (keys), and V (values) are matrices derived from the input embeddings via learned linear transformations, d_k is the dimensionality of the keys, and $softmax$ is applied row-wise to ensure that the attention weights sum to one. As the names suggest, the idea is that each token (query) attends to all other tokens (keys) to gather relevant information (values) for its own representation, and the attention scores determine the weight assigned to each value based on the similarity between the query and the keys.

- **Multi-Head Attention**: To enhance the model's ability to capture diverse relationships in the data, the transformer employs multi-head attention. This involves using multiple self-attention mechanisms in parallel, each with its own set of learned parameters. The outputs from these attention heads are then concatenated and linearly transformed to produce the final output. This allows the model to attend to different aspects of the input data simultaneously.

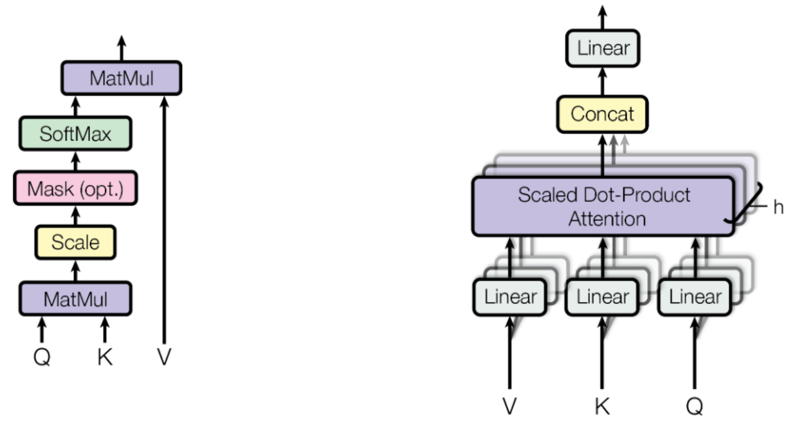


Figure 2.1: Overview of the attention mechanism (left) and of the multi-head configuration (right). Source: arXiv preprint of [1] (arXiv:1706.03762).

- **Feed Forward Layers**: In addition to the attention heads, transformers often include multiple feedforward neural network layers, normalization layers, and residual connections. These components help to stabilize training, improve convergence, and enhance the model's capacity to learn complex patterns in the data.

- **Output Layer (or Head)**: The final layer of the transformer architecture is typically a linear layer followed by a $softmax$ activation function, which produces the output predictions. In robotics applications or in regression tasks, the final layer may be adapted to produce continuous values, such as control actions or state estimates.

- **Masking**: In certain applications, it is necessary to prevent the model from considering certain tokens when computing attention scores. This is often done using masking techniques that, for instance, to ensure that future time steps are not considered when predicting the current action, and thus maintain causality.

2.1.1 Vision Transformer

In this thesis, the Octo model [2], which utilizes a ViT-inspired architecture to process multimodal inputs and generate robot control actions, is adopted as the backbone for the vision-based policy, so a brief overview of the ViT architecture is provided here.

The Vision Transformer (ViT) [3] is a model specifically designed for image inputs and has demonstrated state-of-the-art performance in various computer vision tasks, while being more computationally efficient than traditional Convolutional Neural Networks (CNNs). The ViT employs a patcher which divides an image into fixed-size non-overlapping patches, which are then flattened and linearly embedded into a sequence of tokens, then processed by the transformer architecture. In this context, the positional encodings are added to the patch embeddings to retain spatial information about the arrangement of patches in the original image. This allows the model to learn complex relationships between different regions of the image, while also allowing for the integration of additional modalities, such as proprioceptive data or task specifications, by embedding them as additional tokens in the input sequence.

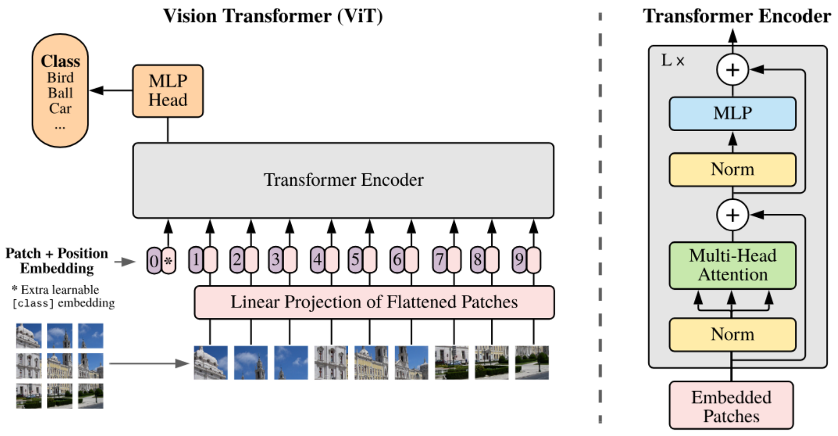


Figure 2.2: Overview of the ViT architecture, and of the transformer encoder model. Source: arXiv preprint of [3] (arXiv:2010.11929).

One crucial aspect to mention is the readout token, which is a special token appended to the input sequence that serves as a summary representation of the entire input. The readout token was firstly introduced in the BERT model [4], denoted as `[CLS]` token, where it is used to aggregate information from the entire input sequence for classification tasks. In the ViT architecture, the same concept is applied to obtain a global representation of the image, which can then be used for downstream tasks, such as image classification or action generation in robotics.

2.2 Diffusion Models

Diffusion models are a class of generative models that have recently gained significant attention in the machine learning community, especially in the field of image generation [5]. The core idea is inspired by non-equilibrium thermodynamics, and consist in learning to reverse a gradual noising process applied to the data. The diffusion process involves two main steps: the forward process, where noise is progressively added to the data, and the reverse process, where the model learns to denoise and recover the original data from the noisy version, such that at inference time, the model can generate new samples by starting from pure noise and iteratively denoising it.

The forward and reverse processes can be mathematically described as follows:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{2.3}$$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{2.4}$$

Where x_t represents the data at time step t, β_t is a variance schedule that controls the amount of noise added at each step, and μ_θ and Σ_θ are the learned mean and covariance functions parameterized by θ.

To train the model, the most common approach is to minimize the variational bound on the negative log-likelihood of the data, which can be simplified to a mean-squared error loss between the predicted noise and the actual noise added during the forward process:

$$L(\theta) = \mathbb{E}_{x_0,\epsilon,t}\left[||\epsilon - \epsilon_\theta(x_t, t)||^2\right] \tag{2.5}$$

Where x_0 is the original data, ϵ is the noise added, and ϵ_θ is the model's prediction of the noise.

It's worth noting that the diffusion process can be conditioned on additional information, such as class labels or other modalities, to guide the generation process. This is typically done by concatenating the conditioning information to the input of the model at each time step, resulting in the following modified reverse process:

$$p_\theta(x_{t-1}|x_t, c) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t, c), \Sigma_\theta(x_t, t, c)) \tag{2.6}$$

Where c represents the conditioning information.

A critical component of the diffusion model is the noise schedule, which defines how the variance schedule $\{\beta_t\}_{t=1}^T$ evolves over the diffusion steps. Common choices for noise schedules include linear, quadratic, and cosine schedules. The schedule affects the model's learning dynamics and generation quality, with well-designed schedules enabling faster convergence and better sample generation.

In this thesis, a diffusion model is employed as ViT output head to generate robot control actions, conditioned on multimodal inputs, such as visual observations and proprioceptive data. The conditioning is given by the readout token that is passed through the transformer, and it encapsulates the relevant information from the input modalities. The idea is to consider the readout token as a noisy version of the desired action, and the diffusion model learns to translate it into a clean action representation.

This approach, proposed by Ghosh *et al.* [2], has outperformed traditional regression-based methods for action prediction in robotic manipulation tasks, thanks to the ability of diffusion models to capture complex, multimodal distributions of actions.

2.3 Reinforcement Learning

Reinforcement Learning (RL) is a subfield of machine learning that focuses on training agents to make sequential decisions to maximize a cumulative reward signal provided by an external environment. The agent interacts with the environment by taking actions based on the current state, and the environment responds by providing a reward and transitioning to a new state. The goal of the agent is to learn a policy π, i.e., a mapping between the current state and the action to be taken, that maximizes the expected cumulative reward over time.

RL problems can be formally modeled as Markov Decision Problems (MDPs), which are defined by the tuple (S, A, P, R, γ), where:

- S: Set of possible states the agent can be in.
- A: Set of possible actions the agent can take.
- $P(s'|s, a)$: State transition probability function, representing the probability of transitioning to state s' given the current state s and an action a.
- $R(s, a)$: Reward function, representing the immediate reward received after taking action a in state s.
- $\gamma \in [0,1]$: Discount factor, which determines the relative importance of future rewards compared to immediate rewards.

The agent's objective is to find an optimal policy π^* that maximizes the expected cumulative reward, also known as the return, defined as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k}) \tag{2.7}$$

Where G_t is the return at time step t, and $R(s_{t+k}, a_{t+k})$ is the reward received at time step $t + k$, given the state s_{t+k} and action a_{t+k}.

To achieve this goal, it can be convenient to define the state-value function $V^\pi(s)$ and the action-value function $Q^\pi(s, a)$, which represent the expected return when starting from state s and following policy π, and the expected return when starting from state s, taking action a, and then following policy π, respectively:

$$V^\pi(s) = \mathbb{E}_\pi[G_t|s_t = s] \tag{2.8}$$

$$Q^\pi(s, a) = \mathbb{E}_\pi[G_t|s_t = s, a_t = a] \tag{2.9}$$

Given these definitions, the optimal policy can be defined as the policy that maximizes the value or the action-value function for all states and actions. So, the optimal policy yields the highest possible expected return from any given state $V^*(s)$ or state-action pair $Q^*(s, a)$.

$$V^*(s) = \max_{\pi} V^\pi(s) \tag{2.10}$$

$$Q^*(s, a) = \max_{\pi} Q^\pi(s, a) \tag{2.11}$$

In practice, estimating the value function and action-value function can be challenging, especially in high-dimensional state and action spaces. Therefore, various RL algorithms have been developed to solve MDPs by directly learning the optimal policy (without explicitly estimating value functions) or approximating the value functions, using techniques such as dynamic programming, Monte Carlo methods, and temporal-difference learning. Many of these algorithms rely on the Bellman equations, which provide a recursive relationship between the value functions and the expected rewards, and can be used to iteratively update the estimates of the value functions. The Bellman equations have the following forms:

$$V^\pi(s) = \mathbb{E}_{a\sim\pi}\left[R(s, a) + \gamma \sum_{s'} P(s'|s, a)V^\pi(s')\right] \tag{2.12}$$

$$Q^\pi(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a)\mathbb{E}_{a'\sim\pi}\left[Q^\pi(s', a')\right] \tag{2.13}$$

It is worth mentioning that, regardless of the approach used, a fundamental aspect of RL is addressing the exploration-exploitation trade-off, which is a challenge unique to RL settings. Indeed, while trying to maximize the cumulative reward, the agent must also explore the environment to discover new states and actions that may lead to higher rewards in the future. Balancing exploration and exploitation is crucial for effective learning, and various strategies, such as ϵ-greedy policies and entropy regularization have been proposed to tackle this challenge.

Finally, with the advent of deep learning, Deep Reinforcement Learning (DRL) has emerged as the most widely adopted approach to solve complex RL problems, especially in continuous high-dimensional state and action spaces. In DRL, deep neural networks are used to approximate the policy, value functions, or both, enabling the agent to learn in continuous state and action spaces, which are at the core of many real-world applications, such as robotic manipulation. In this context, the neural networks are defined as function approximators, parameterized by a set of weights θ, resulting in the notations $V^\pi_\theta(s)$, $Q^\pi_\theta(s, a)$, and π_θ, and are trained using gradient-based optimization techniques to minimize a suitable loss function. From now on, unless otherwise specified, the distinction between traditional RL and DRL will be omitted for brevity, and the term RL will refer to both paradigms.

2.3.1 Taxonomy of RL Algorithms

In this section, a brief overview of how RL algorithms can be categorized is provided, focusing on the main distinctions that are relevant to this research.

- **On-Policy vs. Off-Policy**: On-policy algorithms, such as SARSA, learn the value of the policy while following it, meaning that the data used for learning is generated by the current policy. Off-policy algorithms, such as Q-learning, learn the value of a different policy than the one used to generate the data. This allows off-policy methods to leverage data from different policies and from past experiences, making them more sample-efficient but also more complex to implement and tune.

- **Online vs. Offline**: Online RL algorithms learn and update the policy in real-time as the agent interacts with the environment. Offline RL algorithms learn from a fixed dataset of previously collected experiences, without further interaction with the environment, making them suitable for scenarios where data collection is expensive or risky.

- **Model-Based vs. Model-Free**: Model-based RL algorithms learn a model of the environment's dynamics, i.e., the state transition probabilities and the reward function, and use this model to plan actions. Model-free algorithms, on the other hand, learn a policy or value function by direct interaction with the environment, without explicitly modeling its dynamics. Model-free methods are generally more flexible and easier to implement, but they may require more data to learn effectively.

- **Value-Based vs. Policy-Based vs. Actor-Critic**: The last, and most relevant, distinction that can be made is based on the approach used to learn the optimal policy.

  - *Value-Based*: Value-based algorithms, such as DQN [6], focus on learning the action-value function $Q(s, a)$, to then maximize it at each state to obtain the optimal policy. This approach is generally more sample-efficient, but it can struggle with high-dimensional action spaces and continuous actions.

  - *Policy-Based*: Policy-based algorithms, such as PPO [7], directly learn the policy $\pi(a|s)$ without explicitly estimating the value functions. This optimization is typically done using gradient ascent on the expected cumulative reward. Policy-based methods can handle continuous action spaces and can learn stochastic policies, but they may require more data to converge and more careful tuning of hyperparameters.

– *Actor-Critic*: Actor-critic algorithms aim at combining the strengths of both value-based and policy-based methods by learning both the policy (actor) and the value function (critic) simultaneously. The actor is responsible for selecting the actions based on the current policy, while the critic evaluates the actions taken by the actor by estimating the value function. These methods can be more stable and efficient than the individual approaches, as the critic provides feedback to the actor, helping it to improve its policy over time.

To train the state-based agents for dataset generation, the focus will be on off-policy, online, model-free, actor-critic RL algorithms, with a particular emphasis on the SAC algorithm, which will be described in detail in the next section.

2.3.2 Soft Actor-Critic Algorithms

Soft Actor-Critic (SAC) is an off-policy, model-free, actor-critic RL algorithm that optimizes a stochastic policy, aiming at maximizing a trade-off between expected cumulative reward and entropy [8].

In the context of RL, entropy is a measure of uncertainty or randomness in the policy's action selection. The entropy of a policy can be defined for a given state s as:

$$\mathcal{H}(\pi(\cdot|s_t)) = -\mathbb{E}_{a_t\sim\pi}[\log \pi(a_t|s_t)] \tag{2.14}$$

Where $\mathcal{H}(\pi(\cdot|s_t))$ is the entropy of the policy at state s_t, the expectation is taken over the actions a_t sampled from the policy $\pi(a_t|s_t)$, and $\log \pi(a_t|s_t)$ is the logarithm of the probability of taking action a_t in state s_t according to the policy. The entropy is maximal when the policy selects actions uniformly at random, and minimal (zero) when the policy is deterministic, always selecting the same action for a given state.

By incorporating an entropy term into the objective function, SAC encourages exploration and prevents premature convergence to suboptimal policies. Indeed, the agent is incentivized to maintain a certain level of randomness in its action selection, which can help in discovering better strategies and avoiding local optima. The modified objective function can be expressed as:

$$J(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t (R(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t)))\right] \tag{2.15}$$

Where α is a temperature parameter that controls the trade-off between reward maximization and entropy, and that can also be actively adjusted during training [9]. In particular, its active adjustment allows the agent to adapt its exploration strategy over time, starting with a higher entropy to encourage exploration in the early stages of training, and gradually reducing it as the agent becomes more confident in its learned policy, making the entropy converge to a target value.

In the following, the foundational framework of soft policy iteration is presented, along with the practical implementation details of the SAC algorithm.

Soft Policy Iteration

The SAC algorithm is based on the soft policy iteration framework, which consists in the iterative application of two main steps:

- **Soft Policy Evaluation**: In this step, the action-value function $Q^\pi(s, a)$ and the state-value function $V^\pi(s)$ are estimated and updated based on the current policy π. The update is performed using the soft Bellman equation, with the inclusion of the entropy term in the value functions:

$$Q^\pi(s_t, a_t) = R(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim P(\cdot|s_t, a_t)} \left[V^\pi(s_{t+1}) \right] \tag{2.16}$$

$$V^\pi(s_t) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t)} \left[Q^\pi(s_t, a_t) - \alpha \log \pi(a_t|s_t) \right] \tag{2.17}$$

- **Soft Policy Improvement**: In this step, the policy π is updated to maximize the expected cumulative reward, using the action-value function as a guide. The policy update is performed by minimizing the Kullback-Leibler divergence between the current policy and the distribution induced by the action-value function:

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} D_{\text{KL}} \left(\pi'(\cdot|s_t) \,\middle\|\, \frac{\exp\left(\frac{1}{\alpha} Q^{\pi_{old}}(s_t, \cdot)\right)}{Z^{\pi_{old}}(s_t)} \right) \tag{2.18}$$

  Where $Z^{\pi_{old}}(s_t)$ is a normalization constant ensuring that the right-hand side forms a valid probability distribution.

The soft policy iteration process is repeated until convergence, resulting in an optimal policy that balances reward maximization and exploration through entropy regularization. Indeed, the term “soft” in the name of the framework explicitly refers to the inclusion of the entropy term in the policy evaluation and improvement steps, which encourages the agent to maintain a certain level of randomness in its action selection.

Soft Actor-Critic Implementation

The practical implementation of the SAC algorithm involves the use of function approximators, typically deep neural networks, to represent the policy and value functions, as previously discussed when introducing DRL.

In the following, the main components with their respective loss functions are described:

- **Critic Networks**: Two separate neural networks, $Q_{\theta_1}(s,a)$ and $Q_{\theta_2}(s,a)$, are used to approximate the action-value function. The estimated Q value is given by the minimum of the values returned by the two networks to mitigate overestimation bias, since taking the minimum helps to provide a more conservative estimate of the action-value function, removing positive biases that can arise from function approximation errors:

$$Q(s_t, a_t) = \min\left(Q_{\theta_1}(s_t, a_t), Q_{\theta_2}(s_t, a_t)\right) \tag{2.19}$$

The two critic networks are trained to minimize the expected squared Bellman error, using a replay buffer D to store past experiences. In particular, they are updated by minimizing the error between the predicted Q value and a target Q value y_t computed using two target networks, $Q_{\bar{\theta}_1}(s,a)$ and $Q_{\bar{\theta}_2}(s,a)$, which are delayed copies of the critic networks, kept to stabilize training. The resulting loss function for each critic network is given by:

$$J_Q(\theta_i) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim D}\left[\left(Q_{\theta_i}(s_t,a_t) - y_t\right)^2\right] \tag{2.20}$$

$$y_t = r_t + \gamma \mathbb{E}_{a_{t+1}\sim\pi_\phi(\cdot|s_{t+1})}\left[Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) - \alpha \log \pi_\phi(a_{t+1}|s_{t+1})\right] \tag{2.21}$$

Where y_t is the target Q value computed using the target networks, and $Q_{\bar{\theta}}$ represents the Q value estimated by the two target networks. The target networks' parameters $\bar{\theta}_i$ are updated using a soft update mechanism, as follows:

$$\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i \tag{2.22}$$

Where $\tau \in$ [0,1] is a hyperparameter that controls the update rate.

- **Actor Network**: A separate neural network, $\pi_\phi(a|s)$, is used to represent the policy. The actor network is trained by minimizing the expected KL divergence between the current policy and the distribution induced by the action-value function. This is equivalent to maximizing the expected Q value while also considering the entropy of the policy, leading to the following loss function:

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim D}\left[\mathbb{E}_{a_t\sim\pi_\phi(\cdot|s_t)}\left[\alpha \log \pi_\phi(a_t|s_t) - Q_\theta(s_t, a_t)\right]\right] \tag{2.23}$$

Since the policy is stochastic, actions are typically sampled from a Gaussian distribution parameterized by the actor network, which outputs the mean $\mu_\phi(s_t)$ and standard deviation $\sigma_\phi(s_t)$ of a standard Normal distribution, for each state s_t. The actions are then sampled using the reparameterization trick [10], and then transformed using a squashing function (e.g., tanh) to ensure that they lie within a valid range.

$$a_t = f_\phi(s_t; \epsilon_t) = \tanh(\mu_\phi(s_t) + \sigma_\phi(s_t) \odot \epsilon), \quad \epsilon \sim \mathcal{N}(0, I) \tag{2.24}$$

- **Temperature Parameter**: The temperature parameter α can be either fixed or learned during training, to actively balance the exploration-exploitation trade-off. When learned, it is updated by minimizing the following loss function, which encourages the policy's entropy to match a target entropy value $\mathcal{H}_{\text{target}}$:

  $$J_\alpha(\alpha) = \mathbb{E}_{a_t \sim \pi_\phi(\cdot|s_t)} \left[-\alpha \left(\log \pi_\phi(a_t|s_t) + \mathcal{H}_{\text{target}} \right) \right] \tag{2.25}$$

  With this formulation, the temperature parameter α is adjusted to ensure that the policy maintains a desired level of entropy, promoting exploration while still focusing on reward maximization.

Given the components and loss functions described above, the SAC algorithm can be summarized in the steps presented in Algorithm 1.

Algorithm 1 Soft Actor-Critic (SAC) [9]

```
Input: $Q_{\theta_1}$, $Q_{\theta_2}$, $\pi_\phi$, $\theta_1$, $\theta_2$, $\phi$      ▷ Q-function and policy with initial parameters
 1: $\bar{\theta}_1 \leftarrow \theta_1$, $\bar{\theta}_2 \leftarrow \theta_2$      ▷ Initialize target networks
 2: $\mathcal{D} \leftarrow \emptyset$      ▷ Initialize empty replay buffer
 3: for each iteration do
 4:     for each environment step do
 5:         $a_t \sim \pi_\phi(a_t|s_t)$      ▷ Sample action from policy
 6:         $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$      ▷ Sample next state from environment
 7:         $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$      ▷ Store transition
 8:     end for
 9:     for each gradient step do
10:         $\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta_i} J_Q(\theta_i)$ for $i \in \{1,2\}$      ▷ Update Q-functions
11:         $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi J_\pi(\phi)$      ▷ Update policy
12:         $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha J(\alpha)$      ▷ Adjust temperature
13:         $\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$ for $i \in \{1,2\}$      ▷ Update target networks
14:     end for
15: end for
Output: $\theta_1, \theta_2, \phi$      ▷ Optimized parameters
```

2.4 Imitation Learning

Imitation Learning (IL), or Learning from Demonstrations (LfD), is a paradigm in machine learning where an agent learns to perform tasks by mimicking expert demonstrations, rather than through trial-and-error interactions with the environment as in RL. The main objective of IL is to learn a policy that can replicate a given set of expert behaviors, enabling the agent to perform tasks without the need for explicit reward signals.

The most straightforward approach to IL is Behavioral Cloning (BC), which involves training a policy to directly map states to actions based on the expert demonstrations. This means that the agent learns to predict the expert's actions given the observed states, effectively treating the problem as a supervised learning task. The policy, parameterized by θ, is trained to minimize the following loss function:

$$\mathcal{L}_{\text{BC}}(\theta) = \mathbb{E}_{(s,a)\sim D}\left[||a - \pi_\theta(s)||^2\right] \tag{2.26}$$

Where D is the dataset of expert demonstrations, consisting of state-action pairs (s, a), and $\pi_\theta(s)$ is the policy's predicted action for state s.

The main advantage of BC is its simplicity and efficiency, as it does not require additional labeling or reward engineering, while still enabling the model to learn complex behaviors directly from expert data. However, BC suffers from several limitations, such as compounding errors, covariate shift, and difficulty in generalizing to unseen contexts. All these issues arise from the fact that the policy is trained only on the states present in the expert demonstrations, and may not be able to handle situations that deviate from those seen during training, in some cases leading to a divergence from the expert behavior due to error accumulation over time.

In this thesis, a variant of BC is employed as the main training signal for the transformer-based models, due to its effectiveness in learning complex policies from high-dimensional inputs, and since it allows for faster and more stable training compared to RL approaches. To alleviate some of the limitations of BC, some of the most common techniques are employed, such as data augmentation, regularization, and the use of large and diverse datasets, which help improve the model's generalization capabilities and robustness to distributional shifts. Additionally, different samples are weighted differently during training, to prioritize more relevant or informative demonstrations. Further details on the specific implementation choices will be provided in the subsequent chapters.

2.5 Advanced Learning Paradigms

In this section, two advanced learning paradigms that are relevant to this research are briefly introduced: representation learning and multi-task learning. Furthermore, cross-embodiment learning, which can be seen as a specific case of multi-task learning, is also discussed.

2.5.1 Representation Learning

Representation learning is a subfield of machine learning that focuses on learning useful and informative representations from raw data, which can be used to improve the performance of downstream tasks, such as classification, regression, or control [11]. The main idea behind representation learning is to automatically discover features or embeddings that capture the underlying structure and patterns in the data, enabling the model to generalize better and learn more efficiently.

In the context of robotics and control, representation learning refers to the process of learning compact and meaningful representations of sensory inputs, such as images, proprioceptive data, or other modalities, that can be used to inform decision-making and action selection. In particular, the goal is to structure the learned representations in a way that captures the relevant information for the task at hand, while also being invariant to irrelevant factors, such as noise, distractions, or disturbances.

One of the prominent approaches to representation learning is through the employment of contrastive learning techniques. The core idea of contrastive learning is to learn models that map similar inputs to nearby points (positive pairs) in the feature space, while mapping dissimilar inputs (negative pairs) to distant points. In supervised contrastive learning, the positive and negative pairs are determined based on class labels or other annotations. For example, in an image classification task, positive pairs could be images displaying the same object with different viewpoints or lighting conditions, while negative pairs could be images of different objects.

In this work, representation learning techniques are utilized to learn robust and transferable visual features from a dataset containing diverse trajectories collected from multiple robotic embodiments and tasks.

2.5.2 Multi-Task Learning

Multi-Task Learning (MTL) is a machine learning paradigm where a single model is trained to perform multiple related tasks simultaneously. MTL aims to leverage shared information and representations across tasks to improve the overall performance, data efficiency, and generalization capabilities of the model. This is particularly relevant in robotics, where an agent may need to perform a variety of tasks that share common skills or knowledge. Indeed, even if the tasks differ in their specific objectives or environments, they may still benefit from leveraging common underlying structures or features derived from shared sensory inputs or action spaces.

The advantages of MTL have been demonstrated in both RL [12] and IL. In IL, several studies show the viability of generalist agents, as discussed in Section 3.2, including the agent used in this thesis [2], and the benefit of leveraging models pretrained on diverse datasets for downstream tasks [13, 14].

Cross-Embodiment Learning

Cross-embodiment learning can be seen as a specific case of MTL, where the different tasks correspond to performing similar behaviors using different embodiments. The main challenge in cross-embodiment learning is to learn representations and policies that are transferable, but still accurate. Indeed, different robots may have varying kinematics, dynamics, and sensory modalities, that can significantly affect input signals and solving strategies. To overcome this issues, the model must capture the underlying structure of the tasks and how it relates to the different embodiments, while also being able to adapt to the specific characteristics of each robot.

A broader perspective on cross-embodiment learning is provided in Section 3.3, where related works are discussed in detail. Additionally, since cross-embodiment learning is the main focus of this thesis, specific details will be provided in all the subsequent chapters.

Lastly, it is worth mentioning that the vision-based policy employed in this thesis aim at achieving MTL capabilities both across different tasks and embodiments.

Chapter 3

State of the Art

This chapter provides an overview of the existing literature and research that motivates and contextualizes the work presented in this thesis. The discussion aims at positioning this research within the landscape of robotic manipulation learning, with a particular focus on generalization across different embodiments.

The chapter is structured as follows: the first section reviews the key concepts for robotic manipulation learning, with a particular focus on diffusion models and grasping techniques. Then, in the second section, generalist agents are discussed, highlighting their relevance to the proposed approach. The third section delves into cross-embodiment learning, exploring various strategies to address each of the challenges involved. Finally, the last section examines methods that leverage demonstrations for policy learning, dividing them between contributions relevant to the state-based agents and to the vision-based agent proposed in this thesis.

3.1 Robotic Manipulation Learning

Robotic manipulation has been a central topic in robotics learning research. It involves some unique challenges, such as high-dimensional multimodal states, complex dynamics, and the need for precise control in continuous action spaces.

One of the foundational approaches to tackle robotic manipulation learning is proposed by Zhao *et al.* [15], which introduces a supervised learning framework that leverages action chunking and temporal ensembling to improve the training and inference of transformer-based visuomotor policies. Action chunking consists in predicting a sequence of future actions instead of a single action at each time step, which helps the model to capture temporal dependencies and plan ahead. Temporal ensembling, on the other hand, involves aggregating predictions over multiple time steps to obtain the final action, which enhances robustness and smoothness in the generated behaviors.

Furthermore, their contribution proved the effectiveness of transformer-based architectures for robotic manipulation tasks and highlighted the importance of modeling temporal dependencies in action sequences to achieve coherent and effective behaviors.

3.1.1 Diffusion Models for Robotic Manipulation

Building on Zhao *et al.* [15] foundation, subsequent research has investigated alternative methods to ensure action coherence. A notable example involves diffusion models, shown to be effective by Chi *et al.* [16] for visuomotor policy learning in robotic manipulation. Their method employs a diffusion-based policy framework to model temporal dependencies in action sequences, enabling flexible manipulation strategies that can predict multi-modal action distributions, while being more robust to noisy and suboptimal demonstrations. Subsequent works have further explored the potential of diffusion models, proving their effectiveness in learning different tasks simultaneously [17], adapting to new environments [18], and utilizing 3D representations [19].

These advancements highlight the potential of diffusion models in enhancing the generalization capabilities of robotic manipulation policies and the ability to capture complex action distributions. Nevertheless, these methods often require substantial computational resources and, in particular, they may not be suitable for real-time applications due to the high overhead introduced by the diffusion process at inference time. For the vision-based agent proposed in this work, a lightweight diffusion-based action decoder is employed [2], which is specifically designed to reduce the computational burden while maintaining the benefits of diffusion models for action generation.

3.1.2 Grasping

Grasping is a fundamental aspect of robotic manipulation, as it is often the most critical step in interacting with objects, since it requires precise control and coordination of the robot's end-effector. In the context of cross-embodiment learning, grasping presents additional challenges, as different robotic platforms may have varying gripper designs, with different kinematics and grasping strategies.

The literature on robotic grasping can be broadly categorized into two main approaches: grasping pose detection and end-to-end learning. Grasping pose detection methods focus on identifying optimal grasping points on objects, often given a 3D representation of the scene and a specific gripper model, to then execute the grasp using a predefined control strategy [20, 21], while end-to-end learning approaches aim at directly mapping sensory inputs to grasping actions, mainly using deep learning techniques [22, 23]. In this thesis, the focus is on end-to-end learning methods, as they are simpler to integrate into a generalist agent framework.

Regarding cross-embodiment learning, recent studies have investigated the development of gripper-independent grasping policies. Notable examples include Unigrasp [24] that cast grasping as a sequence of contact point prediction till the number of fingers of the gripper is reached, AdaGrasp [25] which aims at learning adaptive gripper-aware grasping policies through a cross-convolution operation between visual features and gripper attributes, and NeuralGrasps [26] that leverages implicit representations built by leveraging contact point similarity across grippers. Even though these methods have shown promising results in generalizing across different grippers, they require a grasp execution module to be paired with the learned policy or specific datasets for training.

The key takeaways drawn from these works are that gripper-aware representations are crucial for achieving successful graspings, especially when dealing with diverse gripper designs, and that learning such representations usually requires high-dimensional data, such as 3D point clouds or meshes. In this thesis, those insights are used to provide the vision-based agent with a gripper-aware representation that can be integrated into a generalist agent for cross-embodiment learning.

3.2 Generalist Agents

Following the success of LLM in various language tasks, researchers have explored the potential of generalist agents in robotics. Those models have demonstrated remarkable generalization capabilities across a wide range of tasks [27], while being able to leverage vast amounts of data to learn complex patterns, while scaling performance with model size [28].

In robotics, several works have explored the potential of generalist agents for manipulation tasks. One of the first attempts in this direction is Gato [29], a transformer-based model trained on a diverse dataset of tasks, including robotic manipulation, game playing, and language processing. Gato demonstrated the ability of a single model to exploit knowledge across different tasks and modalities, showcasing the potential of generalist agents.

Later works introduced more specialized models, such as RT-1 [30], a vision-based transformer model trained on a large-scale dataset of robotic manipulation tasks, and RoboCat [31], a self-improving generalist agent that leverages both imitation learning and reinforcement learning to enhance its performance over time and generalize to new embodiments. More recently, FP3 [32] proved the effectiveness of leveraging 3D representations to enhance the generalization capabilities of a generalist agent. These models have shown promising results in various manipulation tasks, even with only language instructions as input, but they often require large-scale datasets and significant computational resources for training and inference due to their complexity.

To address these challenges, some works have focused on leveraging smaller and faster architectures. In this thesis, the Octo model, proposed by Ghosh *et al.* [2], was selected as the backbone of the vision-based policy due to its reduced computational requirements, openly available codebase and weights, and its flexible fine-tuning pipeline. It employs a transformer-based architecture, inspired by the ViT-Small architecture [3], with a lightweight diffusion head for action generation. Octo is pretrained on a subset of the Open X-Embodiment dataset [33] and demonstrates strong generalization and fine-tuning capabilities across different robotic platforms and tasks.

Another aspect that is worth mentioning in this context is the datasets used to train generalist agents. In recent years, Robonet [34] and the Open X-Embodiment dataset [33] are two of the most notable examples of open-source, large-scale datasets for robotic manipulation. The dataset plays a crucial role, as it needs to be as representative as possible to have an efficient task learner agent. Indeed, in this work, a comprehensive multi-embodiment dataset is generated to effectively finetune the vision-based generalist agent proposed.

3.3 Cross-Embodiment Learning

As introduced in Section 2.5.2, cross-embodiment learning aims at transferring knowledge and skills across different robotic platforms with varying morphologies. This area of research has gained significant attention due to the increasing diversity of robotic systems and the consequent need for adaptable and versatile control policies to avoid expensive retraining for each new robot. Nevertheless, cross-embodiment learning remains a complex problem, as it involves addressing several challenges, including differences in appearance, perception systems, dynamics, and morphologies [35].

To tackle these challenges, researchers have proposed various approaches, of which the most relevant to this thesis are discussed in the following. In particular, four main categorizations are identified, based on the primary focus of the methods: knowledge transfer, skill transfer, dynamics gap bridging, and vision gap bridging. The insights drawn from these works have been primarily used to design the vision-based agent proposed.

3.3.1 Knowledge Transfer

Generalist agents, introduced in Section 3.2, form the backbone of most of the knowledge transfer approaches, which aim at leveraging knowledge acquired from one embodiment to improve the learning process on a different one, even when considering significantly different settings.

COMPASS [36], CrossFormer [37] and the cross-embodiment policy proposed by Yang *et al.* [38] are valuable examples of this category. They propose architectures trained via IL or a combination of IL and RL to learn generalist policies that can perform significantly different tasks, e.g., manipulation and navigation, across different embodiments. In particular, Yang *et al.* [38] utilizes a transformer-based architecture with a diffusion head to achieve strong cross-task knowledge transfer, similarly to the architecture adopted in this thesis. All those methods serve as practical evidence of the potential of knowledge transfer approaches in cross-embodiment learning, even if in this thesis the focus is on manipulation tasks only.

3.3.2 Skill Transfer

Another set of approaches focuses on skill transfer, which differs from knowledge transfer since the main goal is to transfer specific skills or behaviors learned in one embodiment to another, providing approaches for more specific settings. The methods in this category often aim at learning a shared latent representation that captures the essential features of different skills across various embodiments, enabling their transfer between robots with diverse morphologies. To achieve this goal, representation learning techniques, introduced in Section 2.5.1, are often employed.

For instance, Zakka *et al.* [39] proposes XIRL, which aims at learning embodiment-invariant visual representation through temporal cycle-consistency learning, enabling the transfer of skills between different robotic platforms. Similarly, XSkill [40] and R3M [41] leverage InfoNCE (Noise Contrastive Estimation) [42] time constrative losses to learn a shared latent space for skill transfer, such that similar skills across different embodiments are mapped to nearby points in the latent space. Finally, Wang *et al.* [43] proposes a multiple adversarial training framework to align different encoders and decoders for different embodiments.

Those approaches have shown promising results in transferring skills across different robotic platforms, even if they may struggle in dealing with challenging or fine-grained manipulation tasks, as they may lack embodiment-specific details. In this thesis, the main insight drawn is that representation learning is a key enabler for cross-embodiment skill transfer; nevertheless, similar difficulties are observed and discussed in the experimental chapter.

3.3.3 Dynamics Gap Bridging

Another approach to cross-embodiment learning focuses on bridging the dynamics gap between different robotic platforms. This involves learning dynamic models that can generalize across various embodiments, often by leveraging the transformer architecture due to its flexibility.

In the literature, several works have explored the possibility of modeling joints as tokens, translating the action prediction problem into a sequence modeling problem. Notable examples include MetaMorph [44] and Meta-Controller [45] that retain a stronger connection between the tokens and the robot's morphology, while AnyMorph [46] and MAT [47] employ a more abstract representation of the robot's joints to enhance the flexibility of the model.

These approaches exhibit strong generalization capabilities across different embodiments, but they are focused on locomotion tasks, which hinder significant dynamic and kinematics differences between the robots, requiring a strong joint-level understanding. In contrast, in manipulation tasks, the differences in joint configurations do not affect the end-effector behavior as much, since Inverse Kinematics (IK) solvers can be used to map the end-effector actions to the robot's joints, so it is not worth using such computationally expensive methods. Nevertheless, key insights from these works can be leveraged; indeed Bohlinger *et al.* [48] recently demonstrated the effectiveness of preprocessing the robot's joint states as tokens before feeding them to a transformer-based model, which also processes general observations to predict end-effector actions. In this thesis, a similar approach is adopted, but the focus is on the mutual interaction between the robot's gripper and the object to be manipulated, rather than on the robot's joints.

3.3.4 Vision Gap Bridging

The last aspect that is worth mentioning in the context of cross-embodiment learning is the vision gap bridging, which aims at addressing the differences in visual perception across different robotic platforms. This is crucial since for vision-based policies, the visual input varies significantly between different arms and grippers, in terms of colors, shapes, and sizes.

A class of methods that has been explored in this context aims at directly addressing the visual domain gap by isolating the robotic arm from the background to then separately process the two components [49], or to then substitute the original arm with the current robot's arm [50]. Even though these methods have shown promising results, they require complex image processing pipelines, and they may struggle to generalize to unseen environments.

Alternative approaches propose data-centric solutions, such as Dasari *et al.* [51] that investigate the impact of different datasets for pertaining visuo-motor policies, aiming at addressing the visual domain gap by leveraging large-scale and diverse datasets. Polybot [52], on the other hand, leverages contrastive learning techniques to learn visual representations conditioned on the robot's end-effector state rather than on the robot's appearance. This thesis is more related to the latter approaches, as pretrained models are leveraged to extract visual features, while the gap is bridged through contrastive losses.

3.4 Leveraging Demonstrations for Policy Learning

Leveraging demonstrations for learning control policies is often more efficient than learning from scratch through RL, especially in robotic manipulation, where exploration can be challenging and time-consuming. Additionally, demonstrations can provide valuable prior knowledge about the task, guiding the learning process and improving sample efficiency.

In the next two sections, methods relevant to the state-based agents (3.4.1) and to the vision-based agent (3.4.2) proposed in this thesis are discussed separately, as they pertain to different learning paradigms, namely RL and IL.

3.4.1 Reinforcement Learning with Demonstrations

In the RL context, the most straightforward approaches to leverage demonstrations are offline RL methods, which aim at learning policies from fixed datasets of demonstrations without any environment interaction. Notable examples include AWR [53], which performs advantage-weighted regression to extract high-return actions from the offline data, and more complex methods that aim at mitigating extrapolation errors by constraining the learned policy to remain close to the demonstrations while maximizing a learned Q-function [54, 55, 56]. These offline RL methods generally achieve strong performance, but they cannot bridge huge gaps between the demonstrations and the target policy, like in the case of demonstrations collected with a different embodiment.

To overcome the limitations of offline learning, at the cost of requiring environment interaction, several works have proposed hybrid offline-online RL. Advantage Weighted Actor-Critic (AWAC) [57] is a notable example that utilizes an advantage-weighted likelihood objective to effectively combine offline demonstrations with an online actor-critic RL algorithm. On the other hand, several approaches [58, 59, 60] investigate practical design choices to reliably mix offline and online learning, demonstrating that even simple methods can achieve strong performance when appropriately configured. These hybrid approaches generally outperform both pure offline and pure online methods, but they often require careful tuning to balance the contributions of the offline data and the online interactions.

In this work, the state-based agent presented will be trained using a hybrid offline-online RL approach, leveraging an AWAC-style BC loss and practical insights drawn from the second class of methods to improve sample efficiency and overall performance. Indeed, the additional challenge is to bridge the domain gap introduced by the fact that the demonstrations are collected with a different embodiment with respect to the one used in the policy learning phase.

3.4.2 Weighted Imitation Learning

Imitation Learning (IL) is a widely used paradigm for learning control policies from demonstrations. However, standard IL methods assume that the demonstrations are optimal, which is often not the case in practice. To address this limitation, several works have proposed weighted Imitation Learning methods, which assign different weights to the demonstrations based on their quality or relevance to the target task [61, 62]. This allows the model to focus more on high-quality demonstrations while still leveraging the information present in suboptimal ones, and at the same time maintains the simplicity and efficiency of standard IL.

Nevertheless, these methods often require additional labeling, estimation of demonstration quality, or access to expert demonstrations. Additionally, they involve the use of additional components to estimate the weights, which introduce further complexity to the training process. In this thesis, an easier approach to weighing the demonstrations is adopted to overcome these limitations by leveraging the information already provided by the state-based agents.

Chapter 4

Methodology

This chapter describes the methodology adopted to obtain a vision-based multi-embodiment policy via supervised finetuning of a pre-trained generalist agent. The choice of employing a vision-based policy is motivated by the need for embodiment generalization, especially when considering novel embodiments not seen during training. Indeed, vision can provide rich contextual information about the environment and the robot's interaction with it, without relying on explicit state representations that are usually less transferable due to embodiment-specific variations that are difficult to derive from raw sensory data. Furthermore, vision-based policies can better adapt to real-world scenarios where precise state information may not be available.

The motivation behind the decision to use supervised finetuning for training the vision-based multi-embodiment policy is driven by several factors. Supervised finetuning is generally more sample-efficient and less computationally demanding than, for instance, RL approaches, which is particularly important when dealing with high-dimensional visual inputs that require large amounts of data to learn effective policies. Additionally, finetuning a pre-trained model allows leveraging prior knowledge and representations learned from large-scale datasets, which can significantly accelerate the learning process and improve the final performance.

A direct consequence of choosing supervised finetuning is the requirement for a large, diverse dataset of successful demonstrations that covers the target robot-gripper-task triplets. To generate this dataset, a two-stage pipeline is adopted. First, state-based agents are trained per triplet using RL accelerated with expert demonstrations and a re-rolling procedure that adapts demonstrations across embodiments. Second, the resulting trained agents are executed in simulation to collect multi-view RGB observations, proprioception, actions, and metadata from successful runs; these executions form the multi-embodiment demonstration dataset. The collected dataset is then used to supervisedly finetune a pre-trained vision model into a multi-embodiment policy, leveraging contrastive losses and weighted behavioral cloning to improve embodiment generalization.

For the dataset generation, the choice of leveraging RL is motivated by the need to obtain an arbitrary number of successful demonstrations for each robot-gripper-task triplet, which may not be feasible with human teleoperation due to time and resource constraints. In this case, state-based agents are preferred since the single embodiment setting is less challenging, allowing for the utilization of low-dimensional state representations that facilitate the learning process and improve sample efficiency. Additionally, the choice of covering all possible robot-gripper combinations for each task is motivated to maximize the diversity of the dataset, forcing the vision-based policy to learn embodiment-invariant features and behaviors.

The remainder of the chapter details each step of this pipeline: Section 4.1 describes state-based training and demo re-rolling; Section 4.2 explains dataset collection and filtering; and Section 4.3 presents the vision model, together with its input modalities, the supervised finetuning losses, and the strategies employed to achieve embodiment generalization.

4.1 State-Based Agents

The training of state-based agents is performed in simulation using a modified version of the Soft Actor-Critic [8, 9], introduced in Section 2.3.2. In particular, to accelerate the learning process and improve the final performance, the RL training is aided by leveraging publicly available expert demonstrations for each task. The expert demonstrations are human teleoperated trajectories collected in simulation with only one particular robot and gripper embodiment.

The key observation enabling the proposed approach is that, although the expert demonstrations are collected with a specific robot-gripper embodiment, if the state and action spaces are properly aligned, they can still provide useful information to train agents with different robot-gripper embodiments for the same task. This is because, regardless of the specific embodiment used to collect the demonstrations, the underlying task dynamics remain the same. In particular, given a proper alignment, it may be possible to solve the task using an action sequence similar to the one collected with a different embodiment.

Building upon this observation, the action sequences from the expert demonstrations collected with a specific robot-gripper embodiment are adapted and re-rolled to obtain a set of, possibly sub-optimal, demonstrations for every other robot-gripper embodiment selected for the task. Even though the re-rolled demonstrations may not be optimal for the new embodiment, and they provide a way lower success rate compared to the original ones, they can still significantly aid the RL training process. Indeed, in complex task scenarios, the ability to leverage even sub-optimal demonstrations can be the key to successfully learn a policy, since it allows the agent to explore more relevant areas of the state space. Furthermore, pathological behaviors, such as reward hacking, are less likely to occur when the agent can rely on demonstrations to guide its learning process, rather than solely on the reward signal.

4.1.1 State and Action Spaces Alignment

To perform the re-rolling, the state and action spaces must be aligned across different robot-gripper embodiments.

Regarding the state space, only the robot-agnostic components are retained, such as the end-effector position and orientation, the object position and orientations, and the gripper state, while robot-specific components, such as joint positions and velocities, are discarded. Nevertheless, this is a common practice in state-based robotic control policies, as the aforementioned robot-agnostic components are usually sufficient to successfully solve the task.

Particular attention must be given to the gripper state, as it is crucial to learn a successful policy, but it varies in the number of dimensions across different gripper types. Since different gripper types have a different number of fingers, but also a different representation of the finger states, a representation mapping is defined to adapt the gripper state from the pre-existing expert demonstrations to the new gripper type. Assuming that the representation is based on finger opening levels, and that the original demonstrations have the smallest number of elements in the gripper state vector, the gripper state is adapted as follows: while the number of additional dimensions is divisible by the original number of dimensions, the original gripper state values are replicated to fill the new dimensions, then any remaining dimensions are filled with the average value of the states from the original demonstration. For example, the 2-dimensional gripper state [0.2, 0.4] would be adapted to the 5-dimensional [0.2, 0.4, 0.2, 0.4, 0.3], where 0.3 is the average of 0.2 and 0.4. Some noise is also added to increase variability in the adapted demonstrations.

Even if this adaptation is highly simplistic, and it may not be accurate in representing the actual gripper state, in practice, it was found to still provide useful information to the learning process. Indeed, this design choice, instead of just considering the gripper opening level as a scalar between fully closed and fully opened, allows for better capturing the state of the gripper in the subsequent learning process, providing a more precise representation of the gripper's configuration. Additionally, it is worth noting that the expert demonstrations are progressively replaced in the replay buffer during training, as the agent collects new experiences, so any inaccuracies in the adapted demonstrations are eventually overcome by the agent's own exploration of the environment.

Concerning the action space, a similar approach is followed. The action is expressed in terms of end-effector movement difference with respect to the current position, represented as a 3D translation and angular displacement, and of gripper opening percentage. This representation is independent of the robot and gripper embodiment, allowing for direct reuse of the action sequences from the expert demonstrations without any modification. Moreover, the IK solver provided by the simulation environment is consistent across different robot-gripper embodiments, allowing for more similar behaviors when executing the same end-effector trajectories.

4.1.2 SAC with Advantage-Weighted BC

The re-rolling process is performed by executing the action sequences from the expert demonstrations in the simulation environment, using a different robot-gripper embodiment. To increase the likelihood of successfully completing the task, after positioning the end-effector and the object according to the demonstration initial state, position adjustments and, eventually, random variations are applied to the end-effector position and orientation to take into account the differences in dimensions and kinematics across different grippers.

Given the adapted trajectories for every robot-gripper embodiment selected for the task, the state-based agents are trained using the SAC algorithm with a BC term added to the actor loss, similarly to the approach presented by Lu *et al.* [60]. However, in this work, the BC term uses a AWAC-style [57] advantage weighting to utilize the information given by the sub-optimal adapted demonstrations only when it is beneficial to maximize the expected return. In particular, the advantage is computed between the current Q-value associated with the state-action pair from the demonstration and the approximate value of the current state. To obtain such an estimate, the value function is approximated as the Q-value of the average action sampled from the current policy, as also proposed by Nair *et al.* [57]:

$$\begin{aligned}\hat{A}^{\pi_k}(s,a) &= Q^{\pi_k}(s,a) - \hat{V}^{\pi_k}(s) \\ &= Q^{\pi_k}(s,a) - \mathbb{E}_{a'\sim\pi_k(\cdot|s)}\left[Q^{\pi_k}(s,a')\right]\end{aligned} \tag{4.1}$$

In practice, the action used to compute the second term is the mean action $\mu_\phi(s)$ given by the current policy for the specific state. This formulation allows for identifying whether the action from the demonstration is better or worse than the average action proposed by the current policy in the same state. Thus, the BC term will encourage the policy to imitate actions from the demonstrations only when they are estimated to be better than the average action from the current policy, while discouraging their imitation otherwise.

Furthermore, the inclusion of the AWAC objective, presented in Equation 4.2, constraints the policy to stay close to the behavioral cloning policy, enforcing a more stable learning process and a more structured exploration of the environment. To not bias the learning process too much towards sub-optimal behaviors, the BC term is linearly annealed during training, eventually being completely removed after a certain number of training steps.

$$\begin{aligned} \pi_{k+1} = \arg\max_{\pi} \mathbb{E}_{a\sim\pi(\cdot|s)} \left[A^{\pi_k}(s,a)\right] \\ \text{s.t. } D_{\text{KL}}\left(\pi(\cdot|s) \| \pi_{BC}(\cdot|s)\right) \leq \epsilon \end{aligned} \tag{4.2}$$

The final actor loss used for training the state-based agents is thus the combination of the standard SAC actor loss and the AWAC-style BC term, resulting in the following expression:

$$\begin{aligned} J_{\pi,\text{BC}}(\phi) = \mathbb{E}_{s_t\sim\mathcal{D}} \left[\mathbb{E}_{a_t\sim\pi_\phi(\cdot|s_t)} \left[\alpha \log \pi_\phi(a_t|s_t) - \min_{i=1,2} Q_{\theta_i}(s_t, a_t)\right]\right] \\ + \lambda_{\text{BC}}(t)\, \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{demo}}} \left[\log \pi_\phi(a|s) \exp\left(\frac{1}{\beta}\hat{A}^{\pi_k}(s,a)\right)\right] \end{aligned} \tag{4.3}$$

Where $\lambda_{\text{BC}}(t)$ is the time-dependent weight of the BC term, which is linearly annealed during training, β is a temperature hyperparameter that controls the sharpness of the advantage weighting, $\mathcal{D}$ is the replay buffer, and $\mathcal{D}_{demo}$ is the set of adapted demonstration transitions.

Regarding the replay buffer, the adapted transitions are kept in a separate buffer $\mathcal{D}_{\text{demo}}$ for computing the BC term, but are also included as initialization to the main one $\mathcal{D}$ to further aid the learning process. In this case, to avoid overfitting to the demonstrations, the capacity of the replay buffer $\mathcal{D}$ is lower than the total number of training steps to ensure that the agent can explore the environment and collect new experiences, eventually replacing the adapted transitions.

The remainder of the SAC training procedure follows the standard approach presented in Section 2.3.2, with adaptive temperature adjustment enabled.

For reproducibility and comparability with other works, the training is performed using the default task settings provided by the `robosuite` framework [63]. The built-in reward shaping is enabled, and the environment randomization parameters are set to their default values.

The complete training procedure is summarized in Algorithm 2.

Algorithm 2 State-Based Agents Training

```
Input: $D_{expert}$, $\theta_1$, $\theta_2$, $\phi$
 1: $\bar{\theta}_1 \leftarrow \theta_1$, $\bar{\theta}_2 \leftarrow \theta_2$                ▷ Initialize target networks
 2: Reroll $\mathcal{D}_{\text{expert}}$ to obtain $\mathcal{D}_{\text{demo}}$
 3: $\mathcal{D} \leftarrow \mathcal{D}_{\text{demo}}$                              ▷ Initialize buffer with demos
 4: Initialize $\lambda_{BC}$
 5: for iteration i=1,2,... do
 6:     for each environment step do
 7:         $a_t \sim \pi_\phi(a_t|s_t)$                          ▷ Sample action from policy
 8:         $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$                    ▷ Sample next state
 9:         $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$   ▷ Store transition
10:     end for
11:     for each gradient step do
12:         $\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta_i} J_Q(\theta_i)$ for $i \in \{1,2\}$   ▷ Update Q-functions (2.20)
13:         $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi J_{\pi,\text{BC}}(\phi)$          ▷ Update policy (4.3)
14:         $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha J(\alpha)$              ▷ Adjust temperature (2.25)
15:         $\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$ for $i \in \{1,2\}$   ▷ Update target networks (2.22)
16:         Anneal $\lambda_{\text{BC}}$
17:     end for
18: end for
Output: $\theta_1, \theta_2, \phi$                               ▷ Optimized parameters
```

4.2 Dataset Generation Procedure

The trained state-based agents are then used to generate the dataset of demonstrations for training the vision-based multi-embodiment policy. For each robot-gripper-task triplet, the corresponding trained agent is executed in the simulation environment, and the environment observations, including RGB images from multiple camera viewpoints, actions, and other relevant data, are recorded at each time step.

During dataset generation, to increase the diversity of the collected demonstrations, the default initial state randomization provided by `robosuite` was used, which randomizes the object initial position and orientation within a certain range, as well as the robot end-effector initial position and orientation.

Only successful episodes were stored in the dataset, limiting the maximum number of steps to reach the success condition, to ensure more efficient and consistent demonstrations. Additionally, to further improve the quality of the collected demonstrations, a pre-saving filtering strategy was employed, where only trajectories with a relatively small maximum joint velocity were saved, to avoid jerky and unnatural movements.

4.3 Vision-Based Multi-Embodiment Policy

The vision-based multi-embodiment policy is based on the Octo [2] architecture, which is finetuned to the tasks and robot-gripper embodiments considered in this work. Octo was selected due to the openly available pre-trained models, its modular design, which allows for adaptation to different sensor modalities and new action parameterizations, and its proven effectiveness when finetuned on downstream robotic manipulation tasks. As previously discussed in Chapter 3, leveraging pre-trained models for vision-based robotic policies has been shown to significantly improve performance and generalization capabilities, avoiding the need to train large models from scratch, which can be prohibitively expensive in terms of computational resources, time, and data availability.

The model is a ViT that processes a main RGB image observation, and optionally a wrist camera RGB image, and can accommodate additional information, such as proprioceptive data or other input modalities. The token prediction can be conditioned on a natural language instruction, preprocessed using the `t5` text encoder [64], or a goal image representing the desired final state. To predict the action, the model uses a diffusion-based decoder to produce multi-modal action distributions, given a readout token from the transformer backbone. The diffusion decoder allows for modeling complex action distributions, capturing multi-modality, while leaving more capacity to the transformer backbone to learn useful

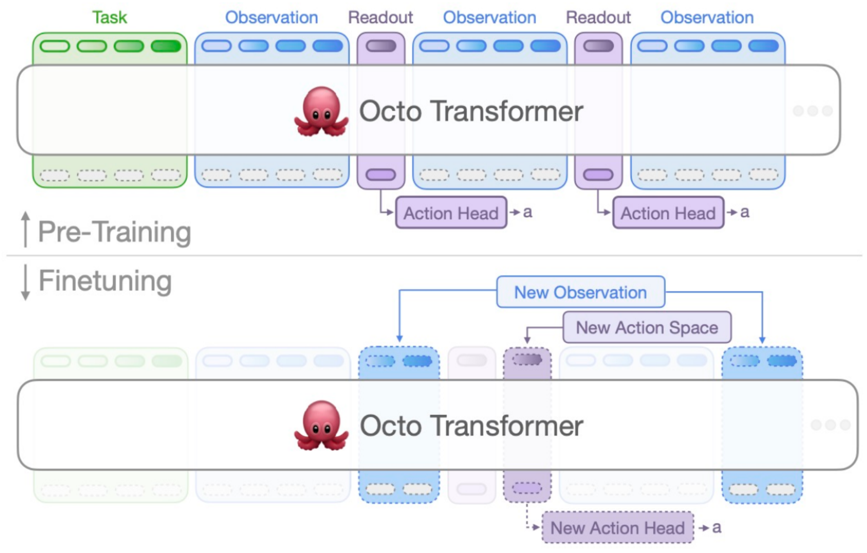


Figure 4.1: Overview of the Octo architecture. Source: arXiv preprint of [2] (arXiv:2405.12213).

representations from the input modalities. The model is pretrained using a selected subset of the large-scale Open X-Embodiment dataset [33], which is one of the largest available datasets for vision-based robotic manipulation tasks.

Architecturally, due to the flexibility of the ViT architecture, it exhibits a modular design where modality-specific encoders convert raw inputs into tokenized embeddings; a shared transformer backbone performs cross-attention over these tokens, maintaining the causal temporal structure; and a diffusion decoder maps the resulting latent context to actions. In particular, image observations are processed by ViT-style patch encoders into spatial tokens, textual instructions are encoded and broadcast across the temporal dimension, while, if present, additional information is embedded and appended as tokens.

To ensure causality, a masking strategy is employed in the attention layers. Each input token can only attend and be attended to by tokens from previous and current time steps, such that future information cannot be accessed. Additionally, the readout token, introduced in Section 2.1, is masked such that it cannot be attended to by any input token. In this way, the readout token can only aggregate information from the other tokens, effectively summarizing the entire input sequence up to the current time step, but it cannot influence the processing of the input tokens.

The following sections describe the key aspects of the finetuning procedure. In section 4.3.1, the overall finetuning configuration is presented. Section 4.3.2 details the input modalities used, including the novel gripper point cloud representation, while section 4.3.3 describes the contrastive losses introduced to improve embodiment generalization. Finally, section 4.3.4 explains the weighted BC loss employed during finetuning.

To have a reference of each of the components described in the following sections, Figure 4.2 provides an overview of the vision-based multi-embodiment policy architecture, including the input/output modalities, the model components, and the training losses.

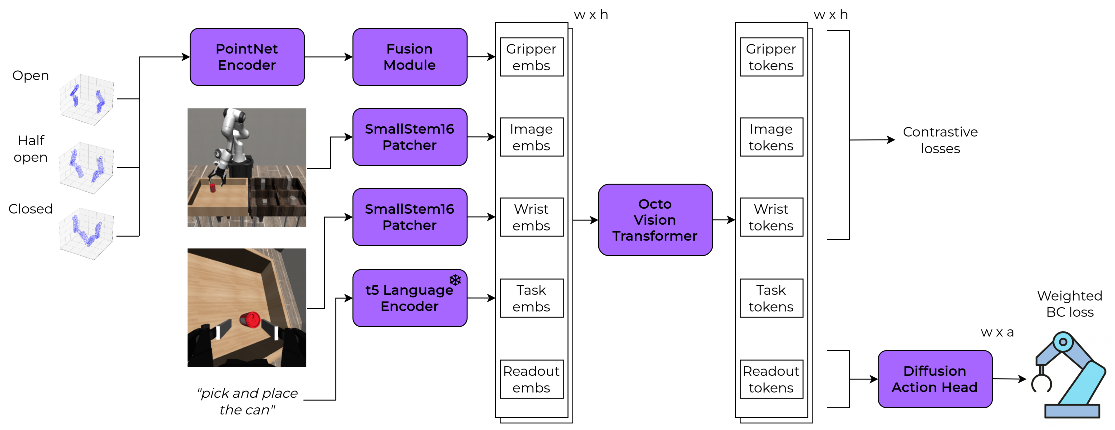


Figure 4.2: Overview of the vision-based multi-embodiment policy architecture, including input modalities, model components, and training losses. In the figure, “embs” stands for “embeddings”, while h, w, and a are the transformer hidden size, the window of input temporal observation size, and the action chunk size, respectively. The same figure is reported in horizontal format in the Appendix A.1 for better readability.

4.3.1 Fine-Tuning Configuration

To effectively finetune the pre-trained Octo model for the specific tasks and robot-gripper embodiments considered in this work, several key design choices presented in the following are made.

First of all, in accordance with the pretraining, to improve the performance of vision-based policies trained via IL, it is beneficial to provide a temporal input window of observations rather than a single frame, so that the model can better capture temporal dependencies in the input data.

Temporal dependencies are also crucial for action prediction. Indeed, predicting a sequence of future actions instead of a single one can lead to better results, as it allows the model to plan ahead and better model the task dynamics. This latter approach is referred to as action chunking [15], and it has been proven to be effective in improving the performance of vision-based policies. Furthermore, at inference time, to obtain an even smoother and coherent action sequence, the temporal ensembling technique [15] is employed, which consists of averaging the overlapping action predictions from multiple input windows shifted in time. In practice, the model, for each input window of observations, predicts a sequence of future actions, and the action that is actually executed at each time step is obtained by averaging the predictions from all the input windows that include that specific time step as part of their predicted action sequence.

The final action executed at time step t is thus computed as follows:

$$a_t = \frac{1}{N} \sum_{i=0}^{N-1} e^{-\alpha i} \, \hat{a}_t^{(i)} \tag{4.4}$$

$$\hat{a}_t^{(i)} = \pi\left(s_{t-i\,:\,t-i+w-1}\right)[t] \tag{4.5}$$

Where a_t is the final action executed at time step t, $\hat{a}_t^{(i)}$ is the action predicted for time step t from the input window starting at time step $t-i$, N is the number of overlapping input windows considered, w is the size of each input window, and α is a decay factor that gives more importance to predictions from input windows closer in time to the current time step.

Another important design choice is the task conditioning method. Even though the model supports goal conditioning, since it is not possible, or highly challenging [50], to have goal images for robot-gripper configurations not seen during training, only natural language instructions are used. This choice is motivated by the necessity of generalization, but it is worth noting that goal conditioning was found to be more effective in certain scenarios, as it provides a more direct representation of the desired outcome, from which the model can infer the necessary actions to achieve it.

Finally, the diffusion action head is reset with randomly initialized weights before finetuning, as the action space considered in this work differs from the one used during pre-training. The action space considered is the same as that introduced in Section 4.1.1.

4.3.2 Input Modalities

The main input modalities are the RGB image from a front-facing camera and from a wrist camera. The front-facing camera provides a broader view of the environment, allowing the model to understand the overall scene and the spatial relationships between objects. The wrist camera, on the other hand, offers a close-up view of the end-effector and the object being manipulated, which is crucial for understanding how to interact with it effectively. Similar to the proposed fine-tuning configuration, the images undergo a series of augmentations to improve the generalization capabilities of the model, including random brightness, contrast, and saturation adjustments. On the other hand, random cropping and rescaling are not applied, as they were found to be detrimental to performance in preliminary experiments.

Building on the considerations derived in Section 3.1.2, another input modality is introduced to the model: a point cloud representation of the gripper structure.

Gripper Point Cloud Representation

The driving idea behind the introduction of a gripper point cloud representation as an additional input modality is to provide the model with explicit information about the gripper's geometry and structure. This information can be crucial for manipulation tasks, as the gripper's shape and size can significantly influence how it interacts with objects in the environment and how the sequence of actions needed to solve the task. Additionally, through the employment of the attention mechanism in the transformer backbone, the model can learn to focus on the most relevant features of the gripper structure and how they relate in particular to the wrist camera view, which captures the interaction between the gripper and the object.

The point cloud representation is obtained from the 3D model of the gripper used in simulation. The 3D model is used to sample a fixed number of points uniformly distributed over the gripper surface till reaching a desired density, resulting in a point cloud that captures the overall shape and structure of the gripper. Finally, farthest point sampling is applied to downsample the point cloud to a fixed number of points, 2,048 in this work, to ensure a consistent input size for the model. This is without loss of generality, as in a real-world scenario, the point cloud could be acquired using depth sensors or 3D scanning techniques, or even be retrieved from a database of known gripper models.

To feed the point cloud representation to the model, a dedicated PointNet-based architecture [65] is employed, due to its good balance between effectiveness and computational efficiency, and due to the fact that more complex architectures would not provide significant benefits in this scenario. Indeed the idea is to provide the model with a compact representation of the gripper structure, rather than extracting highly detailed features from the point cloud, to be used in a similar way as the task description to condition the action prediction. Additionally, before feeding the point cloud to the PointNet encoder, some augmentations are applied, including random rotations, translations, jittering and pixel dropout, to avoid overfitting also due to the limited number of gripper types considered in this work.

In particular, an encoder-decoder PointNet model is pretrained with a reconstruction objective, then only the encoder part is retained to be used to extract a fixed-size embedding from the point cloud, at three different opening levels, i.e., fully closed, half-opened, and fully opened. The three embeddings are then concatenated and linearly projected to obtain the final gripper representation, which is appended as tokens to the transformer backbone, similarly to the proprioceptive information. This design choice is motivated by other works that have shown the benefits of providing multiple views of the gripper structure [24, 25].

As depicted in Figure 4.3, the gripper point cloud representation includes only the fingers of the gripper, as they are the parts that directly interact with the objects during manipulation tasks and mostly affect the shape and dimensions of the component, while the rest of the gripper structure is omitted to focus the model's attention on the most relevant features.

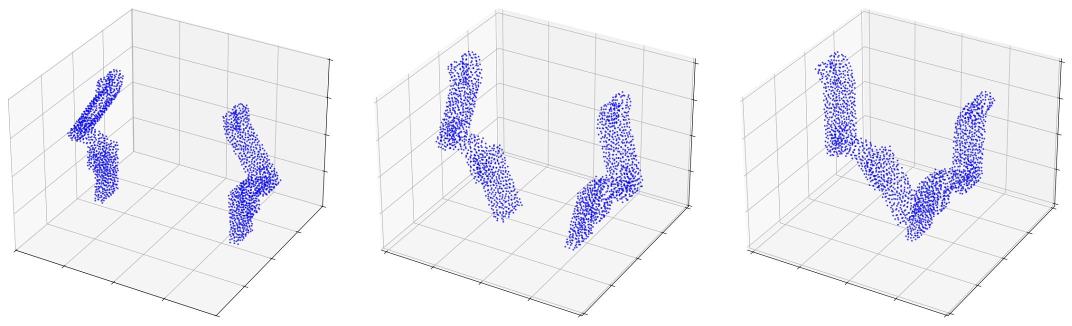


Figure 4.3: Examples of gripper point cloud representations (*Robotiq140Gripper*) used as input to the model in open (left), half-open (center), and closed (right) configurations.

4.3.3 Contrastive Losses for Embodiment Generalization

Another key addition to the training procedure is the introduction of contrastive losses to improve the embodiment generalization capabilities of the model. The core idea is to encourage the model to learn similar representations for different robot-gripper embodiments when they are performing the same task, and they are in similar states, as also proposed by Yang *et al.* [52]. This design choice addresses the vision gap problem, enabling the model to focus on the robot and object state, rather than the specific robot and gripper appearance.

As depicted in Figure 4.2, the contrastive losses are applied to the output tokens from the transformer backbone, specifically to the tokens corresponding to the main image, the wrist camera image, and the gripper point cloud representation. The task and the action readout tokens are instead excluded. The task token is excluded since only two tasks are considered in this work, so the model could easily learn to cluster the representations based on the task identity. On the other hand, the action readout token is excluded to avoid interfering with the action prediction objective, while leaving greater flexibility to the transformer backbone prediction. This choice will be discussed further in the next Chapter 5.

Main Image

To guide the attention mechanism towards learning embodiment-invariant features, the Soft Nearest Neighbors (SNN) loss was employed as contrastive loss [66, 67], due to its ability to consider multiple positive and negative samples in the same batch, providing a more robust learning signal, with respect to the commonly used InfoNCE loss [42]. The SNN for a batch (x, y) of size B, where x_i are the embeddings and y_i are the corresponding labels, is defined as:

$$\mathcal{L}_{\text{SNN}} = -\frac{1}{B} \sum_{i=1,\dots,B} \log \left(\frac{\sum_{j=1,\dots,B;\ j\neq i;\ y_i=y_j} \exp\left(-\frac{\|x_i - x_j\|^2}{\tau}\right)}{\sum_{k=1,\dots,B;\ k\neq i} \exp\left(-\frac{\|x_i - x_k\|^2}{\tau}\right)} \right) \tag{4.6}$$

Where τ is a temperature hyperparameter that controls the softness of the similarity distribution. As the InfoNCE loss, the SNN loss encourages the model to bring closer in the feature space the samples with the same label, while pushing away samples with different labels, considering all the samples in the batch as potential positives or negatives.

In this work, the labels y_i are a combination of a hard task label and a soft similarity score based on a distance metric. The hard task label indicates whether two samples belong to the same task, while the soft similarity score is computed to encourage the model to learn similar representations for states that are close in the task space, even if they are performed with different robot-gripper embodiments. This use of soft similarity targets is consistent with insights from soft contrastive learning literature, which demonstrates that softened supervision can improve representation quality [68].

The similarity score is computed as follows:

$$\mathcal{S}(s_i, s_j) = \exp\left(-\frac{d(s_i, s_j)}{\Delta_d}\right) \cdot \frac{\mathbb{I}[\,\text{task}(s_i) = \text{task}(s_j)\,]}{\psi(t_i, t_j)} \tag{4.7}$$

So the considered soft SNN loss becomes:

$$\mathcal{L}_{\text{SNN}} = -\frac{1}{B} \sum_{i=1,\dots,B} \log \left(\frac{\sum_{j=1,\dots,B;\ j\neq i} \mathcal{S}(s_i, s_j) \exp\left(-\frac{\|x_i - x_j\|^2}{\tau}\right)}{\sum_{k=1,\dots,B;\ k\neq i} \exp\left(-\frac{\|x_i - x_k\|^2}{\tau}\right)} \right) \tag{4.8}$$

In equation (4.7), $d(\cdot, \cdot)$ denotes a combined state distance, that, for the main image tokens, is defined as:

$$\begin{aligned} d_{\text{state}}(s_i, s_j) = {} & w_{p^r} \|p_i^r - p_j^r\|_2 + w_{q^r}\, 2 \arccos\left(|\langle q_i^r, q_j^r \rangle| \right) \\ & + w_{p^o} \|p_i^o - p_j^o\|_2 + w_{q^o}\, 2 \arccos\left(|\langle q_i^o, q_j^o \rangle| \right) \end{aligned} \tag{4.9}$$

The first two terms account for the robot end-effector position p^r and orientation q^r expressed as quaternions, while the last two terms account for the object position p^o and orientation q^o. The object terms, if included, allow the similarity to reflect not only robot pose but also relative object configuration. The weights $w_{p^r}, w_{q^r}, w_{p^o}, w_{q^o}$ are tunable hyperparameters that balance the contribution of each component to the overall distance metric. Overall, the distance metric captures the key aspects of the robot and object state that are relevant for the manipulation tasks considered, allowing the model to learn state-dependent similarities across different embodiments.

The scalar Δ_d is a tunable distance threshold that controls how rapidly similarity decays with state distance. The indicator $\mathbb{I}[\cdot]$ enforces that only samples from the same task are treated as potential positives, while the scalar function $\psi(t_i, t_j)$ implements a timestep penalty which down-weights pairs that are temporally far apart.

The timestep penalty used during training is defined as a simple piecewise function:

$$\psi(t_i, t_j) = \begin{cases} 1, & |t_i - t_j| \leq \Delta_t, \\ \dfrac{|t_i - t_j|}{\Delta_t}, & |t_i - t_j| > \Delta_t, \end{cases} \tag{4.10}$$

Where $t_i, t_j \in [0{,}1]$ are normalized timestep percentages and Δ_t is an hyperparameter. This choice penalizes pairs that are separated by more than the allowed temporal window, effectively reducing their weight in the soft nearest-neighbor objective, which is particularly useful for tasks in which some states can be reached at multiple points in time, e.g., in a lifting task, the robot state before and after the grasping execution is similar. The normalization ensures that the penalty is consistent across trajectories of different lengths.

Wrist Image

For wrist views, the same similarity formulation introduced in equation (4.7) is used, but building on the assumption that object-gripper relative geometry is more informative than absolute poses, the distance metric is computed differently. The motivation is that the wrist camera is mounted on the robot's end-effector, so its viewpoint is heavily influenced by the relative position between the gripper and the object being manipulated, and by object position. To capture those factors, the distance metric is based on a distance that emphasizes the relative configuration between gripper and object.

A relative vector $v = p^r - p^o$ is defined, representing the vector from the gripper to the object. For a pair of samples i, j, the signed per-axis differences are computed, i.e., $\delta_{v,a} = v_i^a - v_j^a$ for $a \in \{x, y, z\}$, and summed up in a sign-sensitive manner to obtain a base distance. To further enhance the sensitivity to large per-axis differences, a threshold-based penalty term is added. This term penalizes differences that exceed a certain threshold Δ_v by adding a normalized excess distance to the overall metric, that grows linearly with the amount by which the difference exceeds the threshold. Finally, the robot orientation and object position differences are also included to obtain the final distance metric, since they significantly affect the wrist camera view. The resulting distance is metric:

$$\begin{aligned} d_{\text{obj-grip}}(s_i, s_j) = \sum_{a\in\{x,y,z\}} |v_i^a - v_j^a| + \frac{\sum_{a\in\{x,y,z\}} \max\left(|v_i^a - v_j^a| - \Delta_v, 0\right)}{3\Delta_v} + \\ + w_{q^r}\, 2\arccos\left(|\langle q_i^r, q_j^r\rangle|\right) + w_{p^o}\, \|p_i^o - p_j^o\|_2 \end{aligned} \tag{4.11}$$

Where Δ_v is a tunable per-axis distance threshold that controls how rapidly similarity decays with object-gripper relative distance, and w_{q^r}, w_{p^o} are weights for the robot orientation and object position terms, respectively.

Gripper Point Cloud

The processed gripper point-cloud representation is used to compute a contrastive loss to further enhance embodiment-invariant features and to maintain consistent gripper representations across different tasks. In this case, the SNN loss is employed with hard gripper-type labels to define positive and negative samples, as the gripper representation should be independent of the specific task and state.

Annealing Strategy

To effectively integrate the contrastive losses into the training process without affecting convergence or badly impacting task performance, an annealing strategy is employed. The contrastive loss terms are assigned an exponential weight decay factor that decreases their influence over time. This approach allows the model to initially focus on learning embodiment-invariant features, while gradually shifting its attention towards optimizing task performance as training progresses.

The overall constrastive loss function used during training is thus defined as:

$$\mathcal{L}_{\text{contrastive}} = \lambda_{\text{decay}}^{\text{exp}}(t)\left[\lambda_{\text{img}}\mathcal{L}_{\text{SNN,img}} + \lambda_{\text{wrist}}\mathcal{L}_{\text{SNN,wrist}} + \lambda_{\text{pc}}\mathcal{L}_{\text{SNN,pc}}\right] \tag{4.12}$$

Where $\mathcal{L}_{\text{SNN,img}}$, $\mathcal{L}_{\text{SNN,wrist}}$, and $\mathcal{L}_{\text{SNN,pc}}$ are the SNN losses computed on the main image embeddings, wrist image embeddings, and gripper point cloud embeddings, respectively. The scalar weights λ_{img}, λ_{wrist}, and λ_{pc} control the relative importance of each contrastive loss term, while $\lambda_{\text{decay}}^{\text{exp}}(t)$ is the exponential decay factor that reduces the influence of the contrastive losses over time.

4.3.4 Weighted Behavioral Cloning

To further improve the learning process, a weighted behavioral cloning loss is employed, where each sample in the dataset is assigned a weight based on its quality. The quality of each sample is estimated using information already obtained during the state-based agent training process. In this way, there is no need to tune and train additional modules or manually label the dataset samples, but the model can still focus more on high-quality samples, utilizing knowledge already available.

For each demonstration sample, the estimated advantage value $\hat{A}(s, a)$, computed as in Equation (4.1), is extracted from the corresponding state-based agent. The advantage value provides an estimate of how much better the action from the demonstration is compared to the average action proposed, following the intuition that samples with a high advantage value are more likely to be beneficial for the learning process. Indeed, the approach is inspired by AWR [53], which uses a similar weighting strategy to prioritize high-quality samples, but that requires training an additional value function to estimate the advantage values, which can be tricky for a dataset with demonstrations coming from multiple embodiments.

To compute the weight for each sample, the advantage values are normalized to zero mean and unit variance across each task-robot-gripper triplet, and then across the entire dataset, to ensure that the weights are comparable across different tasks and embodiments. The per-sample weight is then computed as follows:

$$w(s, a) = 1 + \gamma_{\text{WBC}} \cdot \tanh \hat{A} \tag{4.13}$$

Where γ_{WBC} is a scaling factor that controls the overall range of the weight, and the tanh application helps to limit the influence of outliers in the input distribution, improving the stability of the weighting scheme. The final weights are mean-normalized to 1.0 across samples in each training batch to ensure consistent gradient magnitudes, leading to an overall training loss defined as:

$$\mathcal{L}_{\text{WBC}}(\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\tilde{w}(s, a) \left\| a - \pi_\theta(s) \right\|_2^2 \right] \tag{4.14}$$

Where $\tilde{w}(s, a)$ are the batch mean-normalized weights.

The overall training loss used to finetune the vision-based multi-embodiment policy is thus defined as the sum of the weighted behavioral cloning loss and the contrastive losses:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{WBC}} + \mathcal{L}_{\text{contrastive}} \tag{4.15}$$

Chapter 5

Experiments

In this chapter, the experimental setup and results obtained to evaluate the proposed methods are presented.

Section 5.1 describes the simulation environment, tasks, and robot-gripper configurations considered. Section 5.2 details the training procedure and results of the state-based RL agents used to generate the multi-embodiment dataset. Section 4.2 outlines the dataset generation process. Finally, Section 5.4 describes the finetuning procedure of the vision-based multi-embodiment policy, including the architecture, training details, and extensive evaluation of its performance and generalization capabilities, including ablation studies on the proposed methodology.

5.1 Experimental Setup

The considered simulation environment is `robosuite` [63], which provides a variety of robotic arms, grippers, and pre-defined manipulation tasks. It is built on top of MuJoCo [69] physics engine, and it also provides controllers and IK solvers for the included robots.

The tasks considered are the *Lift* and *PickPlaceCan*, where the objective is to lift a cube from the table and to pick a can and place it in a target bin, respectively. The first task is considered due to its simplicity, allowing for focus on the embodiment generalization capabilities. In contrast, the second task is more complex, requiring a more challenging manipulation sequence, including grasping an object with a round shape from a more variable initial position, and placing it in a specific target location, overcoming two small barriers.

The robots considered to train the vision-based agent are the *Panda*, *IIWA*, and *Jaco* arms, while the grippers are the *PandaGripper*, *RethinkGripper*, *Robotiq140Gripper*, and *RobotiqThreeFingerGripper*. This selection provides a good variety of robot and gripper types, with different kinematics, sizes, and appearances.

It is important to notice that all the robot-gripper combinations are considered for each task, resulting in a total of 24 different robot-gripper-task triplets.

At test time, two additional robots, the *Kinova3* and the *UR5e*, and two additional grippers, the *Robotiq85Gripper* and the *JacoThreeFingerGripper*, are included to evaluate the zero-shot generalization capabilities of the vision-based multi-embodiment policy.

From now on the *Gripper* suffix is omitted when referring to grippers for brevity, except when necessary for clarity.

The two tasks, robot arms, and grippers considered are displayed in Figures 5.1, 5.2, 5.3, and 5.4, respectively. Additionally, in the Appendix, Figures A.2 and A.3 report all the state-based agents solving the *Lift* and *PickPlaceCan*, respectively, while Figure A.4 displays all the 30 considered robot-gripper configurations.

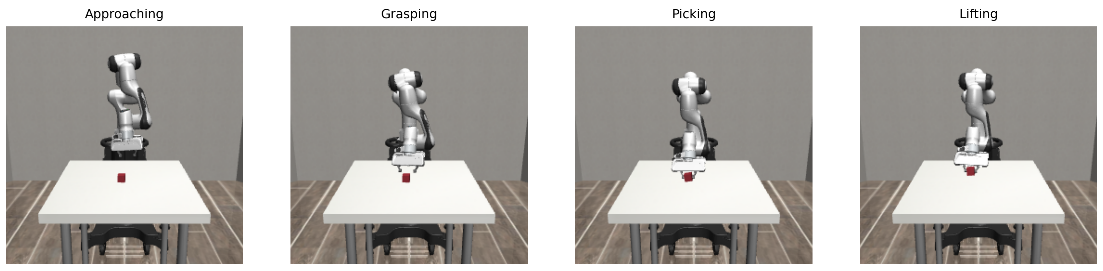


Figure 5.1: *Lift* task progress in `robosuite` with *Panda* robot and *Panda* gripper.

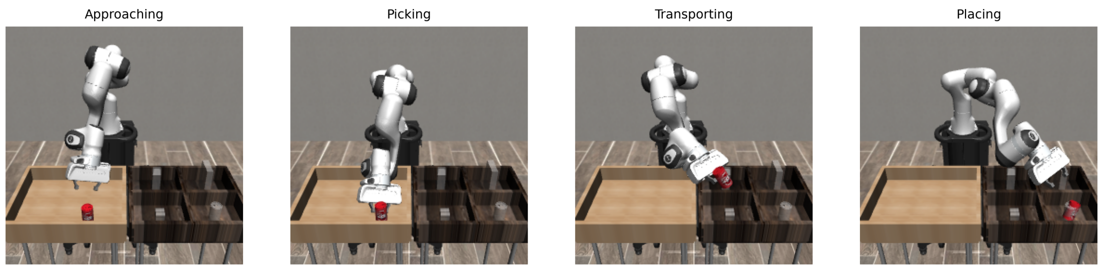
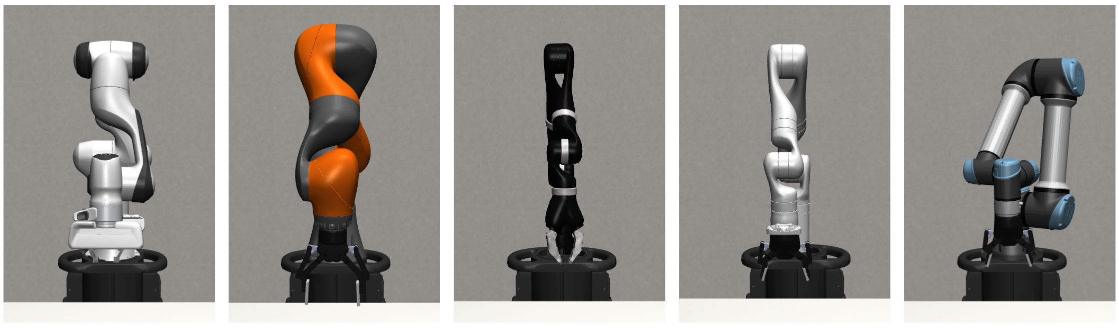


Figure 5.2: *PickPlaceCan* task progress in `robosuite` with *Panda* robot and *Panda* gripper.

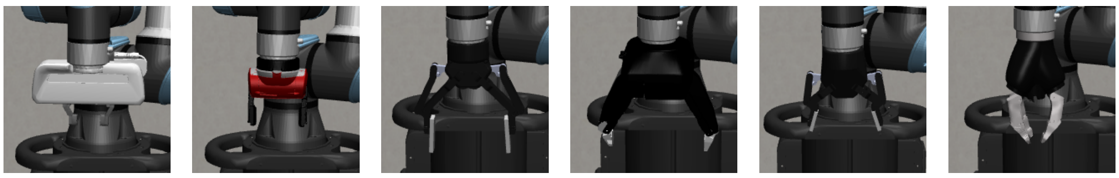
5.2 State-Based Agent Training

The demonstrations used to train the state-based agents are obtained from the publicly available dataset provided by the `robomimic` project [70]. The trajectories considered are the proficient human teleoperated ones, collected using the *Panda* robot with the *PandaGripper*, for both tasks. The dataset contains 200 demonstrations per task.



(a) *Panda* **(b)** *IIWA* **(c)** *Jaco* **(d)** *Kinova3* **(e)** *UR5e*

Figure 5.3: Robot arms used in the experiments: (a) *Panda*, (b) *IIWA*, (c) *Jaco* for training, (d) *Kinova3*, (e) *UR5e* for testing.



(a) *Panda* **(b)** *Rethink* **(c)** *Robotiq-140* **(d)** *Robotiq-ThreeFinger* **(e)** *Robotiq85* **(f)** *Jaco-ThreeFinger*

Figure 5.4: Grippers used in the experiments: (a) *Panda*, (b) *Rethink*, (c) *Robotiq140*, (d) *RobotiqThreeFinger* for training, (e) *Robotiq85*, (f) *JacoThreeFinger* for testing.

The state-based agents are trained following the procedure described in section 4.1. The actor and critic networks are both 3-layer MLP with 256 hidden units per layer and ReLU activations, layer normalization after each hidden layer, and a dropout rate of 0.01. The actor network outputs the mean and standard deviation of a Gaussian distribution, for a total of 14 output dimensions. The actions are then sampled from this distribution and then squashed using a tanh function to obtain predicted actions in the [-1, 1] range. The critic networks output a single Q-value for each state-action pair.

Following the state and action space alignment procedure, the state space is defined as the concatenation of the robot end-effector position and orientation quaternion, the object position and orientation quaternion, the gripper state, and other task-specific state information, such as the object-gripper distance. On the other hand, the action space is defined as the end-effector position delta, and orientation delta expressed as angular displacement, and the gripper opening percentage. The state and action space dimensions are summarized in Tables 5.1 and 5.2.

Table 5.1: State-Based Agents State Space

Component	Dimension
End-effector position	3
End-effector orientation	4
Object position	3
Object orientation	4
Gripper state	2 to 11
Task-specific info	3
Total	19 to 28

Table 5.2: State-Based Agents Action Space

Component	Dimension
End-effector position delta	3
End-effector orientation delta	3
Gripper opening percentage	1
Total	7

The hyperparameters used for training the state-based agents are summarized in Table 5.3. The optimizer employed is AdamW [71], with the learning rates specified in the table, and standard parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay $= 0.01$).

The robots are controlled at a frequency of 20 Hz, in accordance with the demonstration dataset, and the horizon for each episode is set to 100 steps for the *Lift* task and 200 steps for the *PickPlaceCan* task. Longer horizons were considered in preliminary experiments, but they did not provide significant benefits in terms of performance.

5.2.1 Training Results vs Baseline SAC

To test the effectiveness of the proposed approach, a set of baseline SAC agents is trained on the same configurations, but without leveraging any demonstrations. The baseline agents share the same architecture and hyperparameters as the proposed method, except for the target entropy and the replay buffer size. Indeed, to encourage more exploration, aligning with common practices in RL, the target entropy is set to $-\dim(\mathcal{A}) = -7.0$, while the replay buffer size is increased to 1,000,000 transitions. Furthermore, 20,000 initial random environment steps are performed before starting the training process, to bootstrap the exploration.

The results summarized in Table 5.4 report the average success rates and average returns over every robot-gripper configuration for each task, for both the proposed method and the baseline SAC agents. The success rates are computed during the

Table 5.3: State-Based Agents Training Hyperparameters

Hyperparameter	**Value**
Number of training steps	1,000,000
Replay buffer size	250,000
Batch size	512
Discount factor γ	0.99
Actor learning rate λ_π	3×10^{-5}
Critic learning rate λ_Q	1×10^{-4}
Temperature learning rate λ_α	1×10^{-4}
Target network update rate τ	0.005
Actor update frequency	1
Critic update frequency	2
Gradient clipping	5.0
Initial temperature α	0.5
Target entropy	$-\dim(\mathcal{A})/2 = -3.5$
BC loss weight λ_{BC}	0.1
BC loss annealing steps	100,000
AWAC advantage temperature β	1.0

evaluation phase of training, at different checkpoints, computing the average over just 10 evaluation episodes per checkpoint due to computational constraints. This comparison is not meant to be exhaustive, but rather explicative of the benefits of the proposed approach.

For the *Lift* task, since the proposed method achieves very high success rates quickly, the results for it are reported at earlier training steps (100k, 250k, and 500k), with respect to the baseline SAC agents (100k, 500k, and 1M) to provide a more fair comparison. Interestingly, at step 100,000, after the BC term is completely annealed, the proposed agents already achieve reasonable success rates for the *Lift* task, showing that the adapted demonstrations are effective in guiding the learning process. On the other hand, the baseline SAC agents struggle to make any progress at the beginning of training; indeed, at step 100,000 (removing the initial random exploration phase), they still achieve a 0% success rate. Also, at later training steps, the baseline agents are not able to match the performance of the proposed method, reaching lower success rates even after performing a significant amount of additional training steps.

Regarding the more challenging *PickPlaceCan* task, after the initial phase where the BC term is active, the proposed method fails to achieve a significant success rate. However, as training progresses, the early guidance provided by the demonstrations allows the agents to explore more efficiently, and a steady improvement in performance is observed, reaching a success rate of 65.8% after 1,000,000 training steps.

Table 5.4: State-Based Agents Performance Training Results

Task	**Metric**	**Train Step**	**Proposed Method**	**Baseline SAC**
Lift	Success Rate (%)	100k	45.8	0.0
		250k / 500k	89.2	45.8
		500k / 1M	96.7	70.0
	Average Return	100k	36.0	7.89
		250k / 500k	56.03	40.04
		500k / 1M	60.32	51.26
PickPlaceCan	Success Rate (%)	100k	10.0	0.0
		500k	52.5	0.0
		1M	65.8	0.1
	Average Return	100k	23.77	0.78
		500k	75.16	15.03
		1M	115.64	59.56

On the other hand, the baseline SAC agents struggle to achieve any success on this task regardless of the training time. Even if there is an increase in the average return, the agents are not able to discover successful strategies, and the returns tend to plateau after a certain amount of training, signaling that the exploration is not effective enough to solve the task. Notably, the success rate of the adapted demonstrations alone is quite low (for some configurations, it is as little as 2-3%), but the proposed method is still able to leverage them to bootstrap the learning process effectively.

These observations can be further confirmed by looking at the average episode returns achieved during training by some sample configurations, which are displayed in Figures 5.5 and 5.6 for the *Lift* and *PickPlaceCan* tasks, respectively. The plots show the average return over 10 evaluation episodes every 5,000 training steps, for 3 different robot-gripper configurations, selected to represent a variety of embodiments, for both methods. In the *Lift*, the proposed method achieves rapidly high returns to then stabilize, while the baseline SAC agents show a slower improvement over time. In the *PickPlaceCan*, both methods show a way higher variance due to the complexity of the task, but the proposed method is able to steadily improve over time, while the baseline SAC agents struggle to achieve significant returns, especially in the early stages of training. The curves suggest that it is possible that longer training times could potentially allow the baseline SAC agents to achieve better performance, even if the returns seem to plateau after a certain amount of training.

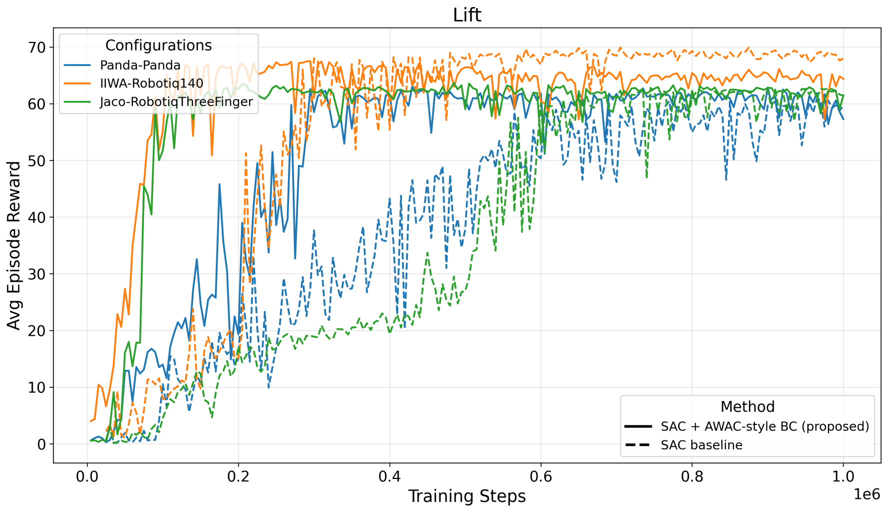


Figure 5.5: State-based agents average return during training for the *Lift* task.

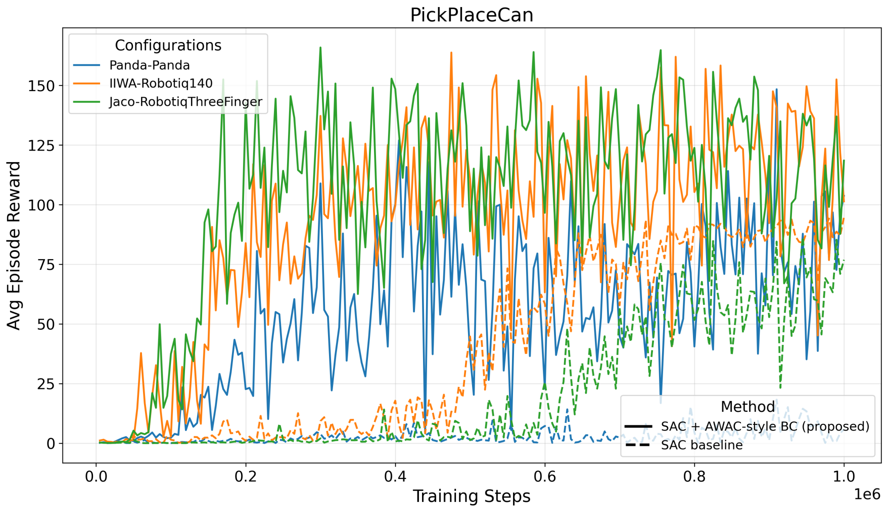


Figure 5.6: State-based agents average return during training for the *PickPlaceCan* task.

It is worth mentioning that to obtain reasonable trajectories for the *Lift* task, a more complex success constraint was used during training, requiring the object to be lifted above a threshold three times higher than the standard one. This choice was made to mitigate reward hacking issues observed during preliminary experiments, where SAC agents learned to tilt or hit the object to trigger the success condition without actually lifting it. On the other hand, the proposed approach was able to overcome this issue, even with the standard success constraint, thanks to the guidance provided by the adapted demonstrations.

5.2.2 Performance Evaluations

The overall success rates achieved by the state-based agents after completing the full training procedure (1M steps) are summarized in Tables 5.5 and 5.6, for the *Lift* and *PickPlaceCan* tasks, respectively. Those results are obtained by running 100 evaluation episodes per robot-gripper-task configuration, for a total of 2,400 episodes.

Table 5.5: *Lift* Task State-Based Agents Success Rates (%)

Lift	*Panda*	*IIWA*	*Jaco*	**Aggregated**
PandaGripper	99	100	98	99.0
Rethink	100	100	99	99.3
Robotiq140	100	99	100	99.7
RobotiqThreeFinger	100	100	100	100
Aggregated	99.8	99.8	99.3	99.6

Table 5.6: *PickPlaceCan* Task State-Based Agents Success Rates (%)

PickPlaceCan	*Panda*	*IIWA*	*Jaco*	**Aggregated**
PandaGripper	55	25	51	43.7
Rethink	60	37	45	47.3
Robotiq140	81	74	95	83.3
RobotiqThreeFinger	75	60	69	68.0
Aggregated	67.8	49.0	65.0	60.6

The results show that for the *Lift* task, the state-based agents achieve very high success rates across all robot-gripper configurations with an overall average success rate of 99.6%. On the other hand, for the more challenging *PickPlaceCan* task, the performance is more variable, with some configurations achieving high success rates (up to 95%), while others struggle more (as low as 25%), even if overall the average remains satisfactory (60.6%). This variability can be attributed to differences in gripper capabilities and kinematics of the different robot arms, which affect both the transferability of the adapted demonstrations, but also the learning process itself. Indeed, some specific configurations may have more limited reachability or dexterity, making it more difficult to perform the required manipulation actions effectively.

It is interesting to note that in the *PickPlaceCan* task, the *Robotiq140* and *RobotiqThreeFinger* grippers mounted on any robot outperform the *Panda-PandaGripper* configuration, which was the one for which expert demonstrations were collected.

This suggests, as will be observable in the next sections, that *Robotiq* grippers provide an easier and more reliable way to grasp the can, thanks to their wider fingers. Nevertheless, this result also highlights the effectiveness of the proposed knowledge transfer method, which allows the agent to leverage demonstrations collected with a specific embodiment to successfully train agents for different embodiments, going beyond the limitations of the original demonstration data.

5.3 Dataset Generation

The 24 trained state-based agents were each used to generate 100 successful episodes, resulting in a total of 2,400 demonstrations. The maximum number of steps allowed to achieve the success condition was limited to 40 for the *Lift* task and 80 for the *PickPlaceCan* task, in such a way as to obtain more efficient and consistent demonstrations.

5.4 Vision-Based Policy Training

The model used for finetuning is based on the Octo-Small architecture [2], which has approximately 27M parameters, which mirrors the ViT-S configuration from Dosovitskiy *et al.* [3]. The choice of the 27M parameter model, instead of the larger Octo-Base with 93M parameters, is motivated by the need for a more compact and memory-efficient architecture, which is easier to finetune with the available computational resources. Furthermore, the implementation provided by Ghosh *et al.* [2] is in JAX, while the training pipeline used is based on PyTorch, so a conversion of the model was necessary, starting from an existing `octo-pytorch`[1] implementation.

Octo-Small is a 12-layer transformer with 6 attention heads, a model dimension of 384 (h in Figure 4.2), and a feedforward dimension of 1536. The main image input resolution is set to 256x256 pixels, while the wrist image resolution is set to 128x128 pixels. The images are processed by a convolutional stem followed by a patch embedding layer, resulting in 16x16 patches. For the language instruction decoding, a pretrained and frozen `t5-base` transformer model [64] is used (111M). A 3-layer MLP with 256 hidden units per layer, ReLU activations, layer normalization, and residual connections is used as diffusion action head, with the standard DDPM objective [5] and a cosine noise schedule [72], with 20 diffusion steps.

The PointNet encoder processes the three point cloud inputs (open, half-open, and closed gripper configurations) using 1D convolutional layers with hidden

[1]https://github.com/emb-ai/octo-pytorch

dimensions [64, 128, 256], followed by batch normalization, ReLU activation, and 0.1 dropout, reducing each point cloud to a 256-dimensional feature via global max pooling. The three point cloud features are then fused together and expanded to 16 tokens of dimension 384 (matching the model dimension h) before being fed to the transformer alongside other input modalities. Those dimensions were selected taking inspiration from standard architectures, while maintaining a compact size to limit the overall model complexity, as this component is used to provide additional context about the embodiment rather than to learn detailed point cloud representations. Nevertheless, after undergoing the pretraining procedure with a reconstruction decoder, the PointNet showed good capabilities in reconstructing the input point clouds, validating its effectiveness in capturing relevant features.

Regarding the temporal configuration, the input window length is set to 2 time steps, which are used to predict a chunk of 4 future actions, respectively w and a in Figure 4.2. At inference time, the exponential decay factor α for temporal ensembling, defined in Equation 4.4, is set to 0.5.

The number of tokens per input type is reported in Table 5.7. The term “broadcasted” indicates that the input is replicated across the temporal dimension for each time step in the input window. Notice that the number of input and output tokens per type is the same, as the ViT architecture maintains the token count throughout the layers.

Table 5.7: Vision-Based Policy Tokens per Input Type

Input Type	**Tokens**
Main image	16 x 16 = 256
Wrist image	8 x 8 = 64
Language instruction	16 (broadcasted)
Gripper point cloud	16 (broadcasted)
Readout action	1

The hyperparameters used for finetuning the vision-based multi-embodiment policy are summarized in Table 5.8. The optimizer employed is AdamW [71], with standard parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay = 0.01), with a cosine annealing learning rate schedule with linear warm-up.

5.4.1 Representation Learning Settings and Results

As described in Section 4.3.3, contrastive learning losses are employed during finetuning to encourage the model to learn embodiment-invariant features. Those losses are based on similarity metrics defined over different input modalities, such as images and point clouds. One of the main challenges when employing such paradigms is the selection of appropriate hyperparameters for the similarity metrics

Table 5.8: Vision-Based Policy Finetuning Hyperparameters

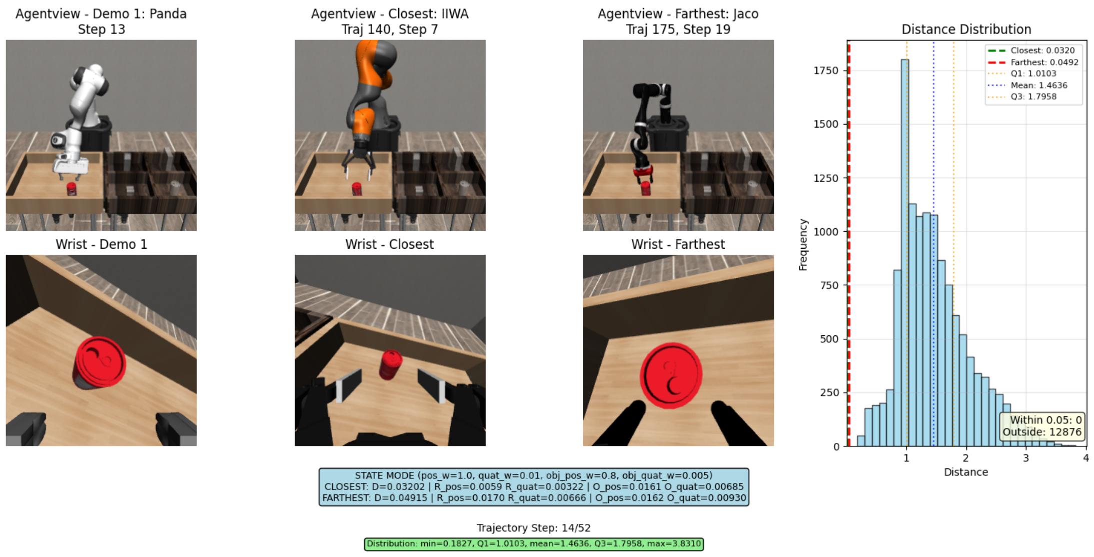
Hyperparameter	**Value**
Number of finetuning steps	50,000
Batch size	256
Peak learning rate	3×10^{-4}
Warmup steps	2,000
Gradient clipping	1.0
Contrastive loss annealing steps	30,000
Weighted BC amplitude γ_{WBC}	0.1
Window size w	2
Action chunk size a	4

used to define positive and negative samples.

Due to the high number of hyperparameters involved, a systematic tuning procedure was not feasible. Instead, the values were selected based on statistical and empirical considerations. The temperature values and the contrastive loss weights were chosen based on common practices in contrastive learning literature, to properly handle the different token types and to balance their contributions to the overall loss. The distance and timestep thresholds, and weights for the distance metrics were selected based on the relative importance and magnitude of each component influencing the similarity, ensuring that the model focuses on the most relevant aspects of the state when learning embodiment-invariant features. Regarding this step, some statistical and empirical analyses were performed on the training dataset to determine reasonable ranges for the distance metrics, considering the variability in robot and object positions and orientations across different demonstrations.

In Figure 5.7, an example of the comparison performed to select the thresholds is depicted. In particular, a script was developed for visualizing, for each timestep in a demonstration, the closest and farthest within-threshold samples in the dataset based on the defined distance metrics, to ensure that the selected thresholds provided a meaningful distinction between similar and dissimilar states. Additionally, the percentage of contributions of each component to the overall distance metric was analyzed to verify that no single component dominated the metric excessively. Even if this analysis only allowed for a qualitative assessment, it provided useful insights into the behavior of the distance metrics, also through the histogram visualization, helping to narrow down the hyperparameter selection process.

Nevertheless, after finding a reasonable range of values for the thresholds, different combinations were tested by training the model and evaluating the quality



```
STATE MODE (pos_w=1.0, quat_w=0.01, obj_pos_w=0.8, obj_quat_w=0.005)
CLOSEST: D=0.03202 | R_pos=0.0059 R_quat=0.00322 | O_pos=0.0161 O_quat=0.00685
FARTHEST: D=0.04915 | R_pos=0.0170 R_quat=0.00666 | O_pos=0.0162 O_quat=0.00930
```

```
Distribution: min=0.1827, Q1=1.0103, mean=1.4636, Q3=1.7958, max=3.8310
```

Figure 5.7: Empirical Analysis for Contrastive Loss Hyperparameter Selection of the learned representations and the task performance.

The final hyperparameter values used for the contrastive losses during finetuning are summarized in Table 5.9.

Table 5.9: Vision-Based Policy Contrastive Losses Hyperparameters

Hyperparameter	**Value**
Temperature for images τ_{img}	0.2
Temperature for point cloud τ_{pc}	0.07
Timestep percentage threshold Δ_t	0.15
Main image weights: $w_{p^r}, w_{q^r}, w_{p^o}, w_{q^o}$	$1.0,\ 1 \times 10^{-2},\ 0.8,\ 5 \times 10^{-3}$
Wrist image L1 threshold Δ_v	0.05
Wrist image weights: w_{q^r}, w_{p^o}	$1 \times 10^{-2},\ 0.2$
Main image distance threshold $\Delta_{d,m}$	0.05
Wrist image distance threshold $\Delta_{d,w}$	0.03
Contrastive loss weights: $\lambda_{\text{img}}, \lambda_{\text{wrist}}, \lambda_{\text{pc}}$	0.4, 0.4, 0.2

It is worth mentioning that the contrastive learning losses rapidly converged during training, with most of the improvement occurring within the first 2,000 finetuning steps. This behavior is likely due to the fact that the model already possesses a strong prior from the pretraining phase, and the contrastive losses serve to fine-tune the representation space to better capture embodiment-invariant features.

Results

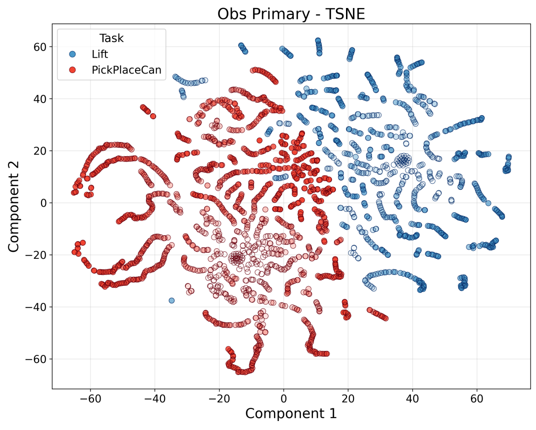
To evaluate the effectiveness of the contrastive losses in structuring the learned representation space, a series of analyses were performed on the embeddings extracted from the transformer backbone after finetuning. In particular, t-Distributed Stochastic Neighbor Embedding (t-SNE) projection visualizations were created to observe how the embeddings cluster, and accuracy and homogeneity scores were computed to quantitatively assess the separation of different robot and gripper types in the embedding space given k-Nearest Neighbors (k-NN) groupings. t-SNE is a dimensionality reduction technique that is particularly well-suited for visualizing high-dimensional data, as it preserves local structures and relationships between data points, while k-NN clustering is a clustering algorithm that groups data points based on their proximity in the feature space.

The visualization and the metrics' computations are performed considering one random trajectory per configuration not used during training, for a total of 24 trajectories, to have a comprehensive overview of the learned representations. The output tokens, corresponding to the main image, wrist image, gripper point cloud, and action readout input embeddings, are obtained by forwarding the observations through the transformer backbone.

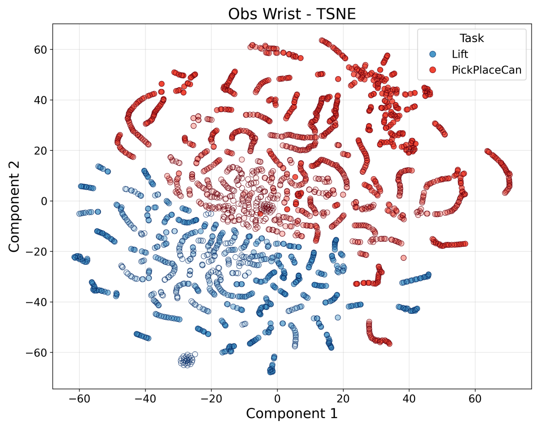
It is important to mention that the results reported are obtained using a model that showed the best representation learning results, and task performance comparable to the best model reported in Section 5.4.2. This choice was made to provide a clearer and more interpretable analysis of the learned representations, as the best-performing model in task resolution showed slightly more noisy results. Furthermore, as discussed in Section 5.4.3, representation learning quality does not always correlate directly with task performance, but the effectiveness of the learned representations in capturing embodiment-invariant features is still a valuable aspect to analyze, which may be leveraged in future works.

In Figure 5.8, the t-SNE projections of the embeddings for the main image, wrist image, gripper point cloud, and action readout are shown. Each point is colored based on the hard part of the labels for the specific embedding considered, i.e., the task for all the embeddings except for the gripper point cloud embeddings, which are colored based on the gripper type. Additionally, a color fading is proposed (except for the gripper point cloud embeddings) based on the timestep percentage within the episode, to visualize how the embeddings evolve. This visualization helps to assess whether the learned embeddings capture the task's temporal structure, since it is reasonable to assume that similar timesteps show similar states, and the state-based contrastive losses explicitly take into account temporal proximity when defining positive and negative samples.

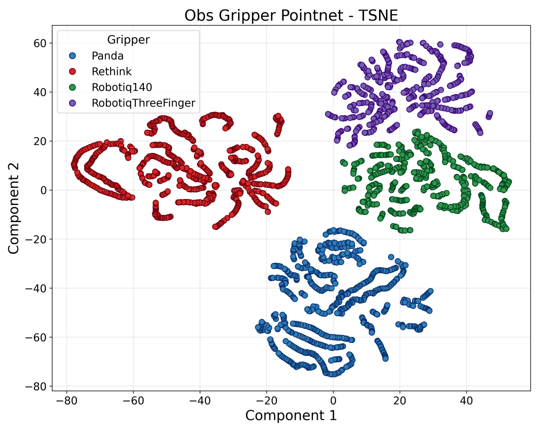
Regarding the main image embeddings in Figure 5.8a, it is possible to observe that the two tasks are separated, and the time evolution has been modeled effectively.



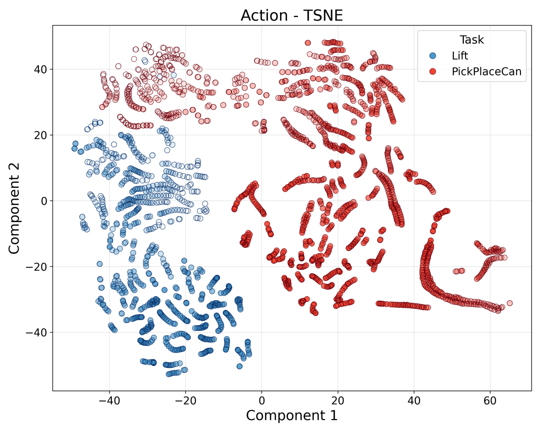
(a) Main Image Embeddings



(b) Wrist Image Embeddings



(c) Gripper Point Cloud Embeddings



(d) Action Readout Embeddings

Figure 5.8: t-SNE Projections of Embeddings with Contrastive Losses.

Indeed, for both tasks, apart from some outliers, the embeddings belonging to the beginning of the episodes are clustered together, and as time progresses, the embeddings move away from that initial cluster, towards different regions of the embedding space. This behavior also indicates that later timesteps have more variability in the states, as the embeddings are more spread out.

A similar behavior is observed for the wrist image embeddings in Figure 5.8b, where the two tasks are well separated, but the temporal structure is less clear, especially for the *Lift* task. This is likely due to the fact that the wrist camera provides several similar views while performing the *Lift* task, leading to less distinctive embeddings.

For the gripper point cloud embeddings in Figure 5.8c, the separation between the different gripper types is very clear, with respect to the task-based ones in the previous plots. Indeed, this contrastive loss used only hard labels based on the gripper type, producing well-defined different clusters for each gripper.

Finally, for the action readout embeddings in Figure 5.8d, a similar behavior to the image embeddings is observed, even if in this case the task separation and the temporal structure are way more pronounced. Additionally, the early timesteps for both tasks are clustered closely, indicating that the initial actions taken are quite similar across different configurations. This results is remarkable even considering that no explicit contrastive loss was applied to the action readout embeddings, showing that the model has learned to structure the representation space effectively.

The quantitative results, summarized in Table 5.10 and Table 5.11, confirm the observations from the t-SNE visualizations. The computation of the scores was performed using k-NN clustering with k values of 3, 5, and 10. Basically, for each embedding, the k nearest neighbors are found, and the majority class among those neighbors is used to predict the label for the sample. The accuracy and homogeneity scores are then computed based on those predicted labels. The accuracy score is the ratio of correctly classified samples to the total number of samples, while the homogeneity score measures how much each cluster contains only members of a single class, with a value of 1.0 indicating perfect homogeneity, and a value of 0.0 indicating completely mixed clusters.

The formulas for both metrics are provided below:

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left(y_i = \hat{y}_i\right) \tag{5.1}$$

Where N is the total number of samples, y_i is the true label of sample i, and $\hat{y}_i$ is the predicted label based on the majority class in the k-NN grouping.

$$\begin{aligned} \text{Homogeneity} &= 1 - \frac{H(C|K)}{H(C)}, \quad \text{if } H(C) > 0, \quad \text{else } 1.0 \\ H(C|K) &= -\sum_{k=1}^{K} \sum_{c=1}^{C} \frac{N_{c,k}}{N} \log\left(\frac{N_{c,k}}{N_k}\right) \\ H(C) &= -\sum_{c=1}^{C} \frac{N_c}{N} \log\left(\frac{N_c}{N}\right) \end{aligned} \tag{5.2}$$

Where $H(C|K)$ is the conditional entropy of the classes given the clusters, $H(C)$ is the entropy of the classes, $N_{c,k}$ is the number of samples of class c in cluster k, N_k is the total number of samples in cluster k, N_c is the total number of samples of class c, and N is the total number of samples. The entropy is a measure of how mixed the classes are within the clusters, with lower values indicating better separation, while the conditional entropy quantifies the uncertainty of class labels given the cluster assignments.

In Table 5.10, just hard labels were considered, so the task type for the images and action readout embeddings, and the gripper type for the gripper point cloud embeddings. On the other hand, in Table 5.11, timestep percentage-based labels are also considered, together with the task labels, for the images and action readout embeddings, to assess how well the temporal structure is captured by the learned embeddings. The timestep percentage-based labels are defined by dividing the episode into 5 equal segments, and assigning a label based on which segment the timestep belongs to, with a margin of 2% around the segment boundaries to account for variability. The resulting timestep segments are as follows: [0%-19%], [21%-39%], [41%-59%], [61%-79%], [81%-100%]. Those segments' labels are then combined with the hard task labels to obtain the final clusters' labels.

Table 5.10: Representation Learning Contrastive Losses Results: Hard Labels (k-NN: k=3,5,10).

Embedding Type	**k = 3**		**k = 5**		**k = 10**	
	Acc	**Hom**	**Acc**	**Hom**	**Acc**	**Hom**
Main image	1.0	0.99	1.0	0.96	0.98	0.89
Wrist image	1.0	0.96	0.99	0.91	0.98	0.85
Gripper point cloud	1.0	0.99	1.0	0.99	1.0	0.97
Action readout	1.0	0.98	1.0	0.98	1.0	0.96

Table 5.11: Representation Learning Contrastive Losses Results: Timestep + Hard Task Labels (k-NN: k=3,5,10).

Embedding Type	**k = 3**		**k = 5**		**k = 10**	
	Acc	**Hom**	**Acc**	**Hom**	**Acc**	**Hom**
Main image	0.94	0.90	0.92	0.86	0.85	0.76
Wrist image	0.93	0.88	0.90	0.83	0.81	0.70
Action readout	0.94	0.89	0.92	0.87	0.86	0.80

The results with the hard labels confirm the arguments based on the t-SNE visualizations, showing that the learned embeddings are highly structured, with near-perfect accuracy and homogeneity scores across different k values. Additionally, the gripper point cloud embeddings achieve a very high score due to the employment of hard labels during training, but also, the action readout tokens achieve comparable scores without any explicit supervision.

Regarding the results with the soft labels, they follow a pattern similar to the hard label ones, even if with lower scores. This suggests that while the model is proficient at distinguishing between different tasks, it also encodes temporal information in a meaningful way, allowing for the separation of different phases within an episode.

The two quantitative analyses highlight the increased difficulty in structuring the wrist image latent space, likely due to the limited variability in the wrist camera views across different tasks and configurations. Nevertheless, the results prove the effectiveness of the proposed contrastive losses in constructing a well-organized representation space.

It is also worth mentioning that applying contrastive losses directly on the action readout embeddings was tested, but it badly affected the task performance, probably due to the excessive constraints imposed on the action space, so it was avoided in the final finetuning procedure. Instead, the model was able to learn a well-structured action embedding space indirectly, thanks to the supervision provided by the other contrastive losses applied to the observation embeddings.

5.4.2 Task Performance Evaluations

The finetuned vision-based multi-embodiment policy task performance is evaluated on all the robot-gripper-task triplets considered during training, as well as on the unseen robot-gripper configurations for zero-shot generalization assessment. The evaluation procedure consists of running 10 episodes per configuration and computing the average success rate.

During inference, instead of performing a prediction of 4 actions at each time t, the predictions are performed only at even timesteps, and actions t and $t+1$ are executed for each prediction, while actions $t+2$ and $t+3$ are kept to be exponentially averaged and summed to the next prediction, according to the temporal ensembling approach. This choice is motivated by the fact that executing only 1 action per chunk does not provide significant benefits in terms of performance, while it doubles the computational cost at inference time.

In table 5.12, the success rates are aggregated over different dimensions to provide a comprehensive overview of the model's performance. For each task, and each robot/gripper considered, the table reports success rate (SR): over all configurations (including that particular task and robot/gripper); over all configurations having only seen components; and over all configurations with at least one unseen component. Unseen components are displayed in blue, while the robots and the grippers are separated by a midrule.

In table 5.13, the overall success rates are summarized, aggregating the results over all configurations for each task, just dividing between configurations only with seen components and configurations with at least one unseen component.

The per-configuration non-aggregated results are instead reported in Appendix A.2.

Table 5.12: Vision-Based Policy Performance Evaluations

Task	Robot/Gripper	Total SR (%)	Seen SR (%)	Unseen SR (%)
Lift	*Panda*	61.7	77.5	30.0
	IIWA	63.3	72.5	45.0
	Jaco	56.7	65.0	40.0
	Kinova3	65.0	-	65.0
	UR5e	20.0	-	20.0
	PandaGripper	58.0	76.7	30.0
	Rethink	34.0	40.0	25.0
	Robotiq140	78.0	86.7	65.0
	RobotiqThreeFinger	86.0	83.3	90.0
	Robotiq85	62.0	-	62.0
	JacoThreeFinger	2.0	-	2.0
PickPlaceCan	*Panda*	13.3	20.0	0.0
	IIWA	18.3	22.5	10.0
	Jaco	20.0	22.5	15.0
	Kinova3	10.0	-	10.0
	UR5e	0.0	-	0.0
	PandaGripper	14.0	16.7	10.0
	Rethink	4.0	6.7	0.0
	Robotiq140	20.0	30.0	5.0
	RobotiqThreeFinger	24.0	33.3	10.0
	Robotiq85	12.0	-	12.0
	JacoThreeFinger	0.0	-	0.0

Table 5.13: Vision-Based Policy Performance Evaluations Summary

Task	Total SR (%)	Seen SR (%)	Unseen SR (%)
Lift	53.33	71.67	41.11
PickPlaceCan	12.33	21.67	6.11

Analyzing the results, it is evident that the vision-based multi-embodiment policy performs significantly better on the *Lift* task compared to the more complex *PickPlaceCan* task. This discrepancy can be attributed to the increased difficulty of the latter task. The *Lift* task primarily requires the agent to learn a lifting motion and grasping strategy on a cube, placed in an initial position that varies in a small area around the center of the table, in front of the robot base. In contrast, the *PickPlaceCan* task involves a more intricate sequence of actions, including approaching, grasping, lifting, and placing a cylindrical object into a target bin. Furthermore, the initial position of the can is randomized over an area more than an order of magnitude larger than the cube in the *Lift* task, and the presence of small barriers around the target bin adds additional complexity to the placement phase and requires a higher lift and transfer precision.

Another aspect that may have significantly hindered the performance on the *PickPlaceCan* task is the higher diversity in the strategy adopted to perform the task across different robot-gripper embodiments. Indeed, different grippers provide different grasping capabilities, differing in the approach direction and object to gripper distance; for instance, the *RethinkGripper* requires a top-down approach to successfully grasp the can, while the *Robotiq140Gripper* allows for a side approach due to its wider finger span, and this aspect is clearly reflected in the final performance results, where the average success rate achieved is 4.0% and 20.0%, respectively. Additionally, different robots have different kinematics, leading to significantly different visual observations while transferring the object to the target bin, which may have further complicated the learning process.

By qualitatively analyzing the execution of the tasks, it is possible to observe that the model also often struggles during the approach phase, leading to failed attempts. This observation suggests that the model may have difficulty in accurately perceiving the can's position from the visual inputs, especially when considering the variability introduced by different robot-gripper embodiments, and in some cases, this results in employing suboptimal or average strategies to approach and grasp the can. In this regard, the contrastive losses seemed to slightly help in understanding common features across embodiments, but not enough to achieve significant performance, as evidenced by the ablation studies presented in the next Section 5.4.3.

Regarding the more successful *Lift* task, the results indicate that the model is able to generalize reasonably well to unseen robot and gripper configurations, achieving a success rate of 41.11% on configurations with at least one unseen component. This outcome suggests that the contrastive losses employed during finetuning effectively encouraged the model to learn embodiment-invariant features, enabling it to adapt to new configurations not encountered during training. Additionally, the high success rates achieved on configurations with only seen components (71.67%) demonstrate that the model is capable of leveraging the knowledge acquired during training to perform well on familiar setups.

The success rate on seen robots is pretty consistent across tasks, with the *Panda* achieving 77.5% on *Lift* and 20.0% on *PickPlaceCan*, the *IIWA* achieving 72.5% and 22.5%, and the *Jaco* achieving 65.0% and 22.5%. This consistency suggests that the model is able to effectively utilize the knowledge acquired to bridge the gap between different robot embodiments. For instance, the *Jaco* robot is smaller and requires a longer approach to the object compared to the *Panda* and *IIWA*, but the model is still able to adapt its strategy accordingly. It is worth noting that, even if the *UR5e* robot has a different appearance, movement style, and initial position compared to the training robots, the model is still able to achieve reasonable performance in the *Lift* task (20.0% success rate), while it completely fails in solving the *PickPlaceCan* task due to the aforementioned differences. On the other hand, the *Kinova3* robot, which has a more similar appearance and movement style to the training robots, achieves a higher success rate in both tasks, even outperforming the seen robots in the *Lift* task. This outcome suggests that visual similarity and kinematic resemblance to the training robots may play a significant role in the model's ability to generalize to unseen robot embodiments.

Concerning grippers, the results show a more varied performance across tasks. It is possible to observe that the *Robotiq* grippers generally outperform the other grippers, likely due to their wider finger span and more stable grasping capabilities. In particular, the *Robotiq140* achieves the highest success rates in both tasks, and the unseen *Robotiq85* is able to outperform the *Rethink* in both tasks. On the other hand, the almost complete inability to achieve a successful execution displayed by the *JacoThreeFinger* can be attributed to several factors. Firstly, this gripper, even if it has three fingers like the *RobotiqThreeFinger*, has a significantly different design and requires a more precise grasping strategy, which may not have been adequately represented in the training dataset. Furthermore, the limited number of demonstrations involving three-finger grippers during training may have hindered the model's ability to learn effective grasping strategies for this specific gripper type.

Comparison with Octo-Small Baseline

To provide some context, the performance achieved by the proposed vision-based multi-embodiment policy can be compared with the results reported by Ghosh *et al.* [2] for the Octo model. Even if the experimental setups are not directly comparable, since different simulation environments and tasks are considered, some observations can still be made.

In the paper, the Octo-Small model is fine-tuned on 4 different tasks with approximately 100 expert demonstrations each. It is important to note that for each task, only a single robot-gripper embodiment is used during training, and the model is fine-tuned on one task at a time. The reported success rates range from 70% to 90% on the in-distribution configurations and for novel object instances, while the performance on new environments is lower, with a success rate of approximately 40%.

These results are comparable to the ones achieved by the proposed method on the *Lift* task, where a 71.67% success rate is obtained on seen configurations and a 41.11% success rate on unseen configurations. Notably, in this work, the model is trained to handle multiple robot-gripper embodiments and two different tasks simultaneously, with machine-generated suboptimal demonstrations. The performance on the more complex *PickPlaceCan* task is lower, with a 21.67% success rate on seen configurations and a 6.11% success rate on unseen configurations, highlighting the challenges associated with multi-embodiment and multi-task learning in vision-based robotic manipulation.

5.4.3 Ablation Studies

To better understand the impact of the different components of the proposed finetuning procedure on the embodiment generalization capabilities, a series of ablation studies is conducted. In particular, the following aspects are analyzed:

- **Contrastive Losses**: The model is finetuned without the contrastive loss terms in the overall loss function (Equation (4.12)).
- **Gripper Representation Input**: The model is finetuned without the gripper point cloud input.
- **Weighted BC**: The model is finetuned without the weighted BC loss term (Equation (4.14)), only using the standard BC loss (Equation (2.26)).
- **Pretraining**: The model is finetuned starting from a randomly initialized transformer backbone, instead of using the pretrained weights.

The results of the ablation studies are reported in Figure 5.9 and Figure 5.10, where the summarized success rates for each configuration are displayed for the *Lift* and *PickPlaceCan* tasks, respectively. The grouping is the same proposed in Table 5.13.

The ablation results indicate that, even if the model achieves the highest success rates when all components are included, the removal of individual components does not drastically affect the overall performance, especially for the *Lift* task. The only exception is represented by the removal of the pretrained weights, which leads to a significant drop in performance, highlighting the importance of the pretrained representation for effective learning of generalist agents.

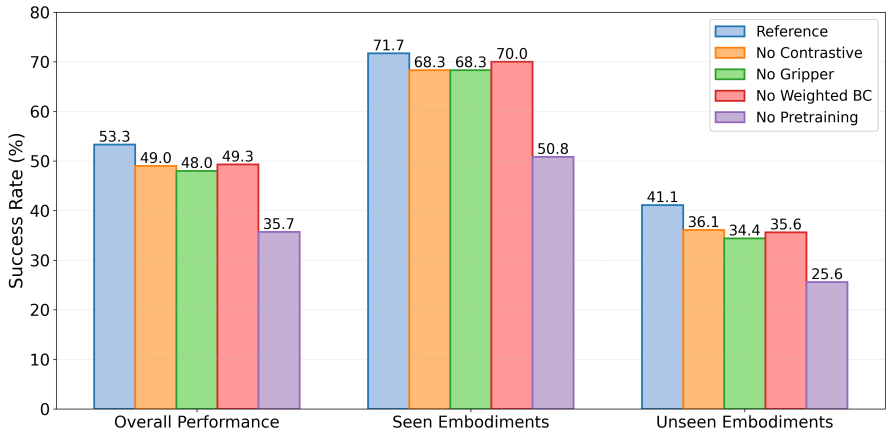


Figure 5.9: Ablation studies for the *Lift* task

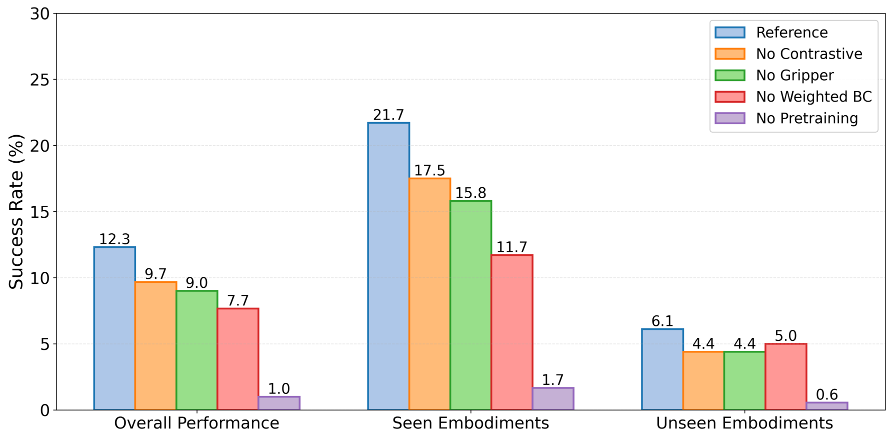


Figure 5.10: Ablation studies for the *PickPlaceCan* task

Analyzing the results for the *PickPlaceCan* task, it is possible to observe a more pronounced impact of the different components on the performance. In particular, the removal of the weighted BC loss leads to a significant decrease in success rates, especially for seen configurations, indicating that this component plays a crucial role in guiding the model towards effective strategies for the more complex task. On the other hand, the removal of the contrastive losses and the gripper representation input has a more moderate effect on performance, suggesting that while these components contribute to embodiment generalization, they are not as critical as the weighted BC loss for task success.

Consequently, the use of the contrastive losses can still be considered beneficial, as a more structured representation space is learned, which may facilitate further finetuning or adaptation to new tasks and embodiments in future work. Conversely, the use of the gripper point cloud representation as an additional input is less justified, as its removal seems to have a limited impact on the final performance, and its inclusion does not provide a clear advantage. This suggests that a different approach might be necessary to effectively incorporate gripper-specific information into the model, especially considering the proven importance of this aspect in multi-embodiment manipulation task settings.

Diffusion Action Decoder Ablation

Similar to what was proposed by Ghosh *et al.* [2], preliminary experiments were conducted to assess the effectiveness of employing a diffusion-based action decoder instead of a standard regression-based action decoder. As discussed in previous sections, the diffusion-based decoder is expected to better capture the multimodal nature of the action distribution and to leave more flexibility to the transformer backbone in structuring the learned representation space. This aspect was particularly considered important given the multi-embodiment nature of the problem addressed in this work.

Indeed, during preliminary experiments, when employing a simple MLP-based action decoder, the model struggled to learn effective policies, achieving lower task performance and less structured representations. Following these observations, the diffusion-based action decoder was adopted for all the experiments reported in this work, and the MLP-based decoder was not considered further.

Chapter 6

Conclusions

6.1 Discussion

This thesis addresses the challenge of cross-embodiment generalization in robotic manipulation, focusing on two main contributions: a novel dataset generation and knowledge transfer pipeline for RL training, and a contrastive, language-conditioned vision-based policy architecture for multi-embodiment control.

The first contribution proved to be a powerful tool for overcoming the scarcity of embodiment-diverse datasets. By aligning state and action spaces and leveraging advantage-weighted behavioral cloning, the approach enabled efficient RL even for the more complex *PickPlaceCan* task. The experiments showed that this pipeline not only accelerates learning but also mitigates common issues such as reward hacking and pathological movements, with respect to pure RL. However, the quality of the adapted demonstrations and the differences between source and target embodiments remain critical. Additionally, when generating the dataset, the demonstration quality is higher than standard RL exploration, but still sub-optimal with respect to expert performance, which can hinder subsequent learning.

Regarding the vision-based policy, the contrastive supervised learning framework successfully created a shared representation space that facilitated generalization across multiple embodiments. The use of a diffusion model for action generation allowed for flexible and expressive control, while the language conditioning enabled task versatility. The experimental results demonstrated strong performance on both seen and unseen robot-gripper-task combinations for the *Lift* task, highlighting the potential of the method for cross-embodiment generalization. Nonetheless, challenges remain in scaling to more complex tasks, such as *PickPlaceCan*, where performance dropped notably. This suggests that while the architecture is promising, further refinements are needed to handle manipulation scenarios where precise object interactions and movement are critical.

Those findings underscore the importance of high-quality demonstrations, robust representation learning, and the need for architectures that can effectively capture the nuances of complex tasks to achieve reliable cross-embodiment generalization. Although significant progress has been made, the limitations observed provide some clear directions for future research.

6.2 Future Directions

Building on the insights gained from this thesis, several future research directions can be pursued:

- **Knowledge Transfer Enhancements:** Explore more advanced techniques to re-utilize knowledge acquired from previous embodiments and tasks, such as meta-learning or continual learning approaches, to further improve sample efficiency and adaptability of state-based RL agents.
- **Dataset Quality and Scalability:** Investigate methods to enhance the quality of the generated demonstrations. Additionally, understand if it is possible to obtain similar performance with smaller datasets, without every possible robot-gripper-task combination included, enabling better scalability to a wider range of embodiments and tasks.
- **Complex Task Handling:** Investigate enhancements to the vision-based policy architecture and training procedures to better handle complex tasks such as *PickPlaceCan*. This could involve incorporating hierarchical learning, or exploring alternatives to include RL in the training process.
- **Real-World Deployment:** Even if the vision-based policy was pretrained with real-world demonstrations, the proposed approach has only been validated in simulated environments. Future work could focus on transferring the learned policies to real robotic systems to assess their performance and adaptability in practical scenarios.

Appendix A

Appendix

A.1 Vision-Based Policy Additional Experiments

In this section, additional experiments related to the vision-based policy are reported. In particular, the use of multiple diffusion heads for action generation, as well as the possibility of further finetuning the model using RL after the supervised training phase, are discussed. Finally, the inclusion of state-based inputs alongside visual observations is considered.

A.1.1 Multiple Action Heads

While developing the vision-based policy architecture described in Section 4.3, different design choices were explored. One such choice that is worth mentioning is the use of multiple diffusion heads for action generation, each specialized for a specific gripper type, instead of a single shared diffusion head for all grippers. The rationale behind this experiment is that different grippers may require distinct action distributions due to their unique kinematics and interaction capabilities, and thus having separate heads could potentially improve performance. The idea is also motivated by prior works [43, 52], which propose using different policy heads for different embodiments.

For this experiment, the model is finetuned using the same training procedure as described in Section 4.3, with the only difference being the use of separate diffusion heads for each gripper type, computing the loss for different grippers through masking. Even though the training converged successfully, the evaluation results showed no significant improvement compared to the single-head architecture, but rather a slight decrease in performance even when dealing with the easier *Lift* task. This suggests that the additional complexity introduced by multiple heads is not beneficial in this context, possibly due to the fact that the diffusion model is already capable of capturing the necessary variations in action distributions across different grippers within a single head.

Additionally, using multiple heads increases the model size and computational requirements, but also significantly limits the ability to generalize, since unseen grippers cannot be handled without a dedicated head. Therefore, the single-head architecture was preferred for its simplicity, efficiency, and better generalization capabilities.

A.1.2 RL Finetuning

After the supervised finetuning of the vision-based policy, an additional RL finetuning phase was explored to further improve the agent's performance, especially for the more complex *PickPlaceCan* task. The idea was to leverage the trained model as a strong initialization point and then refine the policy using RL techniques. The RL finetuning was performed optimizing only the diffusion head parameters, while keeping the rest of the model frozen, to reduce the computational complexity and avoid catastrophic forgetting of the learned representations.

Mainly online RL methods were considered, due to the fact that through the employment of weighted BC during the supervised training phase, the model has already been using most of the available information from the dataset. The procedures tested included a variation of the PPO algorithm specifically designed for diffusion policies [73], as well as a simpler TD3-based approach where the base imitation learning policy was used to aid the RL-head exploration process [59].

The tests were performed only on the *PickPlaceCan* task, since the *Lift* task was already solved with high success rates. However, the results did not show any significant improvement over the purely supervised finetuned model, with the performance remaining roughly the same. This could be due to several factors, including the limited exploration capabilities of the diffusion head when the rest of the model is frozen, providing an insufficient representation for an effective RL optimization. Indeed, as previously mentioned, the model seems to struggle to capture the can position, and this issue might be the limiting factor for this kind of approach.

Future work could explore different strategies to integrate RL into the training process, such as complete finetuning of the entire model as proposed by Wang *et al.* [74], at the cost of increased computational requirements, or reducing model size and complexity.

A.1.3 State Inputs

Another additional experiment involved augmenting the vision-based policy with state-based inputs alongside visual observations. The motivation behind this approach is that state information, such as robot end-effector position and orientation, can provide valuable context that may enhance the policy's decision-making capabilities, especially in complex manipulation tasks. The state inputs were concatenated to the other input embeddings as additional tokens before being processed by the ViT backbone. The rest of the architecture and training procedure remained unchanged from the original vision-based policy.

The results from this experiment, compared with the standard vision-based policy, are summarized in Table A.1.

Table A.1: Vision-Based Policy Performance With State Inputs VS Without State Inputs

Task	**Method**	**Total SR (%)**	**Seen SR (%)**	**Unseen SR (%)**
Lift	Reference	53.33	71.67	41.11
	+ State Inputs	48.33	75.83	30.0
PickPlaceCan	Reference	12.33	21.67	6.11
	+ State Inputs	11.00	19.17	5.56

As is clear from the results, the inclusion of state inputs does not lead to significant performance improvements compared to the standard vision-based policy. While there is a slight increase in success rates for seen configurations in the *Lift* task, there are some decreases in performance in the other settings, especially for unseen *Lift* configurations. This suggests that while state information can provide additional context, the vision-based policy is already capable of extracting sufficient information from visual observations alone. Moreover, the addition of state inputs might introduce redundancy or even confusion in the learning process, potentially hindering the model's ability to generalize effectively across different embodiments and tasks.

A.2 Additional Figures and Results

In this section, additional figures and results related to the methodology and experiments are provided for completeness.

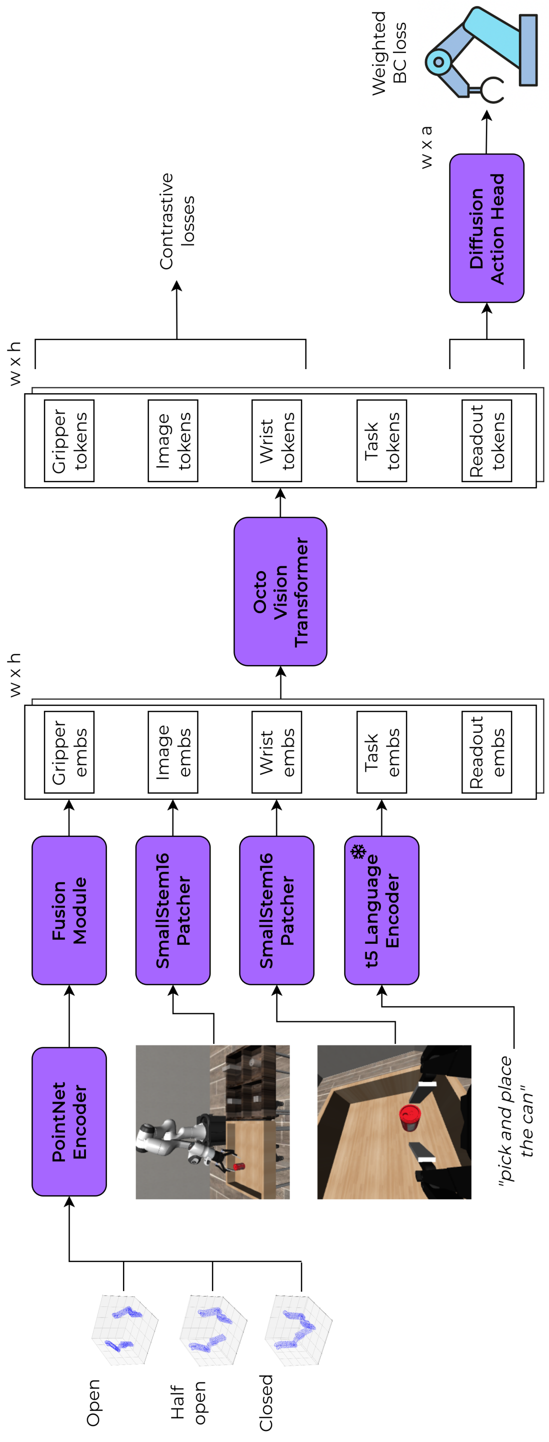


Figure A.1: Overview of the vision-based multi-embodiment policy architecture in horizontal format for better readability, as an alternative to Figure 4.2.

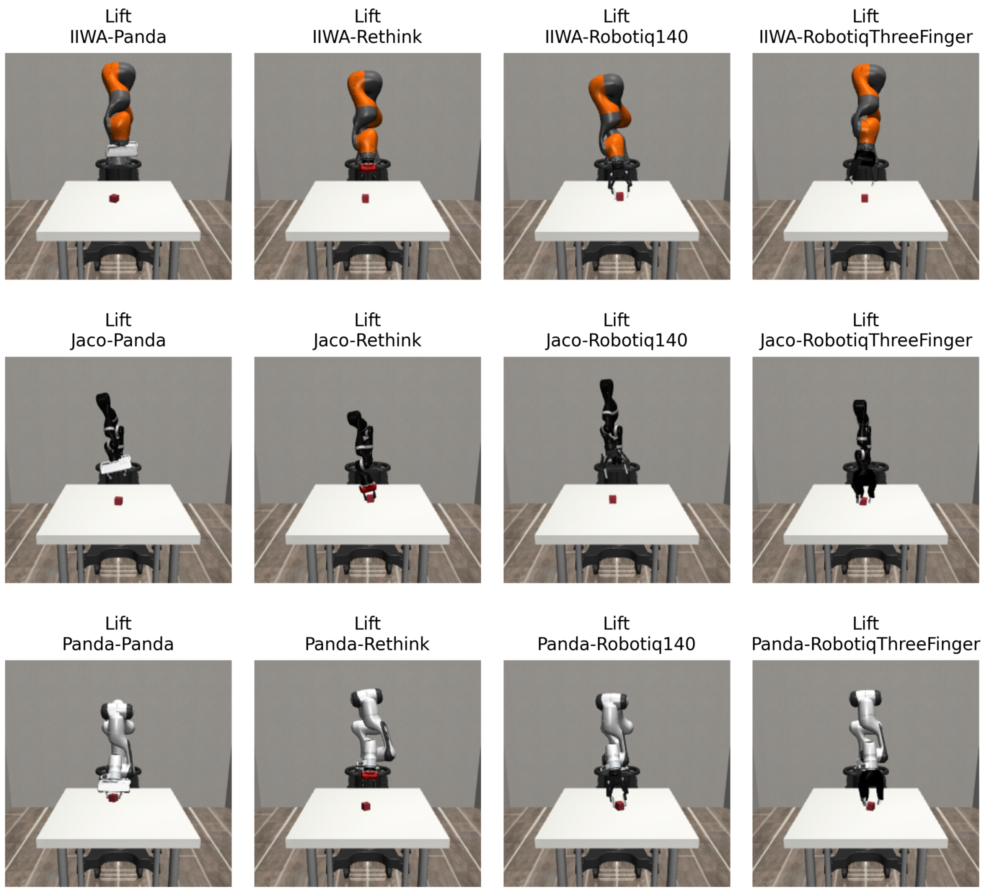


Figure A.2: Visualization of all robot-gripper configurations solved by the state-based agents for the *Lift* task.

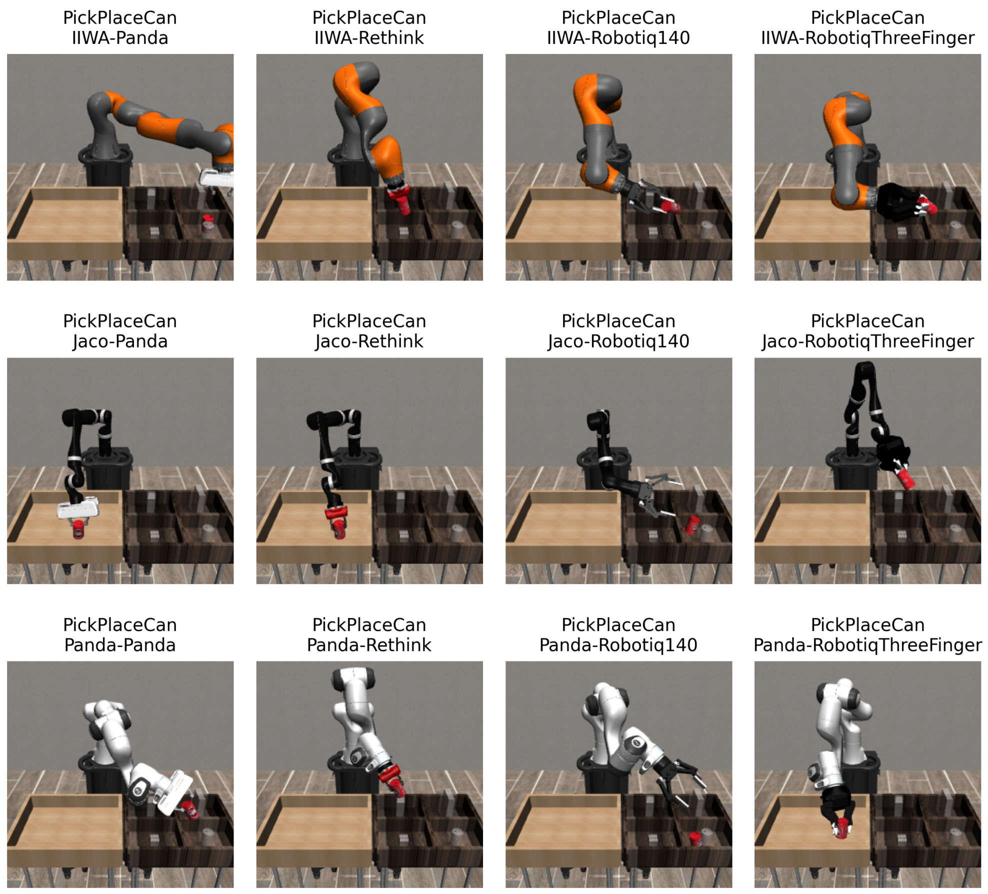


Figure A.3: Visualization of all robot-gripper configurations solved by the state-based agents for the *PickPlaceCan* task.

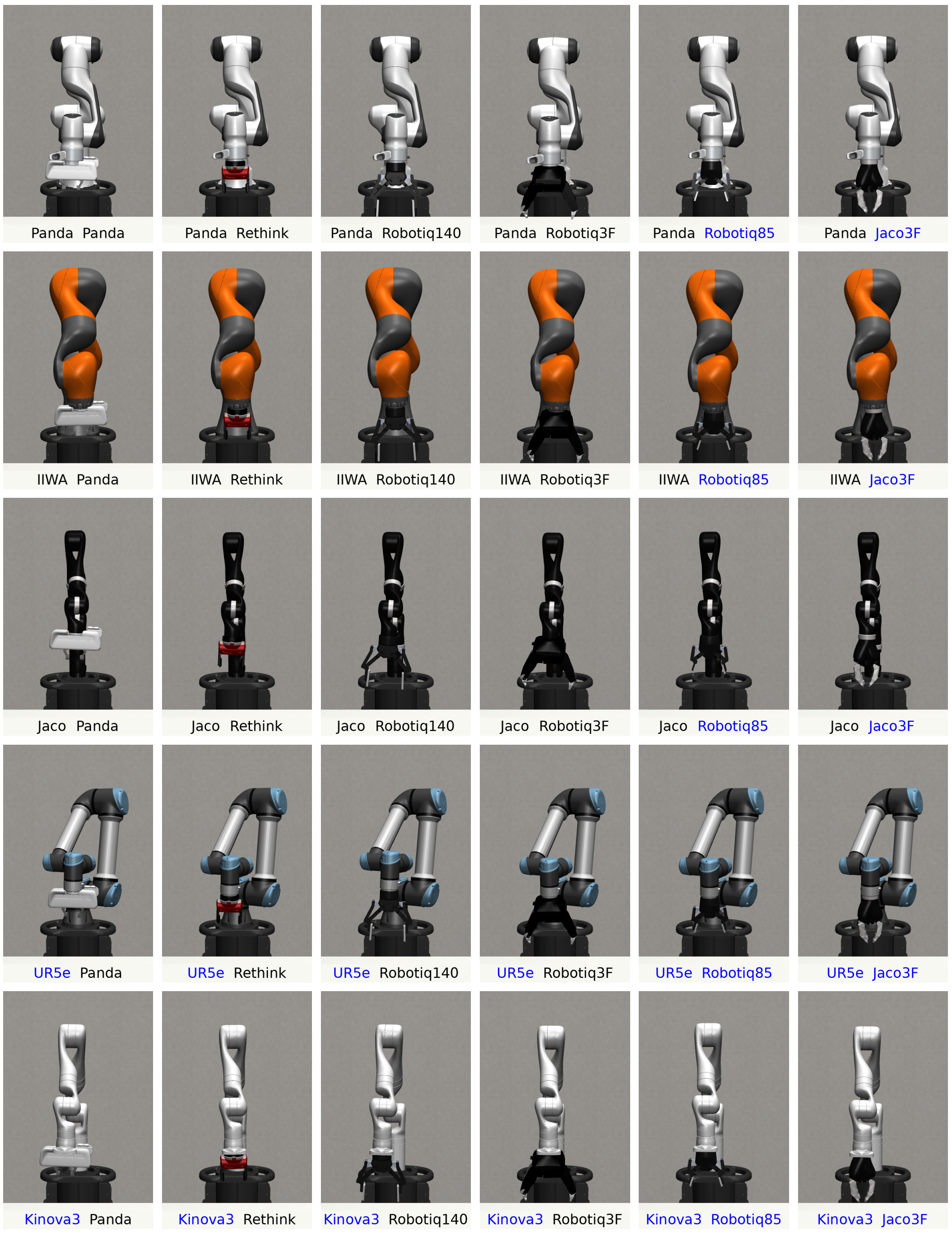


Figure A.4: Visualization of all robot-gripper configurations (5 robots $\times$ 6 grippers) considered in the experiments with the vision-based policy. The unseen components during training are reported in blue in the titles of each image.

Table A.2: Vision-Based Multi-Embodiment Per-Configuration Performance Evaluations

Robot	**Gripper**	***Lift*** **SR (%)**	***PickPlaceCan*** **SR (%)**
Panda	*Panda*	90	20
	Rethink	20	20
	Robotiq140	100	0
	RobotiqThreeFinger	100	40
	Robotiq85	60	0
	JacoThreeFinger	0	0
IIWA	*Panda*	60	20
	Rethink	60	0
	Robotiq140	80	20
	RobotiqThreeFinger	90	50
	Robotiq85	90	20
	JacoThreeFinger	0	0
Jaco	*Panda*	80	10
	Rethink	40	0
	Robotiq140	80	70
	RobotiqThreeFinger	60	10
	Robotiq85	80	30
	JacoThreeFinger	0	0
UR5e	*Panda*	0	0
	Rethink	10	0
	Robotiq140	30	0
	RobotiqThreeFinger	80	0
	Robotiq85	0	0
	JacoThreeFinger	0	0
Kinova3	*Panda*	60	20
	Rethink	40	0
	Robotiq140	100	10
	RobotiqThreeFinger	100	20
	Robotiq85	80	10
	JacoThreeFinger	10	0

Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. u. Kaiser, and I. Polosukhin, «Attention is all you need», in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017 (cit. on pp. 4, 6).

[2] D. Ghosh *et al.*, «Octo: An open-source generalist robot policy», in *Robotics: Science and Systems XX*, Robotics: Science and Systems Foundation, Jul. 15, 2024, ISBN: 979-8-9902848-0-7. DOI: `10.15607/RSS.2024.XX.090` (cit. on pp. 6, 8, 18, 20, 22, 33, 34, 51, 62, 65).

[3] A. Dosovitskiy *et al.*, «An image is worth 16x16 words: Transformers for image recognition at scale», presented at the International Conference on Learning Representations, Oct. 2, 2020 (cit. on pp. 6, 7, 22, 51).

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of deep bidirectional transformers for language understanding*, May 24, 2019. DOI: `10.48550/arXiv.1810.04805`. arXiv: `1810.04805[cs]` (cit. on p. 7).

[5] J. Ho, A. Jain, and P. Abbeel, «Denoising diffusion probabilistic models», in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 6840–6851 (cit. on pp. 7, 51).

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, *Playing atari with deep reinforcement learning*, Dec. 19, 2013. DOI: `10.48550/arXiv.1312.5602`. arXiv: `1312.5602[cs]` (cit. on p. 11).

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, Aug. 28, 2017. DOI: `10.48550/arXiv.1707.06347`. arXiv: `1707.06347[cs]` (cit. on p. 11).

[8] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, «Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor», in *Proceedings of the 35th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, Jul. 3, 2018, pp. 1861–1870 (cit. on pp. 12, 28).

[9] T. Haarnoja *et al.*, *Soft actor-critic algorithms and applications*, Jan. 29, 2019. DOI: `10.48550/arXiv.1812.05905`. arXiv: `1812.05905[cs]` (cit. on pp. 12, 15, 28).

[10] D. P. Kingma and M. Welling, «Auto-encoding variational bayes», Dec. 23, 2013 (cit. on p. 14).

[11] Y. Bengio, A. Courville, and P. Vincent, «Representation learning: A review and new perspectives», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013, ISSN: 1939-3539. DOI: `10.1109/TPAMI.2013.50` (cit. on p. 17).

[12] C. D'Eramo, D. Tateo, A. Bonarini, M. Restelli, and J. Peters, «Sharing knowledge in multi-task deep reinforcement learning», presented at the International Conference on Learning Representations, Sep. 23, 2019 (cit. on p. 18).

[13] R. Boige, Y. Flet-Berliac, A. Flajolet, G. Richard, and T. Pierrot, *PASTA: Pretrained action-state transformer agents*, Dec. 4, 2023. DOI: `10.48550/arXiv.2307.10936`. arXiv: `2307.10936[cs]` (cit. on p. 18).

[14] C. Ying, Z. Hao, X. Zhou, X. Xu, H. Su, X. Zhang, and J. Zhu, *PEAC: Unsupervised pre-training for cross-embodiment reinforcement learning*, Nov. 18, 2024. DOI: `10.48550/arXiv.2405.14073`. arXiv: `2405.14073[cs]` (cit. on p. 18).

[15] T. Zhao, V. Kumar, S. Levine, and C. Finn, «Learning fine-grained bimanual manipulation with low-cost hardware», in *Robotics: Science and Systems XIX*, Robotics: Science and Systems Foundation, Jul. 10, 2023, ISBN: 978-0-9923747-9-2. DOI: `10.15607/RSS.2023.XIX.016` (cit. on pp. 19, 20, 35, 36).

[16] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. C. Burchfiel, and S. Song, «Diffusion policy: Visuomotor policy learning via action diffusion», presented at the Robotics: Science and Systems XIX, vol. 19, Jul. 10, 2023, ISBN: 978-0-9923747-9-2 (cit. on p. 20).

[17] H. He, C. Bai, K. Xu, Z. Yang, W. Zhang, D. Wang, B. Zhao, and X. Li, «Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning», in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. NIPS '23, Red Hook, NY, USA: Curran Associates Inc., Dec. 10, 2023, pp. 64 896–64 917 (cit. on p. 20).

[18] Z. Liang, Y. Mu, M. Ding, F. Ni, M. Tomizuka, and P. Luo, «AdaptDiffuser: Diffusion models as adaptive self-evolving planners», in *Proceedings of the 40th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, Jul. 3, 2023, pp. 20 725–20 745 (cit. on p. 20).

[19] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, «3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations», presented at the Robotics: Science and Systems XX, vol. 20, Jul. 15, 2024, ISBN: 979-8-9902848-0-7 (cit. on p. 20).

[20] D. Morrison, J. Leitner, and P. Corke, «Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach», presented at the Robotics: Science and Systems XIV, vol. 14, Jun. 26, 2018, ISBN: 978-0-9923747-4-7 (cit. on p. 20).

[21] H. Kasaei and M. Kasaei, *MVGrasp: Real-time multi-view 3d object grasping in highly cluttered environments*, Oct. 5, 2022. DOI: `10.48550/arXiv.2103.10997`. arXiv: `2103.10997[cs]` (cit. on p. 20).

[22] S. Song, A. Zeng, J. Lee, and T. Funkhouser, «Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations», *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4978–4985, Jul. 2020, ISSN: 2377-3766. DOI: `10.1109/LRA.2020.3004787` (cit. on p. 20).

[23] W. Yuan, A. Murali, A. Mousavian, and D. Fox, «M2t2: Multi-task masked transformer for object-centric pick and place», presented at the 7th Annual Conference on Robot Learning, Aug. 30, 2023 (cit. on p. 20).

[24] L. Shao, F. Ferreira, M. Jorda, V. Nambiar, J. Luo, E. Solowjow, J. A. Ojea, O. Khatib, and J. Bohg, «UniGrasp: Learning a unified model to grasp with multifingered robotic hands», *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2286–2293, Apr. 2020, ISSN: 2377-3766. DOI: `10.1109/LRA.2020.2969946` (cit. on pp. 21, 38).

[25] Z. Xu, B. Qi, S. Agrawal, and S. Song, «AdaGrasp: Learning an adaptive gripper-aware grasping policy», in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, ISSN: 2577-087X, May 2021, pp. 4620–4626. DOI: `10.1109/ICRA48506.2021.9560833` (cit. on pp. 21, 38).

[26] N. Khargonkar, N. Song, Z. Xu, B. Prabhakaran, and Y. Xiang, «Neural-Grasps: Learning implicit representations for grasps of multiple robotic hands», in *Proceedings of The 6th Conference on Robot Learning*, ISSN: 2640-3498, PMLR, Mar. 6, 2023, pp. 516–526 (cit. on p. 21).

[27] T. B. Brown *et al.*, «Language models are few-shot learners», in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20, Red Hook, NY, USA: Curran Associates Inc., Dec. 6, 2020, pp. 1877–1901, ISBN: 978-1-7138-2954-6 (cit. on p. 21).

[28] D. Hernandez, J. Kaplan, T. Henighan, and S. McCandlish, *Scaling laws for transfer*, Feb. 2, 2021. DOI: `10.48550/arXiv.2102.01293`. arXiv: `2102.01293[cs]` (cit. on p. 21).

[29] S. Reed *et al.*, «A generalist agent», *Transactions on Machine Learning Research*, Aug. 29, 2022, ISSN: 2835-8856 (cit. on p. 21).

[30] A. Brohan *et al.*, «RT-1: Robotics transformer for real-world control at scale», in *Robotics: Science and Systems XIX*, Robotics: Science and Systems Foundation, Jul. 10, 2023, ISBN: 978-0-9923747-9-2. DOI: `10.15607/RSS.2023.XIX.025` (cit. on p. 21).

[31] K. Bousmalis *et al.*, «RoboCat: A self-improving generalist agent for robotic manipulation», *Transactions on Machine Learning Research*, Sep. 6, 2023, ISSN: 2835-8856 (cit. on p. 21).

[32] R. Yang, G. Chen, C. Wen, and Y. Gao, *FP3: A 3d foundation policy for robotic manipulation*, Mar. 11, 2025. DOI: `10.48550/arXiv.2503.08950`. arXiv: `2503.08950[cs]` (cit. on p. 21).

[33] A. O'Neill *et al.*, «Open x-embodiment: Robotic learning datasets and RT-x models : Open x-embodiment collaboration0», in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 6892–6903. DOI: `10.1109/ICRA57147.2024.10611477` (cit. on pp. 22, 34).

[34] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn, «RoboNet: Large-scale multi-robot learning», in *Proceedings of the Conference on Robot Learning*, ISSN: 2640-3498, PMLR, May 12, 2020, pp. 885–897 (cit. on p. 22).

[35] H. Niu, J. Hu, G. Zhou, and X. Zhan, *A comprehensive survey of cross-domain policy transfer for embodied agents*, Aug. 27, 2024. DOI: `10.48550/arXiv.2402.04580`. arXiv: `2402.04580[cs]` (cit. on p. 22).

[36] W. Liu, H. Zhao, C. Li, J. Biswas, S. Pouya, and Y. Chang, *COMPASS: Cross-embodiment mobility policy via residual RL and skill synthesis*, Feb. 22, 2025. DOI: `10.48550/arXiv.2502.16372`. arXiv: `2502.16372[cs]` (cit. on p. 23).

[37] R. Doshi, H. Walke, O. Mees, S. Dasari, and S. Levine, *Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation*, Aug. 21, 2024. DOI: `10.48550/arXiv.2408.11812`. arXiv: `2408.11812[cs]` (cit. on p. 23).

[38] J. Yang, C. Glossop, A. Bhorkar, D. Shah, Q. Vuong, C. Finn, D. Sadigh, and S. Levine, *Pushing the limits of cross-embodiment learning for manipulation and navigation*, Feb. 29, 2024. DOI: `10.48550/arXiv.2402.19432`. arXiv: `2402.19432[cs]` (cit. on p. 23).

[39] K. Zakka, A. Zeng, P. Florence, J. Tompson, J. Bohg, and D. Dwibedi, *XIRL: Cross-embodiment inverse reinforcement learning*, Dec. 13, 2021. DOI: `10.48550/arXiv.2106.03911`. arXiv: `2106.03911[cs]` (cit. on p. 23).

[40] M. Xu, Z. Xu, C. Chi, M. Veloso, and S. Song, *XSkill: Cross embodiment skill discovery*, Sep. 28, 2023. DOI: `10.48550/arXiv.2307.09955`. arXiv: `2307.09955[cs]` (cit. on p. 23).

[41] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, «R3m: A universal visual representation for robot manipulation», in *Proceedings of The 6th Conference on Robot Learning*, ISSN: 2640-3498, PMLR, Mar. 6, 2023, pp. 892–909 (cit. on p. 23).

[42] A. v. d. Oord, Y. Li, and O. Vinyals, *Representation learning with contrastive predictive coding*, Jan. 22, 2019. DOI: `10.48550/arXiv.1807.03748`. arXiv: `1807.03748[cs]` (cit. on pp. 23, 39).

[43] T. Wang, D. Bhatt, X. Wang, and N. Atanasov, *Cross-embodiment robot manipulation skill transfer using latent space alignment*, Jun. 4, 2024. DOI: `10.48550/arXiv.2406.01968`. arXiv: `2406.01968[cs]` (cit. on pp. 23, 68).

[44] A. Gupta, L. Fan, S. Ganguli, and L. Fei-Fei, «MetaMorph: Learning universal controllers with transformers», presented at the International Conference on Learning Representations, Oct. 6, 2021 (cit. on p. 24).

[45] S. Cho, D. Kim, J. Lee, and S. Hong, *Meta-controller: Few-shot imitation of unseen embodiments and tasks in continuous control*, Dec. 10, 2024. DOI: `10.48550/arXiv.2412.12147`. arXiv: `2412.12147[cs]` (cit. on p. 24).

[46] B. Trabucco, M. Phielipp, and G. Berseth, «AnyMorph: Learning transferable polices by inferring agent morphology», in *Proceedings of the 39th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, Jun. 28, 2022, pp. 21 677–21 691 (cit. on p. 24).

[47] B. Li, H. Li, Y. Zhu, and D. Zhao, «MAT: Morphological adaptive transformer for universal morphology policy learning», *IEEE Transactions on Cognitive and Developmental Systems*, vol. 16, no. 4, pp. 1611–1621, Aug. 2024, ISSN: 2379-8939. DOI: `10.1109/TCDS.2024.3383158` (cit. on p. 24).

[48] N. Bohlinger, G. Czechmanowski, M. Krupka, P. Kicki, K. Walas, J. Peters, and D. Tateo, *One policy to run them all: An end-to-end learning approach to multi-embodiment locomotion*, Apr. 1, 2025. DOI: `10.48550/arXiv.2409.06366`. arXiv: `2409.06366[cs]` (cit. on p. 24).

[49] E. S. Hu, K. Huang, O. Rybkin, and D. Jayaraman, «Know thyself: Transferable visual control policies through robot-awareness», presented at the ICLR 2022 Workshop on Generalizable Policy Learning in Physical World, Apr. 27, 2022 (cit. on p. 24).

[50] L. Y. Chen, K. Dharmarajan, K. Hari, C. Xu, Q. Vuong, and K. Goldberg, «MIRAGE: Cross-embodiment zero-shot policy transfer with cross-painting», presented at the Robotics: Science and Systems XX, vol. 20, Jul. 15, 2024, ISBN: 979-8-9902848-0-7 (cit. on pp. 24, 36).

[51] S. Dasari, M. K. Srirama, U. Jain, and A. Gupta, «An unbiased look at datasets for visuo-motor pre-training», presented at the 7th Annual Conference on Robot Learning, Aug. 30, 2023 (cit. on p. 24).

[52] J. H. Yang, D. Sadigh, and C. Finn, «Polybot: Training one policy across robots while embracing variability», in *Proceedings of The 7th Conference on Robot Learning*, ISSN: 2640-3498, PMLR, Dec. 2, 2023, pp. 2955–2974 (cit. on pp. 25, 38, 68).

[53] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, «Advantage-weighted regression: Simple and scalable off-policy reinforcement learning», Oct. 2, 2020 (cit. on pp. 25, 42).

[54] S. Fujimoto, D. Meger, and D. Precup, «Off-policy deep reinforcement learning without exploration», in *Proceedings of the 36th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, May 24, 2019, pp. 2052–2062 (cit. on p. 25).

[55] I. Kostrikov, A. Nair, and S. Levine, «Offline reinforcement learning with implicit q-learning», presented at the International Conference on Learning Representations, Oct. 6, 2021 (cit. on p. 25).

[56] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, «Overcoming exploration in reinforcement learning with demonstrations», in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, ISSN: 2577-087X, May 2018, pp. 6292–6299. DOI: `10.1109/ICRA.2018.8463162` (cit. on p. 25).

[57] A. Nair, A. Gupta, M. Dalal, and S. Levine, *AWAC: Accelerating online reinforcement learning with offline datasets*, Apr. 24, 2021. DOI: `10.48550/arXiv.2006.09359`. arXiv: `2006.09359[cs]` (cit. on pp. 25, 30).

[58] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine, «Efficient online reinforcement learning with offline data», in *Proceedings of the 40th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, Jul. 3, 2023, pp. 1577–1594 (cit. on p. 25).

[59] H. Hu, S. Mirchandani, and D. Sadigh, «Imitation bootstrapped reinforcement learning», Oct. 13, 2023 (cit. on pp. 25, 69).

[60] Y. Lu *et al.*, «Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios», in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, ISSN: 2153-0866, Oct. 2023, pp. 7553–7560. DOI: `10.1109/IROS55552.2023.10342038` (cit. on pp. 25, 30).

[61] Y.-H. Wu, N. Charoenphakdee, H. Bao, V. Tangkaratt, and M. Sugiyama, «Imitation learning from imperfect demonstration», in *Proceedings of the 36th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, May 24, 2019, pp. 6818–6827 (cit. on p. 26).

[62] H. Xu, X. Zhan, H. Yin, and H. Qin, «Discriminator-weighted offline imitation learning from suboptimal demonstrations», presented at the Deep RL Workshop NeurIPS 2021, Dec. 13, 2021 (cit. on p. 26).

[63] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, K. Lin, A. Maddukuri, S. Nasiriany, and Y. Zhu, *Robosuite: A modular simulation framework and benchmark for robot learning*, Jan. 18, 2025. DOI: `10.48550/arXiv.2009.12293`. arXiv: `2009.12293[cs]` (cit. on pp. 31, 43).

[64] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, «Exploring the limits of transfer learning with a unified text-to-text transformer», *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020, ISSN: 1533-7928 (cit. on pp. 33, 51).

[65] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, «PointNet: Deep learning on point sets for 3d classification and segmentation», in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ISSN: 1063-6919, Jul. 2017, pp. 77–85. DOI: `10.1109/CVPR.2017.16` (cit. on p. 37).

[66] R. Salakhutdinov and G. Hinton, «Learning a nonlinear embedding by preserving class neighbourhood structure», in *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, ISSN: 1938-7228, PMLR, Mar. 11, 2007, pp. 412–419 (cit. on p. 39).

[67] N. Frosst, N. Papernot, and G. Hinton, «Analyzing and improving representations with the soft nearest neighbor loss», in *Proceedings of the 36th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, May 24, 2019, pp. 2012–2020 (cit. on p. 39).

[68] S. Lee, T. Park, and K. Lee, «Soft contrastive learning for time series», *International Conference on Representation Learning*, vol. 2024, pp. 46 815–46 839, May 31, 2024 (cit. on p. 39).

[69] E. Todorov, T. Erez, and Y. Tassa, «MuJoCo: A physics engine for model-based control», in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, ISSN: 2153-0866, Oct. 2012, pp. 5026–5033. DOI: `10.1109/IROS.2012.6386109` (cit. on p. 43).

[70] A. Mandlekar *et al.*, «What matters in learning from offline human demonstrations for robot manipulation», in *Proceedings of the 5th Conference on Robot Learning*, ISSN: 2640-3498, PMLR, Jan. 11, 2022, pp. 1678–1690 (cit. on p. 44).

[71] I. Loshchilov and F. Hutter, «Decoupled weight decay regularization», presented at the International Conference on Learning Representations, Sep. 27, 2018 (cit. on pp. 46, 52).

[72] A. Q. Nichol and P. Dhariwal, «Improved denoising diffusion probabilistic models», in *Proceedings of the 38th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, Jul. 1, 2021, pp. 8162–8171 (cit. on p. 51).

[73] A. Z. Ren, J. Lidard, L. L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz, «Diffusion policy policy optimization», presented at the The Thirteenth International Conference on Learning Representations, Oct. 4, 2024 (cit. on p. 69).

[74] C. Wang, X. Luo, K. W. Ross, and D. Li, «VRL3: A data-driven framework for visual deep reinforcement learning», presented at the Advances in Neural Information Processing Systems, Oct. 31, 2022 (cit. on p. 69).