Embodied Agent with Natural Perceptions: Multi-view, Sequential, and Language Data

A thesis proposal presented

by

Jinghuan Shang

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

Dec 2023

Abstract

Action policy is a mapping from perceptions to controls. The perception data in embodied agent tasks display complex distributions that could be similar to what humans are perceiving. We name these as "natural perceptions". Such data may include images from different viewpoints, and multi-modal data like pixels and natural languages.

Despite conventional Reinforcement Learning (RL) and Imitation Learning (IL) algorithms are generally applicable, challenges on understanding natural perceptions may cause policy learning to fail. In this thesis proposal, we first explore building viewpoint-agnostic representations by feature disentanglement. This enables the agent to imitate from third-person view demonstrations. We then extend the concept of viewpoint-agnostic representation to generic visual understanding of images and videos. We propose 3DTRL that couples estimated 3D information with pixels to help generalization across viewpoints.

Secondly, we investigate policy learning as a sequence modeling problem, where a multi-modal trajectory including the past states, actions and rewards are all used for making the next action decision. In the straightforward sequence modeling methods, state, action, and reward representations plainly attend to each other in self-attention. We propose StARformer that uses both short-term and long-term modeling to improve overall sequence modeling capacity, especially when the past trajectory is long.

In the third part, we explore embodied agent with active vision ability. In the previous parts, the agent is granted hand-coded views and does not have independent control for them. Such limited flexibility of views may be insufficient for policy. We propose SUGARL, an algorithm that jointly learns a sensory policy to control the view and a motor policy to complete the task. By actively controlling the view, the agent can achieve better performance than standard RL, especially when the information from each individual view is limited.

To conclude the thesis, we will investigate how natural languages can be used together with visual perception and can benefit action policies. Large Language Models (LLMs) or Vision-Language Models (VLMs) have abundant commonsense knowledge about daily scenarios and activities. We propose to leverage such general knowledge to optimize the action planning to achieve better efficiency, or to guide the active vision policy to maximize task-related information.

Contents

Chapter 1

Introduction

1.1 Overview

Action policy is a mapping from perceptions to controls, and can be obtained using Reinforcement Learning (RL) [95, 132, 170, 259] and Imitation Learning (IL) [106, 164, 241]. To successfully obtain an action policy, one key aspect is understanding perception data [61, 85, 232]. Since the embodied agent interacts with the environment with a physical body [225], the perception data display complex distributions which could be similar to what humans are perceiving – a sequence of multi-modal, continuously changing data that correlate with actions. We name this "natural perceptions".

Natural perceptions exist in a wide range of tasks. For example, a ground mobility robot moving around will have the visual observations from continuously changing viewpoints. To make better decisions, a robot can leverage all past experiences composed by a sequence of observations and actions [35, 117]. An agent tries to learn from observing human activities needs to align the third-person view observations with its own ego-centric view [230]. An assistant robot collaborating with human may also require natural language as instructions or feedbacks in addition to visual observations [180].

Despite conventional RL and IL algorithms are generally applicable to policy learning, challenges on understanding natural perceptions may cause many failures or sub-par performance. For instance, an action policy trained on observation from top-down view is hard to generalize to another view facing the robot in front of it [109, 211, 230]. A robot operates on a static hand-coded view can not handle the cases where there are occlusions or target objects being out-of-view [109, 213].

Motivated by the challenges in natural perceptions, in this thesis proposal, we

investigate multiple solutions including learning viewpoint-agnostic representations, sequence modeling, and active vision. The main contributions in this thesis proposal can be summarized as follows:

- We explore the third-person imitation learning problem and build viewpoint-agnostic representations by feature disentanglement. This enables the agent to learn from third-person view demonstrations and imitate the behavior when it only has its ego-centric view observations.
- We extend the concept of viewpoint-agnostic representation to generic visual understanding of images and videos. We propose 3DTRL that couples estimated 3D information with pixels to help vision model generalize to different viewpoints, including robot tasks.
- We investigate policy learning as a sequence modeling problem, where a multi-modal trajectory including the past states, actions and rewards are all used for making the next action decision. We propose StARformer that uses both short-term and long-term modeling to improve overall sequence modeling capacity, especially when the past trajectory is long.
- We explore creating an embodied agent with active vision ability. Taking inspirations from human's sensorimotor stage, we propose SUGARL, an algorithm that jointly learns a sensory policy to control the view and a motor policy to complete the task. By actively controlling the view, the agent can achieve better performance than standard RL, especially when the information from each individual view is limited.

To conclude the thesis, we will investigate how natural languages can be used together with visual perception and thus benefit action policies. Large Language Models (LLMs) [28, 64, 181, 244] or Vision-Language Models (VLMs) [143, 148, 194] are shown to have abundant commonsense knowledge about daily scenarios and activities. We propose to leverage such general knowledge to optimize the action planning [26, 27, 111, 146, 256] to achieve better efficiency, or to guide the active vision policy to maximize task-related information. This formulation could be viewed as imitation learning at natural language level, since demonstrative instructions are usually provided to drive LLM generating desired responses.

1.2 Organization

In Chapter 3, we introduce our method of building viewpoint-agnostic representation using feature disentanglement. Humans learn to imitate by observing others.

However, robot imitation learning generally requires expert demonstrations in the first-person view (FPV). Collecting such FPV videos for every robot could be very expensive. Third-person imitation learning (TPIL) is the concept of learning action policies by observing other agents in a third-person view (TPV), similar to what humans do. This ultimately allows utilizing human and robot demonstration videos in TPV from many different data sources, for the policy learning. In this paper, we present a TPIL approach for robot tasks with *egomotion*. Although many robot tasks with ground/aerial mobility often involve actions with camera egomotion, study on TPIL for such tasks has been limited. Here, FPV and TPV observations are visually very different; FPV shows egomotion while the agent appearance is only observable in TPV. To enable better state learning for TPIL, we propose our disentangled representation learning method. We use a dual auto-encoder structure plus representation permutation loss and time-contrastive loss to ensure the state and viewpoint representations are well disentangled. Our experiments show the effectiveness of our approach.

In Chapter 4, we show how the viewpoint-agnostic representation could be extended to generic visual understanding problems. Humans are remarkably flexible in understanding viewpoint changes due to visual cortex supporting the perception of 3D structure. In contrast, most of the computer vision models that learn visual representation from a pool of 2D images often fail to generalize over novel camera viewpoints. Recently, the vision architectures have shifted towards convolution-free architectures, visual Transformers, which operate on tokens derived from image patches. However, these Transformers do not perform explicit operations to learn viewpoint-agnostic representation for visual understanding, as in convolutions. To this end, we propose a 3D Token Representation Layer (3DTRL) that estimates the 3D positional information of the visual tokens and leverages it for learning viewpoint-agnostic representations. The key elements of 3DTRL include a pseudo-depth estimator and a learned camera matrix to impose geometric transformations on the tokens. These enable 3DTRL to recover the 3D positional information of the tokens from 2D patches. In practice, 3DTRL is easily plugged-in into a Transformer. Our experiments demonstrate the effectiveness of 3DTRL in many vision tasks including image classification, multi-view video alignment, and action recognition. The models with 3DTRL outperform their backbone Transformers in all the tasks with minimal added computation.

In Chapter 5, we demonstrate the capability of sequence modeling in offline Reinforcement Learning (offline-RL) as well as Imitation Learning (or Behavioral Cloning, BC), and how different scales of temporal modeling improve the performance. Reinforcement Learning (RL) can be considered as a sequence modeling

task, where an agent employs a sequence of past state-action-reward experiences to predict a sequence of future actions. In this work, we propose **St**ate-**A**ction-**R**eward Transformer (**StAR**former), a Transformer architecture for robot learning with image inputs, which explicitly models *short-term* state-action-reward representations (StAR-representations), essentially introducing a Markovian-like inductive bias to improve *long-term* modeling. StARformer first extracts StAR-representations using self-attending patches of image states, action, and reward tokens within a short temporal window. These StAR-representations are combined with pure image state representations, extracted as convolutional features, to perform self-attention over the whole sequence. Our experimental results show that StARformer outperforms the state-of-the-art Transformer-based method on image-based Atari and DeepMind Control Suite benchmarks, under both offline-RL and imitation learning settings. We find that models can benefit from our combination of patch-wise and convolutional image embeddings. StARformer is also more compliant with longer sequences of inputs than the baseline method. Finally, we demonstrate how StARformer can be successfully applied to a real-world robot imitation learning setting via a human-following task.

In Chapter 6, we introduce active vision agents in RL, and show that active control of perception can achieve better task performance than standard RL. We investigate Active Vision Reinforcement Learning (ActiveVision-RL), where an embodied agent simultaneously learns action policy for the task while also controlling its visual observations in partially observable environments. We denote the former as *motor policy* and the latter as *sensory policy*. For example, humans solve real world tasks by hand manipulation (motor policy) together with eye movements (sensory policy). ActiveVision-RL poses challenges on coordinating two policies given their mutual influence. We propose SUGARL, Sensorimotor Understanding Guided Active Reinforcement Learning, a framework that models motor and sensory policies separately, but jointly learns them using with an intrinsic sensorimotor reward. This learnable reward is assigned by sensorimotor reward module, incentivizes the sensory policy to select observations that are optimal to infer its own motor action, inspired by the sensorimotor stage of humans. Through a series of experiments, we show the effectiveness of our method across a range of observability conditions and its adaptability to existed RL algorithms. The sensory policies learned through our method are observed to exhibit effective active vision strategies.

In Chapter 7, we conclude this thesis proposal and discuss the future work. We propose to investigate how natural languages can be used together with visual perception and thus benefit action policies. Large Language Models (LLMs) [28,

64, 181, 244] or Vision-Language Models (VLMs) [143, 148, 194] are shown to have abundant commonsense knowledge about daily scenarios and activities. LLMs also have shown quantum performance leap on many embodied tasks, including open-world exploration in Minecraft [256], real-robot manipulation [4, 24, 26, 27, 69, 218, 276], and navigation [272]. One line of recent work is LLM-based task planning, where the user directly prompts the off-the-shelf LLMs with examples and task descriptions, and get LLM generated step-by-step plans in language space. Among all kinds of languages, programming languages becomes good proxies between LLM and actual robot control due to its unique structured but explainable nature [111, 146]. Despite the promising long-term planning capability, the generated plan simply follows the most straightforward way to complete task, which is not always optimal in terms of execution effort like distance and time cost. For example, a simple *for* loop may be returned if prompted to bring a number of blocks, while the optimal way is to bring all of them in one batch. Motivated by this, we look into how commonsense knowledge in LLMs can be used to rewrite the LLM-generated plans to achieve better efficiency.

Chapter 2

Literature Review

2.1 Imitation learning from observations

Imitation learning from observations extends imitation learning to the case that experts' action labels are no longer available. Edwards et al. [73] first learns a latent policy and then remaps outputs from latent policy to actual action space. In [150, 269], a context translation model is used to map across contexts considering context differences between demonstration and agent's observation.

Third-person imitation learning further extends the imitation learning from observations by replacing FPV expert demonstrations with TPV ones. Recent literature on third-person imitation learning focuses on how to make connections between TPV and FPV observations by learning visual representations. TPIL [230] extends GAIL [107] and uses a domain confusion constraint to force features from both FPV and TPV are indistinguishable for a discriminator. TCN [209] uses a time-contrastive way to learn representations by self-supervised metric learning. TCN [209] is also extended to other tasks such as skill transfer [163] and playing hard exploration games [15]. mfTCN [71] extends the time-contrastive method to multiple frames. In our work, we extend TPIL to an egocentric FPV setting.

2.2 Disentanglement

Disentanglement is learning independent attributes encoded into separated dimensions of representation space. Typical methods for disentanglement are based on variational auto-encoders (VAE) [130] and generative-adversarial networks (GAN) [86]. β-TCVAE[38] hypothesis that each dimension in representation z is

mutually independent. Both CVAE[226] and Info-GAN [43] provide the model with class label input to learn representations that are independent of these labels. Also, GAN methods like style-GAN [126] get a latent representation as a prior from a normal distribution. CC-VAE [174] adapts CVAE [226] to a self-supervised manner using an extra auto-encoder to learn a condition representation.

In imitation learning context, representations could also be implicitly disentangled. TRAIL [285] extends GAIL [107] to disentangle task-relevant and task-irrelevant information by adding a consistency constraint on samples in an invariant set. The learned representations are agnostic to task-irrelevant factors like colors. Disentangled representations have been also applied to other computer vision tasks and get successful results [63, 119, 184, 219, 278].

Compared with the above methods, our disentanglement method explicitly splits latent representation into two components: state and viewpoint representation. We do not assume that exact viewpoint coordinates (i.e. labels of viewpoints) are provided for disentanglement, and rather train the model in a self-supervised manner.

2.3 Contrastive learning

Contrastive learning performs a comparison across different views that are generated from one single input to learn representations in a self-supervised fashion[14, 66]. The concept of contrastive learning also has been adopted for TPIL [209]. For computer vision tasks, CMC[240] takes advantage of InfoNCE loss [14] on the comparison between anchor sample, positive sample, and a batch of negative samples. Other research on contrastive learning[40–42, 90, 102] focus on avoiding trivial solutions to improve representation quality and increasing training efficiency by getting rid of negative samples. Contrastive learning concepts and techniques are also applied to control tasks like [228]. We extend time-contrastive loss [209] (which is based on triplet loss [104]) using InfoNCE loss [14] that includes a batch of negative samples. We also take advantage of the stop-gradient technique from the above literature to avoid trivial solutions.

2.4 Viewpoint-agnostic Representation

There has been a remarkable progress in visual understanding with the shift from the use of CNNs [33, 101, 139, 235] to visual Transformers [68]. Transformers have shown substantial improvements over CNNs in image [34, 68, 97, 124, 152,

242, 273] analysis and video [11, 22, 153, 195, 202] understanding tasks due to its flexibility in learning global relations among visual tokens. Studies also combine CNNs with Transformer architectures to leverage the pros in both the structures [54, 89, 155, 215, 216]. In addition, Transformer has been shown to be effective in learning 3D representation [279]. However, these advancements in architecture-types have not addressed the issue of learning viewpoint-agnostic representation. Viewpoint-agnostic representation learning is drawing increasing attention in the vision community due to its wide range of downstream applications like 3D object-detection [198],video alignment [37, 72, 83], action recognition [221, 222], pose estimation [98, 233], robot learning [29, 109, 116, 209, 211, 230], and other tasks.

There is a broad line of work towards directly utilizing 3D information like depth [98], pose [58, 59], and point clouds [187, 197], or in some cases deriving 3D structure from paired 2D inputs [254]. However, methods rely on the availability of multi-modal data which is hard to acquire are not scalable. Consequently, other studies have focused on learning 3D perception of the input visual signal in order to generalize the learned representation to novel viewpoints. This is done by imposing explicit geometric transform operations in CNNs [30, 113, 186, 198, 265], without the requirement of any 3D supervision. In contrast to these existing works, our Transformer-based 3DTRL imposes geometric transformations on visual tokens to recover their representation in a 3D space. To the best of our knowledge, 3DTRL is the first of its kind to learn a 3D positional embedding associated with the visual tokens for viewpoint-agnostic representation learning in different image and video tasks.

2.5 Reinforcement Learning to Sequence Modeling

Reinforcement Learning (RL) is typically modeled as a Markov Decision Process (MDP), where actions are made solely based on the current state according to the Markov property. Accordingly, single-step value-estimation methods have been derived from the Bellman equation, including Q-learning [259] and Temporal Difference (TD) learning, along with many other variants such as SARSA [200], TD(λ) [234], TD-Gammon [238], and Actor-Critic [132]. In recent studies, neural networks have been used to approximate the value function in value-based methods, introducing Deep Q-learning [170].

More recent directions [35, 117] formulate RL as a sequence modeling task, where the model uses a sequence of recent experiences, including state-actions-

reward triplets, to predict future actions. This can potentially replace value-estimation methods, and can be trained in a supervised learning manner. However, this approach requires pre-existing trajectories (collected in advance), making it more compliant with offline RL and imitation learning settings. Zheng et al. [280] adapted sequence modeling formulation from offline-settings to online settings. Furuta et al. [81] extended DT [35] to a generalized version to match any given hindsight information.

2.6 Transformers

Transformer architectures [251] were first introduced in language processing tasks [64, 192, 193] to model interactions within a sequence of word embeddings, or, more generally, unit representations or *tokens*. More recently, Transformers have been employed in vision tasks with the key idea of breaking down images/videos into tokens [11, 36, 67, 176], often outperforming convolutional networks (CNNs) in practice. Transformers have also been successfully used to handle sensory information [236] and perform one-shot imitation learning [60]. Chen et al.[35] explored how GPT [193] can be applied to RL in a sequence modeling setting.

Sequence modeling in visual RL is similar to video-based learning in terms of the input data, which consists of observed image (i.e. state) sequences. One challenge in applying Transformers to videos is the large number of input tokens required to be processed. This problem has been investigated in multiple directions, including attention approximation [50, 131, 257], separable attention in different dimensions [11, 22], reducing the number of tokens using local windows [152, 154], and adaptively generating a small number of tokens [202].

StARformer presents a similar concept of using local windows for self-attention in Swin Transformer [152]. Our approach is also closely related to "divided space-time attention" in [22] and "factorized self-attention" in [11], in terms of handling spatial and temporal attention separately in mentioned literature. Because these methods are designed to reduce the number of tokens, our primary goal is to model short- and long-range contexts separately, thus ensuring higher performance on RL trajectories. Our Step-to-Sequence connection is inspired by [97], which was designed for image domain. However, our model is still unique as (a) "spatial" (using Step Transformer) and "temporal" (using Sequence Transformer) attention are performed in multiple Transformer layers, (b) separate sets of tokens with different origins are used, and (c) a set of locally-attended tokens is fed to perform self-attention in conjunction with a set of global tokens for long-term sequence

modeling. Furthermore, we find similar concepts for the use of both attention and convolutional features in [54].

2.7 Active Learning

Active Learning is the concept that an agent decides which data are taken into its learning and may ask for external information in comparison to fitting a fixed data distribution [3, 12, 20, 49, 51, 82, 122, 129, 151, 157, 161, 169, 173, 188, 208, 220, 224, 255, 264, 283]. **Active Vision** focuses on continuously acquiring new visual observations that is helpful for the vision task like object classification, recognition and detection [7, 8, 13, 17, 46, 76, 77, 127, 167, 248, 266, 274], segmentation [31, 158, 179], and action recognition [118]. The active vision is usually investigated under a robot vision scenario that a robot moves around in a scene. However, the policy is usually not required to accomplish a task with physical interactions such as manipulating objects compared to reinforcement learning.

2.8 Active Reinforcement Learning

Active Reinforcement Learning (Active-RL), at a high level, is that the agent is allowed to actively gather new perceptual information of interest simultaneously through an RL task, which can also be called active perception [260]. The extra perceptual information could be reward signal [6, 55, 74, 135, 156, 165], visual observations from new viewpoints [91, 149, 175, 217], other input modalities [39], and language instructions [47, 177, 178, 239]. Though these work may not explicitly use the term Active-RL, we find that they can be uniformly organized in the general Active-RL formulation and we coin the term here. In our work, we study the ActiveVision-RL task in a limited visual observability environment, where at each step the agent is only able to partially observe the environment. The agent should actively seek the optimal observation at each step. Therefore, our setting is more close to [87, 91] and Active Vision problems, unlike research incorporating attention-like inductive bias given a full observation [92, 114, 115, 204, 227, 237, 262]. The ActiveVision-RL agent must learn an observation selection policy, called sensory policy, to effectively choose the optimal partial observation for executing the task-specific policy (motor policy). The unique challenge for ActiveVision-RL is the coordination between sensory and motor policies given there mutual influence. In recent works, the sensory policy can be either trained in the task-agnostic way [91] with enormous exploration data,

or trained jointly with the task [87] with naive environmental reward only. In this work we investigate the joint learning case because of the high cost and availability concern of pre-training tasks [91].

2.9 Robot Learning with View Changes

Viewpoint changes and gaps in visual observations are the common challenges for robot learning [109, 145, 201, 214, 231, 281], especially for the embodied agents that uses its first-person view [78, 93, 206]. To address those challenges, previous works proposed to map visual observation from different viewpoints to a common representation space by contrastive encoding [71, 209, 212] or build implicit neural representations [145, 268]. In many first-person view tasks, the viewpoint control is usually modeled together with the motor action like manipulation and movement [78, 206]. In contrast, in our ActiveVision-RL setting, we explore the case where the agent can choose where to observe independently to the motor action inspired by the humans' ability.

Chapter 3

Self-Supervised Disentangled Representation Learning for Third-Person Imitation Learning

3.1 Overview

Humans learn to imitate by observing others. In robotics, imitation learning enables a robot to learn complex tasks with minimal environmental knowledge [112] based on expert demonstrations. The robot learns a mapping from states (observations) to actions by using the expert trajectories as training data. However, imitation learning is known to be costly in collecting such expert data. Such expert demonstrations should be in the first-person view (FPV) from the same (or very similar) viewpoint to the robot and should include actual action labels. Collecting a sufficient amount of robot data could potentially be very expensive, especially for action labels.

Third-person imitation learning (TPIL) is the concept of learning action policies by observing other agents in a third-person view (TPV) without accessing the action labels. TPIL allows utilizing many human or robot demonstration videos in TPV from different data sources and very different viewpoints. It is similar to that humans could map TPV observations to their egocentric perspective [164, 190] and learn from them. Recent advances in TPIL[163, 209, 230] solved tasks by learning a joint visual state representation space shared by FPV and TPV from synchronized FPV and TPV demonstrations. Such joint state representation can be used to guide downstream policy learning.

In this research, we study a TPIL case where the robot's FPV contains *egomotion*

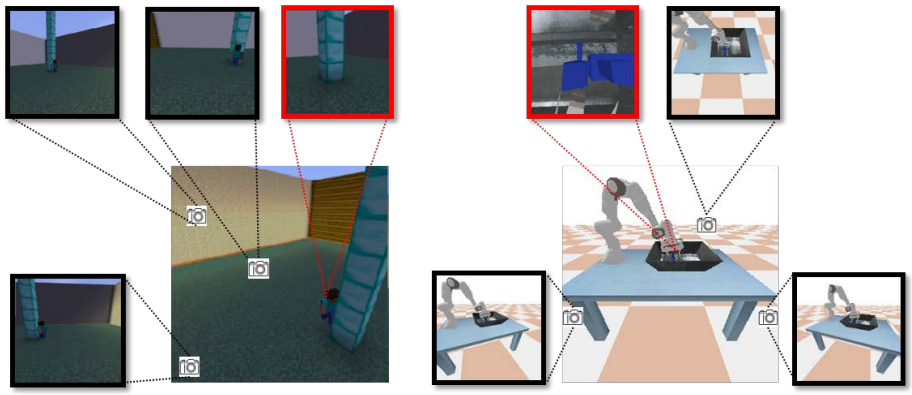


Figure 3.1: Examples of third-person imitation tasks using Minecraft **(a)** and Panda robot sim **(b)**. Red box: FPV that the agent actually observes, displaying agent's egomotion. Black box: TPVs that are taken from multiple fixed cameras. In Minecraft environment, the player in Minecraft is trying to walk to the diamond blocks (blue blocks). In Panda environment, the Panda robot is trying to pick up the object in the tray, based on the observations from a camera mounted on the end-effector.

caused by actions (Figure 3.1). For example, many robot tasks with ground (or aerial) mobility often involve actions with egomotion. Previous research [209, 230] only focused on FPV observations from a static camera, without considering tasks with the egomotion. Learning a joint visual state representation in such setting is very challenging due to the visual differences between egocentric FPV and TPV: (1) FPV videos consist of agent's egomotion while TPV videos consist of a relatively fixed scene with the moving agent, and (2) the agent itself is not visible (or only a very small portion of it is visible) in the FPV videos. The study on TPIL for egocentric videos will further broaden the scope of where TPIL algorithms could be applied.

To this end, we introduce a TPIL approach that learns disentangled state and viewpoint representations to ensure state representations are viewpoint-agnostic. Focusing on such visual differences between FPV and TPV, we propose a dual auto-encoder model to process the FPV and TPV inputs separately. In addition, we split the latent representation z of each auto-encoder into two parts, h and v, to encode disentangled state and viewpoint information, respectively. We propose representation permutation loss to train h and v to be well-disentangled representations. We adapt time-contrastive loss and apply it on h to encode state information. A general reconstruction loss is used to fully train our decoders.Our

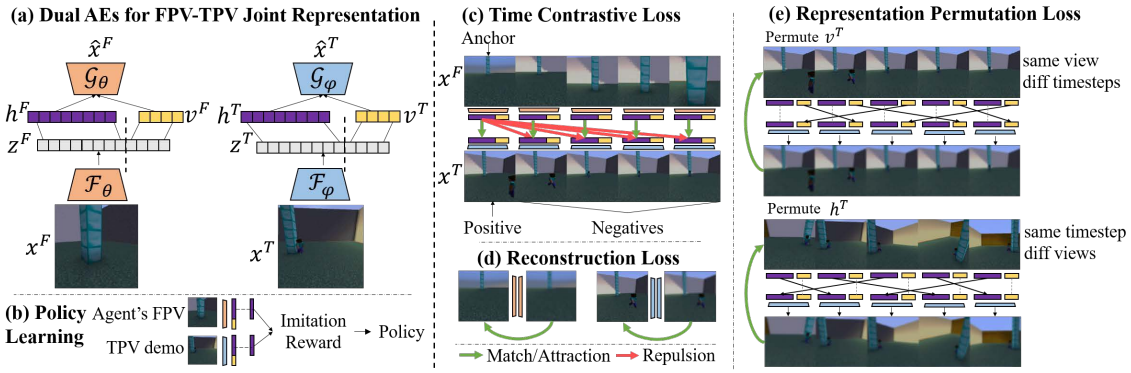


Figure 3.2: This figure includes an overview of the dual AE model we propose, the policy learning method we use after having a state representation space, and losses we use to learn our FPV-TPV joint state representation. (a) Proposed dual AE model. (b) When doing policy learning, we discard v and use h. (c) Time-contrastive loss requires an anchor sample, a positive sample, and a batch of negative samples. This figure uses anchor FPV frame as an example. Symmetrically, a TPV frame can be an anchor. (d) We reconstruct every input FPV/TPV frame and calculate the reconstruction loss. (e) We permute viewpoint representation v among a batch of frames from the same TPV but different timesteps. Symmetrically, we permute state representations h among a batch of frames from the same timestep but different TPVs. We compare the reconstructed frames with original inputs accordingly to have our representation permutation loss.

visual representation learning is based on synchronized FPV and TPV videos from multiple viewpoints. Once finished, our policy learning is done by providing a single TPV expert demonstration without any FPV experts.

3.2 Problem Setup

A discrete-time finite-horizon discounted Markov decision process (MDP) can be noted by $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, T)$, where $\mathcal{S}$ is a state set, $\mathcal{A}$ is an action set, $\mathcal{P}$ is a transition probability distribution, r is a reward function, and T is the horizon. For imitation learning, the agent is given trajectories $\tau = \{(s_t, a_t)\}$ which are states and actions generated from an unknown expert policy π_E and r is not traceable. The agent is required to learn a policy π_θ that recovers π_E.

In visual third-person imitation learning from observation, we consider an observer is learning by solely watching a demonstrator's demonstration from a

different viewpoint than the demonstrator's, and a_t is not traceable. Under this setting, the observer and the demonstrator have visual observations $o_t^T \in \mathcal{O}^T$, $\mathcal{O}^T \subseteq \mathbb{R}^{H\times W\times d}$ and $o_t^F \in \mathcal{O}^F$, $\mathcal{O}^F \subseteq \mathbb{R}^{H\times W\times d}$ respectively, where $d = 3$ if we consider a pure RGB image input. Usually, we call $\mathcal{O}^T$ third-person view (TPV) and $\mathcal{O}^F$ first-person view (FPV). A fact holds that at any moment t, o_t^T and o_t^F are different since they see from different views but o_t^T and o_t^F correspond to a same state s_t. Therefore, visual representation learning approaches explored by [209, 230] tried to solve this problem by finding a function $\mathcal{F} : \mathcal{O} \to \mathcal{H}$ that learns a latent state representation $h = \mathcal{F}(o)$ satisfying $\mathcal{F}(o_t^T) = \mathcal{F}(o_t^F)$ to estimate the true state space $\mathcal{S}$.

In this research, we investigate an egocentric FPV setting that actions cause agent's egomotion. Moreover, we consider TPV demonstrations from multiple different viewpoints. This is more challenging than learning the state embedding only applicable for one third-person viewpoint.

Formally, we have a set of synchronized FPV-TPV trajectories $\{\tau\}$ for visual representation learning, and a single TPV expert demonstration $\tau_E = \{o_t^T\}$ generated by some unknown expert policy π_E for policy learning. For each FPV-TPV trajectory, it has one FPV video and multiple TPV videos from multiple viewpoints: $\tau = (\{o_t^F\}, \{o_t^{T_1}\}, \ldots, \{o_t^{T_n}\})$. Given the egocentric FPV setting, o^F is visually different from o^T since o^F contains egomotion and o^T contains the agent's appearance which is not in o^F. So the challenge is how we design the map function $\mathcal{F}$, i.e. the representation model, that still preserves $\mathcal{F}(o_t^T) = \mathcal{F}(o_t^F)$. After getting such viewpoint-agnostic state representation h, we train a policy π to maximize imitation reward formulated by τ_E to best recover π_E. We follow the previous literature[209] to give a distance-metric based reward function, e.g. $r(t) = -\|h_t^F - h_t^T\|_2$, to perform policy learning, where h_t^F is current agent's FPV observation and h_t^T is given TPV expert demonstration at the same timestep.

3.3 FPV-TPV Joint State Representation Learning

Focusing on the visual differences between $\mathcal{O}^F$ and $\mathcal{O}^T$, we propose our dual auto-encoder (AE) model with explicitly disentangled latent representations that learns a joint representation space $\mathcal{H}$. Then, we introduce our modified time-contrastive loss[209] and representation permutation loss to train our proposed model in a self-supervised fashion.

3.3.1 Dual AE for Joint State Representation Learning

Considering the visual differences between $\mathcal{O}^F$ and $\mathcal{O}^T$, we design two different mapping functions, $\mathcal{F}_\theta : \mathcal{O}^F \to \mathcal{Z}$ and $\mathcal{F}_\varphi : \mathcal{O}^T \to \mathcal{Z}$, instead of learning a single universal function for both $\mathcal{O}^F$ and $\mathcal{O}^T$. We propose a dual AE structure to model such $\mathcal{F}_\theta$ and $\mathcal{F}_\varphi$ as encoders in AEs along with the decoders $\mathcal{G}_\theta$ and $\mathcal{G}_\varphi$. The decoder $\mathcal{G}_\theta$ takes the full representation $z^F = [h^F, v^F]$ as input and outputs a reconstructed FPV image $\hat{x^F} = \mathcal{G}_\theta([h^F, v^F])$. Similarly $\mathcal{G}_\varphi$ reconstructs TPV image $\hat{x^T} = \mathcal{G}_\varphi([h^T, v^T])$.

We further design the dual AE to disentangle the representation vectors to better learn the joint embedding, taking advantage of multiple training losses. Specifically, each of our AE splits full representation vector z into two vectors, $z = [h, v]$, where $[\cdot, \cdot]$ means concatenation (See Figure 3.2(a)). Our motivation is to disentangle state and viewpoint from observation. If well-disentangled, the state representation is viewpoint-invariant, i.e. a joint embedding for TPV and FPV. Representation decomposition has been an effective strategy to obtain independent components (representations) in multiple computer vision tasks such as object recognition [38, 105, 126, 184], and our motivation is to extend and take advantage of such concept for TPIL. In our formulation, h (state) and v (viewpoint) are two independent factors we want to split from raw RGB observations. Similar to [184], which decomposed class-irrelevant features from class-relevant features for object recognition, our formulation has only two factors to decompose. Thus, we separate the latent vector into two parts and ensure that they are disentangled by appropriate loss functions.

The main technical challenge is in learning $\mathcal{F}_\theta$ and $\mathcal{F}_\varphi$, while ensuring (1) h to be an FPV-TPV joint state representation and (2) h and v are well disentangled. In the below subsections, we describe how the time-contrastive loss and the proposed representation permutation loss enable this.

3.3.2 Self-Supervised Disentangled Representation Learning

Time-Contrastive Loss

Time-contrastive[209] was previously proposed to make the state representation follow the temporal order. The temporal order means a state representation h_t should be closed to h_{t+1} but far from h_{t+n} where $n > 1$ in most cases.In practice, we use a triplet that includes anchor (A), positive (P), and negative (N) samples to construct time-contrastive loss. This loss requires representations of A and P

samples are close (attraction), and representations of A and N samples are distant (repulsion).

Original time-contrastive loss[104, 209] is in a triplet-loss form. We extend it to include multiple negative samples for each single A and P sample using Info-NCE loss (See Figure 3.2(c)), inspired by recent advances in contrastive learning[90, 240]. Given two synchronized FPV-TPV demonstration videos, $\{(x_i^F, x_i^T)\}$, we first identify an A(anchor) at timestep a. Suppose x_a^F is the anchor (A) sample, then the x_a^T is the positive (P) sample. We then randomly sample a batch of negative (N) timesteps $\{n_i\}$ that appears ξ away. Then $\{x_{n_i}^T\}$ from a TPV video are N samples of x_a^F. After we get A-P-N samples, we have our modified time-contrastive loss term on FPV anchor:

$$\mathcal{L}_{tc}^F = -\log \frac{d(h_a^F, sg(h_a^T))}{d(h_a^F, sg(h_a^T)) + \sum d(h_a^F, sg(h_{n_i}^T))},$$

where h is state representation vector, $d(\cdot,\cdot)$ is a critic metric on h implemented based on cosine similarity between h vectors[240]:

$$d(h_1, h_2) = \exp\left(\frac{h_1 \cdot h_2}{\|h_1\| \cdot \|h_2\|} \cdot \tau\right),$$

where $\|\cdot\|$ is L2-norm, τ is a scaling coefficient[240], $sg(\cdot)$ is a stop-gradient trick[42, 90] to avoid trivial solutions. Symmetrically, if we let the TPV frame x_a^T be anchor, we have $\mathcal{L}_{tc}^T$ by making x_a^F as positive sample and $\{x_{n_i}^F\}$ negative samples.

To force h to be a joint representation space for FPV and TPV, we add a general matching loss (See Figure 3.2(c) in the middle) for all the $\{(h_i^F, h_i^T)\}$ pairs from the same timestep. We minimize their L2-distance pairwisely:

$$\mathcal{L}_{match} = \mathbb{E}\Big[\|h^T - sg(h^F)\|_2\Big],$$

where $sg(\cdot)$ is stop-gradient, viewing the state representations from FPV as references. Together, the total time-contrastive loss is:

$$\mathcal{L}_{tc} = \mathcal{L}_{tc}^F + \mathcal{L}_{tc}^T + \mathcal{L}_{match}.$$

Representation Permutation Loss

We introduce our representation permutation loss below to supervise h and v to be disentangled (See Figure 3.2(e)). Ideally, h and v should be independent, saying that modifying v only changes the viewpoint information, and modifying h only changes the state information. Therefore, we permute h and v among

a batch of samples and reconstruct images from permuted full representations $z = [h_i, v_j]$. We expect the reconstructed images should be similar accordingly after the permutation. We will describe how we permute and compare reconstructed images accordingly below.

Given a single TPV demonstration, we can get the representations from two arbitrary timesteps: $(h_i^T, v_i^T) = \mathcal{F}(x_i^T)$ and $(h_j^T, v_j^T) = \mathcal{F}(x_j^T)$. Ideally, we should expect their viewpoint information is the same, so we have $v_i^T = v_j^T$. Then, by exchanging v_i^T to v_j^T and doing reconstruction, we would expect that the reconstruction image should still remain same as input at timestep i: $\mathcal{G}([h_i^T, v_j^T]) = x_i^T$. Symmetrically, $\mathcal{G}([h_j^T, v_i^T]) = x_j^T$. We can do these exchanges in a batch of samples from a single TPV demonstration by randomly pairing v to h. Then our viewpoint permutation loss is formulated as an reconstruction error

$$\mathcal{L}_v = \mathbb{E}_i \Big[\|\mathcal{G}([h_i^T, v_k^T]) - x_i^T\|_2 \Big], k \neq i,$$

where i, k are time indices of the selected batch.

Symmetrically, we consider permutations on state representations h. Considering samples from the same timestep but different viewpoints, we expect identical h but different v. Similar as $\mathcal{L}_v$ above, we randomly pair h to different v and have this state permutation loss

$$\mathcal{L}_h = \mathbb{E}_j \Big[\|\mathcal{G}([h_i^{T_k}, v_i^{T_j}]) - x_i^{T_j}\|_2 \Big], k \neq j,$$

where k and j indicate different TPVs. Combine viewpoint and state permutation losses together, we have our representation permutation loss:

$$\mathcal{L}_{permute} = \mathcal{L}_v + \mathcal{L}_h.$$

We do not apply the representation permutation loss above to the latent representation v^F from FPV branch, because the viewpoints can continuously change and are highly correlated with agent's egomotion in its FPV.

Reconstruction Loss

We have this reconstruction loss to ensure that (a) h and v are meaningful latent representations instead of trivial solutions and (b) we have a reliable decoder $\mathcal{G}$ that can reconstruct images for computing representation permutation loss above. We compare a reconstructed image $\mathcal{G}(\mathcal{F}(x))$ with input x (See Figure 3.2(d)):

$$\mathcal{L}_{recon} = \mathbb{E}_x \Big[\sum (\mathcal{G}(\mathcal{F}(x)) - x)^2 \Big]. \tag{3.1}$$

We apply this loss to all the samples we pass through our model, including the samples used when calculating time-contrastive loss and representation permutation loss.

In conclusion, our overall loss function to learn an FPV-TPV joint representation is

$$\mathcal{L} = \alpha \mathcal{L}_{tc} + \beta \mathcal{L}_{permute} + \mathcal{L}_{recon}, \tag{3.2}$$

where α and β are hyper-parameters to control the weight of time-contrastive loss and invariant loss.

3.3.3 Implementation Details

We implement each of our two AEs by seven convolutional layers and seven deconvolutional layers. As for splitting representation, we can assign any dimension to v. Specifically, if the dimension of v is 0, it means we do not split our full latent representation, i.e. $h := z$ and thus $\mathcal{L}_{permute}$ is not applicable. We use TCN [209] as a baseline model that using one universal encoder for FPV and TPV inputs. It can be regarded as a 0-dimension-v model. Since we use h for policy learning, we control all models to have 16-dimension h for a fair comparison between models. We empirically set $\alpha = 1$, $\beta = 1$ in all our models except ablation studies on these hyper-parameters.

3.3.4 Imitation Learning Formulation

We follow the common setting of imitation learning in recent literature[15, 209]. Given synchronized FPV-TPV demonstrations, we first train a joint representation model (See Figure 3.2(a)). Then, given only one extra demonstration $\{x_i^E\}$, we let the agent execute the policy in the environment to maximize the imitation learning reward (See Figure 3.2(b)). The imitation learning reward at each step R_t is assigned based the cosine similarity

$$R_t = \begin{cases} 1, & \cos(h_t^F, h_t^E) \geq \xi \\ 0, & \cos(h_t^F, h_t^E) < \xi \end{cases}, \tag{3.3}$$

where $[h_t^F, v_t^F] = \mathcal{F}_\theta(x_t^F)$ are the latent representations of the agent's first-person observation and h_t^E is the state representation from expert demonstration, both are at current timestep t. h_t^E will be produced by $\mathcal{F}_\theta(x_t^E)$ if we perform a first-person imitation and by $\mathcal{F}_\varphi(x_t^E)$ if it is third-person imitation learning. We use a threshold ξ to discrete the imitation reward, following the formulation as [10, 15].

3.3.5 Policy Model and Policy Training

We use a multi-layer perceptron (MLP) as our policy model to map the state representation to agent actions: $\pi : \mathcal{H} \rightarrow \mathcal{A}$. The input layer and two hidden layers contain the same dimension as the input state representation. For continuous action space, the policy outputs means and standard deviations of Gaussian distributions. For discrete action space, the policy outputs log-likelihoods of actions. We use PPO [207] to optimize the policy based on the imitation learning reward above.

3.4 Experiments

3.4.1 Environments and Tasks

We develop and use two simulated environments to collect data, train, and evaluate methods (See Figure 3.1).

Minecraft Environment

We develop a Minecraft game environment based on Project Malmo [121] and MineRL [93]. The agent is a game player and the task is moving itself to the target position. The agent observation is an RGB visual input from the default FPV in the game without user interfaces such that $\mathcal{O} \subseteq R^{W \times H \times 3}$. The action space is discrete, where $|\mathcal{A}| = 6$, including moving forward, backward, left, and right, and turning (along with the camera) towards left and right. The goal is to reach the target labeled as a pillar of diamond blocks (blue ones).

Panda Environment

We develop a simulated continuous control environment by PyBullet [52]. The agent is a Panda robot mounted on a desk. The task is reaching an object in a tray in front of the robot and picking up the object. The agent observation is an RGB visual input, $\mathcal{O} \subseteq R^{W \times H \times 3}$. The agent has an action space $\mathcal{A} \subseteq [-0.5, 0.5]^{11}$ which is the force applied to 11 joints. The camera is mounted at the end effector, i.e. the "hand" of the Panda robot. The camera follows the motion of the end effector and emulates an egocentric FPV.

3.4.2 Dataset Collection

We collect synchronized FPV-TPV videos from both environments. For the Minecraft environment, we collect 8 different demonstration trajectories with randomized target diamond block locations. Each trajectory contains synchronized 1 FPV video and 8 TPV videos. Third-person viewpoints are not controlled to be the same, so the Minecraft environment is more challenging to get a well-aligned representation. For Panda environment, we collect 30 different demonstration trajectories with randomized initial locations of the target object. The expert policy is driven by an oracle using extra information and inverse kinetics. Each trajectory contains synchronized 1 FPV video and 9 TPV videos corresponding to 9 different viewpoints. All distinct trajectories share the 9 viewpoints. We keep the number of trajectories small to better emulate the realistic setting where the amount of training data is limited. We split 20% data for each dataset as test sets.

3.4.3 Quantitative Evaluations on Representation Space

We first investigate the quality of our joint representation model in terms of alignment error [209] between FPV-TPV sequences. Given a FPV frame x_i^F having time index i and its state representation h_i^F, we find its nearest TPV state representation neighbor h_j^T that has time index j. The alignment error is the mean absolute temporal distance of i and j regularized by video frame count L over all indices i:

$$alignment\_error = \mathbb{E}_i \left[\frac{|i-j|}{L} \right]. \tag{3.4}$$

The lower error means the better quality of imitation reward suggesting a better representation space [209].

As shown in Table 3.1, our model generally has a lower alignment error than the baseline, indicating that our dual AE model can deal with visually different FPV and TPV inputs. Specifically, Minecraft results show that our model can better align videos from various unseen viewpoints than the baseline. And from the comparison between non-zero dimensions of v and 0-d v, having a disentangled representation v yields better performance than not having it, suggesting that our disentangled representation helps learn state representations. We further investigate how the non-zero dimensions of viewpoint representation v affect the representation learning quality. We try 4, 8, and 12 dimensions of v. Table 3.1 tells that 8-d has the lowest alignment error compared to 4 and 12. 12-d has worse performance than 4-d (due to the overfitting), but we note that all the variants are

Table 3.1: Alignment error comparison between different representation models

Model	**Environment**	
	Minecraft	Panda
Multi-view TCN[209]	0.2725	0.2861
Ours 0-d v (i.e. without v and $\mathcal{L}_{permute}$)	0.2606	0.2036
Ours 4-d v	0.2398	0.1892
Ours 8-d v	**0.2329**	**0.1550**
Ours 12-d v	0.2571	0.1999

Table 3.2: Ablation studies about loss functions on Panda environment based on our model with 8-dimension v.

Model based on 8-d v	Alignment Error
$\mathcal{L}_{tc} + \mathcal{L}_{permute} + \mathcal{L}_{recon}$	**0.1550**
$\mathcal{L}_{tc} + \mathcal{L}_{vmatch} + \mathcal{L}_{recon}$	0.2518
$\mathcal{L}_{tc} + \mathcal{L}_{recon}$	0.1922
$\mathcal{L}_{tc} + \mathcal{L}_{permute}$	0.1748
$\mathcal{L}_{tc}$	0.1844
$5\mathcal{L}_{tc} + \mathcal{L}_{permute} + \mathcal{L}_{recon}$	0.1695
$\mathcal{L}_{tc} + 5\mathcal{L}_{permute} + \mathcal{L}_{recon}$	0.1616

better than ours 0-d v baseline. We confirm that having a non-zero-dimension v helps learn a joint representation, while we should choose a suitable dimension of v empirically to ensure the information in v is well encoded, based on how much information we intend v to encode. We also learn that increasing the dimension of v is not always beneficial to learn such disentangled representation, which is the overfitting problem.

We also do ablation studies about our loss functions on Panda environment (see Table 3.2). We first compare our $\mathcal{L}_{permute}$ with a vanilla loss function $\mathcal{L}_{vmatch}$ that could be applied to v. Recall that v should be similar for inputs from the same TPV and different for inputs from different TPVs. We define $\mathcal{L}_{vmatch} = \mathcal{L}_{vsim} + \mathcal{L}_{vdissim}$ where $\mathcal{L}_{vsim} = \cos(v_i, v_j)$ and $\mathcal{L}_{vdissim} = \max\{\cos(v_i, v_j), 0\}$ for similarities and dissimilarities between v respectively. Results show that using $\mathcal{L}_{permute}$ has a lower alignment error (0.1550) than $\mathcal{L}_{vmatch}$ (0.2518), indicating $\mathcal{L}_{permute}$ is reasonable and effective on supervising v. Using $\mathcal{L}_{vmatch}$ leads to a higher alignment error

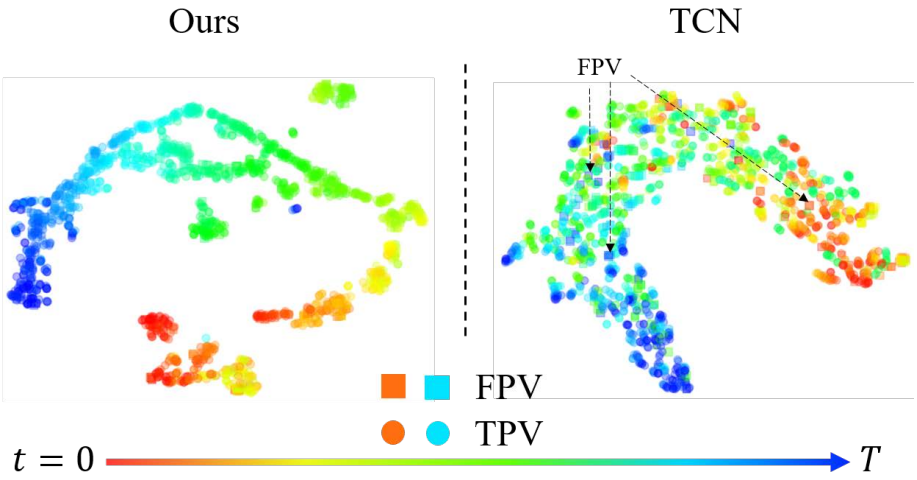


Figure 3.3: Visualization of state representation spaces $\mathcal{H}$ of Panda environment. Squares are FPV representations and dots are TPV representations. The colors from red to blue indicate the timesteps from 0 to T (end of a trajectory).

than not using $\mathcal{L}_{permute}$ (0.1922), suggesting that $\mathcal{L}_{vmatch}$ provides a poor supervise signal. We infer that the $\mathcal{L}_{vdissim}$ part overemphasizes the dissimilarity among v to satisfy $\cos(v_i, v_j) \leq 0$. We also investigate the performance if we remove certain loss functions. By removing $\mathcal{L}_{recon}$ and keep $\mathcal{L}_{permute}$ (0.1748), we know that adding a general reconstruction error would help the training of the decoder. But when we compare using $\mathcal{L}_{tc}$ and $\mathcal{L}_{recon}$ (0.1922) and using $\mathcal{L}_{tc}$ (0.1844), we find that using this $\mathcal{L}_{recon}$ without $\mathcal{L}_{permute}$ is not sufficient to supervise h and v. A well-trained decoder ensures that $\mathcal{L}_{permute}$ is not affected by the under-trained decoder. As for hyper-parameters α and β, increasing α does not yield a better result while the result of increasing β is comparable. We always keep $\mathcal{L}_{tc}$, otherwise the model will easily fail to capture temporal information between states.

3.4.4 Qualitative Evaluations on Representation Space

To evaluate the state representation space qualitatively, we first show a t-SNE visualization [249] of the learned representation space on Panda environment (Figure 3.3). We observe a generally clear temporal order from both methods. Moreover, FPV and TPV frames are aligned together by temporal order by our method, which indicates a good joint representation for FPV and TPV. However, in the visualization of TCN [209]'s representation space, we observe many FPV

Table 3.3: Success rate and cumulative rewards from (single example) imitation learning. *: first-person imitation learning. †: third-person imitation learning.

Model	**Environment**	
	Minecraft	Panda
	(Succ. Rate / Reward Mean ± Std)	
Random Policy	0 / 10.3±9.47	0 / 48.4±11.0
Single-view TCN[209] *	0.12 / 12.6±8.33	0.16 / 64.0±8.42
Ours*	0.46 / 19.8±9.34	0.36 / 69.3±8.45
Multi-view TCN[209]†	0.31 / 18.9±12.4	0.32 / 66.4±9.00
Ours†	**0.52** / **21.9±10.5**	**0.44** / **71.2±7.01**

representations are scattered in the space and are misaligned. This will affect the joint representation and generate a misleading reward function in third-person imitation learning. Therefore our model is more suitable to deal with the visual differences between $\mathcal{O}^F$ and $\mathcal{O}^T$.

We then show how our state representation h and viewpoint representation v encode corresponding information i.e. they are well disentangled. We follow the same permutation and reconstruction procedures as calculating $\mathcal{L}_{permute}$ in Section 3.3.2 on frames in the test set the model has never seen.

Figure 3.4 shows the result of replacing state representation h^{T_j} with h^F in Minecraft frames and replacing viewpoint representation v^{T_2} with v^{T_1} in Panda frames. From the result in Panda environment, we could observe the viewpoint in reconstructed images is coherent, which means changing h does not affect the viewpoint. The states are also well reconstructed if we see the poses of the robot. In the result of Minecraft environment, we could see the model reconstructs corresponding viewpoints by changing v. Even though the reconstructed agents are blurred, their positions are closed to the ground truth. We note that reconstruction is not our ultimate goal in this research, but it provides us an informative self-supervised objective to train disentangled latent representations. The clear disentanglement meets our initial motivation to separate h and v from full representation z.

3.4.5 Imitation Learning Evaluation

We finally evaluate the quality of our state representation in terms of success rates and rewards by applying it for the actual policy learning. We try both first-person

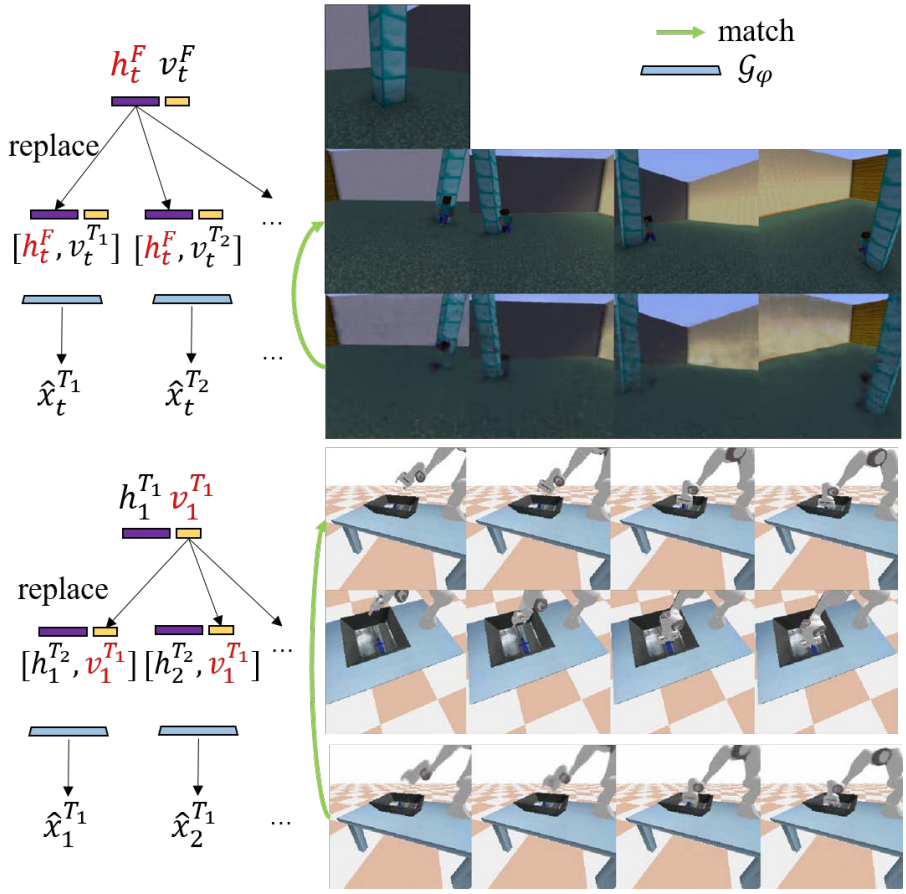


Figure 3.4: Reconstruction results of representation permutation on the test sets. The green arrow means the reconstructed images (from) should be similar with ground truth (to). Up: replacing h^{T_j} of TPV frames (2nd row) by h^F from an FPV frame (1st row) at the same timestep. Bottom: replacing v^{T_j} (of 2nd row) by v^{T_i} (of 1st row) from a different viewpoint.

imitation learning (FPIL) and third-person imitation learning (TPIL) to show that our joint state representation generalizes to both FPV and TPV domains.

Experiment Settings

All representation models are trained by expert trajectories in the training set. We have 6 distinct trajectories for Minecraft environment and 24 for Panda environment. We implemented two baselines for two different settings: FPIL and TPIL. For FPIL, we implement a single-view TCN [209] baseline where the representation is trained solely on FPV demonstrations. For TPIL, we implement a multi-view TCN [209] baseline where the representation is trained by FPV-TPV demonstrations. Our representation models are trained by FPV-TPV demonstrations as in TCN [209], and we apply them to both FPIL and TPIL because our representations are generalizable to both FPV and TPV inputs once learned. All the policies are trained by providing only one FPV expert demonstration for FPIL, or one expert TPV demonstration for TPIL, from the testing set.

We train each policy on Panda for 10^5 steps and on Minecraft for 5×10^4 steps by PPO [207]. We run 100 evaluation epochs for the best model achieved during training steps. Successful execution of a task is defined by reaching to the diamond block in Minecraft environment and picking and lifting the object in Panda environment. A continuous reward is defined by how much distance between the target and the agent has been minimized at any timestep in an epoch. We regularize the reward according to different randomized target locations to a fair comparison. The larger reward means a closer approaching to the target for both environments.

Experiment Results

Table 3.3 shows that our method outperforms the multi-view TCN [209] baseline in both FPIL and TPIL, suggesting our joint state representation space is better aligned in general and is capable of learning the policy in this egocentric TPIL setting.

In FPIL setting (2nd and 3rd rows in Table 3.3), our method outperforms the single-view TCN [209] baseline indicating that (1) the representation learned from only few FPV demonstrations is limited and (2) our method takes advantage of the FPV-TPV demonstrations to shape a better state representation space than using pure FPV demonstration. By introducing third-person demonstrations from multiple viewpoints, we could learn a better representation while keeping the total number of distinct trajectories is small.

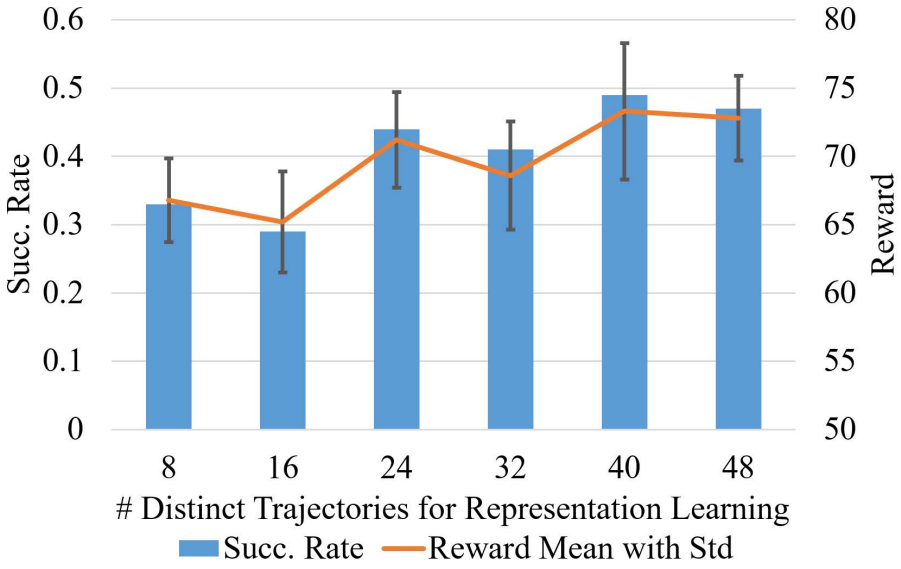


Figure 3.5: Performance under different number of distinct trajectories used in representation learning. The experiment is taken in Panda environment

We also show how the number of trajectories used for representation model training will affect the final policy. This experiment is done using Panda environment and results are shown in Figure 3.5. Our method benefits from more trajectories when the trajectory count is small and the performance saturates after providing 40 trajectories. Even with 8 and 16 FPV-TPV trajectories (0.33 and 0.29 Succ. Rate), our method can reach a better performance than Single-view TCN [209] (0.16), which is trained by 24 FPV trajectories. This highlights the value of our approach, as it uses fewer trajectories to gain better performance.

3.5 Conclusion

In this research, we consider a third-person imitation learning setting where the agent performs tasks that actions may cause agents' egomotion. We solve this egocentric third-person imitation learning by learning a joint state representation space for FPV and TPV inputs. We introduce a dual AE model to encode FPV and TPV inputs separately considering the visual differences between FPV and TPV videos. We explicitly split latent representation into state and viewpoint representations and train them to be disentangled by applying time-contrastive, representation permutation, and reconstruction losses in a self-supervised way. Re-

sults show that our representation space successfully encodes the state information with viewpoint information disentangled. We apply our joint state representation to both third-person and first-person imitation learning, and results show that our state representation is effective for learning a task out of either TPV or FPV expert demonstrations.

Chapter 4

Learning Viewpoint-Agnostic Visual Representations by Recovering Tokens in 3D Space

4.1 Overview

Over the past few years, computer vision models have developed rapidly from CNNs [33, 101, 235] to now Transformers [68, 97, 242]. With these models, we can now accurately classify objects in an image, align image frames among video pairs, classify actions in videos, and more. Despite their successes, many of the models neglect that the world is in 3D and do not extend beyond the XY image plane [84]. While humans can readily estimate the 3D structure of a scene from 2D pixels of an image, most of the existing vision models with 2D images do not take the 3D structure of the world into consideration. This is one of the reasons why humans are able to recognize objects in images and actions in videos regardless of their viewpoint, but the vision models often fail to generalize over novel viewpoints [56, 84, 186].

Consequently, in this paper, we develop an approach to learn viewpoint-agnostic representations for a robust understanding of the visual data. Naive solutions to obtain viewpoint-agnostic representation would be either supervising the model with densely annotated 3D data, or learning representation from a large scale 2D datasets with samples encompassing different viewpoints. Given the fact that such high quality data are expensive to acquire and hard to scale, an approach with a higher sample efficiency without 3D supervision is desired.

To this end, we propose a 3D Token Representation Layer (3DTRL), incorpo-

rating 3D camera transformations into the recent successful visual Transformers [34, 68, 152, 242]. 3DTRL first recovers camera-centered 3D coordinates of each token by depth estimation. Then 3DTRL estimates a camera matrix to transform these camera-centered coordinates to a 3D world space. In this world space, 3D locations of the tokens are absolute and view-invariant, which contain important information for learning viewpoint-agnostic representations. Therefore, 3DTRL incorporates such 3D positional information in the Transformer backbone in the form of 3D positional embeddings, and generates output tokens with the 3D information. Unlike visual Transformers only relying on 2D positional embedding, models with 3DTRL are more compliant with learning viewpoint-agnostic representations.

We conduct extensive experiments on various vision tasks to confirm the effectiveness of 3DTRL. Our 3DTRL outperforms the Transformer backbones on 3 image classification, 5 multi-view video alignment, and 2 multi-view action recognition datasets in their respective tasks. Moreover, 3DTRL is a light-weighted, plug-and-play module that achieves the above improvements with minimal (2% computation and 4% parameters) overhead. In summary, we present a learnable, differentiable layer 3DTRL that efficiently and effectively learns viewpoint-agnostic representations.

4.2 Background: Pinhole Camera Model

3DTRL is based on the standard pinhole camera model widely used in computer vision. Thus, we first briefly review the pinhole camera model. In homogeneous coordinate system, given a point with world coordinate p^{world}, a camera projects a pixel at p on an image by:

$$p = \mathbf{K} \left[\mathbf{R}|\mathbf{t} \right] p^{\text{world}}, \tag{4.1}$$

where $\mathbf{K}$ is the intrinsic matrix and $[\ \mathbf{R}|\mathbf{t}\ ]$ is the extrinsic matrix. $\mathbf{K}$ is further represented by

$$\mathbf{K} = \begin{bmatrix} c & 0 & u_0 \\ 0 & c & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.2}$$

where c is the focal length and (u_0, v_0) are offset. In this work, we explore visual understanding in multi-view setting, thus aiming at learning viewpoint-agnostic

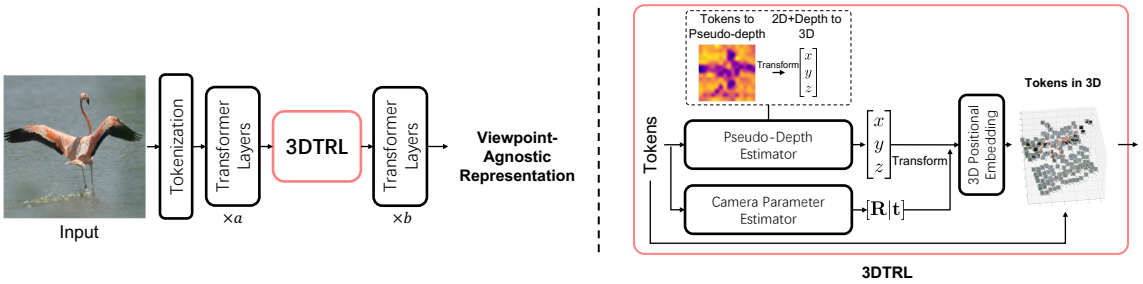


Figure 4.1: Overview of proposed 3DTRL. **Left**: 3DTRL is a module inserted in between Transformer layers. **Right**: 3DTRL has three parts, Pseudo-Depth Estimator, Camera Parameter Estimator, and 3D Positional Embedding layer. Within Pseudo-Depth Estimator, we first estimate depth of each token and then calculate 3D coordinates from 2D locations and depth.

representations. In this setting, a scene may be captured with different cameras positioned at non-identical locations and viewing angles. Here, the world coordinate p^{world} is the same across all the cameras while pixel projection p is different across cameras. We focus on how to estimate the world coordinates p^{world} from their corresponding image pixels at p. Estimating p^{world} from p involves two transformations that correspond to the *inverse* of $\mathbf{K}$ and $[\,\mathbf{R}|\mathbf{t}\,]$ which might not be known beforehand. Thus, we learn to *estimate* them from image patches instead, which is a key procedure in 3DTRL.

4.3 3D Token Representation Layer (3DTRL)

In this section, we detail how 3DTRL estimates 3D positional information in a Transformer and is integrated. We will introduce 3DTRL for image analysis, then we adapt it to video models.

4.3.1 Method overview

3DTRL is a simple yet effective plug-in module that can be inserted in between the layers of visual Transformers (Figure 4.1 left). Given a set of input tokens S, 3DTRL returns a set of tokens with 3D information. The number of tokens and their dimensionality are kept unchanged. Within the module (Figure 4.1 right), 3DTRL first performs 3D estimation using two estimators: (1) pseudo-depth estimator and (2) camera parameter estimator. Then the tokens are associated

with their recovered world 3D coordinates, which are transformed from estimated depth and camera matrix. Finally, 3DTRL generate 3D positional embeddings from these world coordinates and combine them with input S to generate the output of 3DTRL.

In the aspect that we insert 3DTRL in between the Transformer model, 3DTRL implicitly leverages the Transformer layers *before* it to be a part of the 3D estimators, and layers *after* that to be the actual 3D feature encoder (Figure 4.1 left). This avoids adding a large number of parameters while resulting in reasonable estimations. We empirically find that placing 3DTRL at a shallow-medium layer of the network yields better results (see Section 4.4.6), which is a trade-off between model capacity for estimation and 3D feature encoding.

4.3.2 3D Estimation of the Input Tokens

3DTRL first estimates both the camera-centered 3D coordinates of each token using depth estimation and a camera matrix shared by all the tokens of an image. Then, the camera-centered 3D coordinates are transformed to the "world" coordinates by the camera matrix.

Pseudo-depth Estimation. Given input tokens $S = \{s_1, \ldots, s_N\} \in \mathbb{R}^{N \times m}$, 3DTRL first performs pseudo-depth estimation of each token s_n. The pseudo-depth estimator is a function $f : \mathbb{R}^m \to \mathbb{R}$ that outputs the depth $d_n = f(s_n)$ of each token individually. We implement f using a 2-layer MLP. We call this pseudo-depth estimation since it is similar to depth estimation from monocular images but operates at a very coarse scale, given that each token corresponds to an image patch rather than a single pixel in Transformer.

After pseudo-depth estimation, 3DTRL transforms the pseudo-depth map to camera-centered 3D coordinates. Recall that in ViT [68], an image X is decomposed into N patches $\{X_1, \ldots, X_N\} \in \mathbb{R}^{N \times P \times P \times 3}$, where $P \times P$ is the size of each image patch and the tokens S are obtained from a linear projection of these image patches. Thus, each token is initially associated with a 2D location on the image plane, denoted as (u, v). By depth estimation, 3DTRL associates each token with one more value d. Based on the pinhole camera model explained in Section 4.2, 3DTRL transforms u, v, d to a camera-centered 3D coordinate (x, y, z) by:

$$p_n^{\text{cam}} = \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = \begin{bmatrix} u_n z_n / c \\ v_n z_n / c \\ z_n \end{bmatrix}, \text{ where } z_n = d_n. \tag{4.3}$$

Since we purely perform the aforementioned estimation from monocular images and the camera intrinsic matrix is unknown, we simply set c to a constant hyperparameter. To define coordinate system of 2D image plane (u, v), we set the center of the original image is the origin $(0, 0)$ for convenience, so that the image plane and camera coordinate system shares the same origin. We use the center of the image patch to represent its associate (u, v) coordinates.

We believe that this depth-based 3D coordinate estimation best leverages the known 2D geometry, which is beneficial for later representation learning. We later confirm this in our ablation study (in Section 4.4.6), where we compare it against a variant of directly estimating (x, y, z) instead of depth.

Camera Parameter Estimation. The camera parameters are required to transform the estimated camera-centered 3D coordinates p^{cam} to the world coordinate system. These camera parameters are estimated jointly from all input tokens S. This involves estimation of two matrices, a 3×3 rotation matrix $\mathbf{R}$ and a 3×1 and translation matrix $\mathbf{t}$ through an estimator g. We implement g using a MLP. Specifically, we use a shared MLP stem to aggregate all the tokens into an intermediate representation. Then, we use two separated fully connected heads to estimate the parameters in $\mathbf{R}$ and $\mathbf{t}$ respectively. We note the camera parameter estimator as a whole: $[\ \mathbf{R}|\mathbf{t}\ ] = g(S)$. To ensure $\mathbf{R}$ is mathematically valid, we first estimate the three values corresponding to yaw, pitch and roll angles of the camera pose, and then convert them into a 3×3 rotation matrix. Research has shown that the discontinuity occurs at the boundary cases in rotation representation [263, 282], however, such corner cases are rare.

In case of generic visual understanding tasks like object classifications, we expect the camera parameter estimation to perform an "object-centric canonicalization" of images with respect to the "common" poses of the class object. This is qualitatively shown by Figure 4.9.

Transform to World Coordinates. Now, with the camera parameters, 3DTRL transforms estimated camera-centered coordinates p^{cam} into the world space, a 3D space where 3D coordinates of the tokens are absolute and viewpoint-invariant. Following the pinhole camera model, we recover p^{world}:

$$p_n^{\text{world}} = [\ \mathbf{R}^T|\mathbf{R}^T\mathbf{t}\ ]\ p_n^{\text{cam}}. \tag{4.4}$$

4.3.3 Incorporating 3D Positional Information in Transformers

The last step of 3DTRL is to leverage the estimated 3D positional information in Transformer backbone. For this, we choose to adopt a typical technique of incorporating positional embedding that is already used in Transformers [68, 252]. In contrast to 2D positional embedding in ViTs [68], 3DTRL learns a 3D embedding function $h : \mathbb{R}^3 \rightarrow \mathbb{R}^m$ to transform estimated world coordinates p^{world} to positional embeddings p^{3D}. This 3D embedding function h is implemented using a two-layer MLP. Then, the obtained 3D positional embedding is incorporated in the Transformer backbone by combining it with the token representations. The outcome is the final token representations $\{s^{\text{3D}}\}$:

$$s_n^{\text{3D}} = s_n + p_n^{\text{3D}}, \text{ where } p^{\text{3D}} = h(p^{\text{world}}). \tag{4.5}$$

After 3D embedding, the resultant token representations are associated with a 3D space, thus enabling the remaining Transformer layers to encode viewpoint-agnostic token representations. We ablate other ways of incorporating the 3D positional information of the tokens in Section 4.4.6.

4.3.4 3DTRL in Video Models

Notably, 3DTRL can be also easily generalized to video models. For video models, the input to 3DTRL is a set of spatial-temporal tokens $\{S_1, \ldots, S_T\}$ corresponding to a video clip containing T frames, where $S_t = \{s_{t1}, \ldots, s_{tN}\}$ are N tokens from t-th frame. We simply extend our module to operate on an additional time dimension, where depth estimation and 3D positional embedding are done for each spatial-temporal tokens s_{tn} individually: $d_{tn} = f(s_{tn})$, $p_{tn}^{\text{3D}} = h(p_{tn}^{\text{world}})$. Camera parameters are estimated per input frame (S_t) in a dissociated manner, namely $[\ \mathbf{R}|\mathbf{t}\ ]_t = g(S_t)$, resulting in a total of T camera matrices per video. We investigate another strategy of camera estimation in the supplementary, where only one camera matrix is learned for all frames. However, our studies have substantiated the effectiveness of learning dissociated camera matrices per frame.

4.4 Experiments

We conduct extensive experiments to demonstrate the efficacy of viewpoint-agnostic representations learned by 3DTRL in multiple vision tasks: (i) image classification, (ii) multi-view video alignment, and (iii) video action classification. We also qualitatively evaluate the pseudo-depth and camera estimation to confirm 3DTRL works.

4.4.1 Image Classification.

In order to validate the power of 3DTRL, we first evaluate ViT [68] with 3DTRL for image classification task using CIFAR-10 [134], CIFAR-100 [134] and ImageNet-1K [62] datasets.

Table 4.1: Top 1 classification accuracy (%) on CIFAR-10 and 100, ImageNet-1K (IN-1K), viewpoint-perturbed IN-1K (IN-1K-*p*), and ObjectNet. We also report the number of parameters (#params) and computation (in MACs). Note that the MACs are reported w.r.t. IN-1K samples.

Method	#params	MACs	CIFAR-10	CIFAR-100	IN-1K	IN-1K-*p*	ObjectNet
DeiT-T	5.72M	1.08G	74.1	51.3	73.4	61.3	21.3
+3DTRL	5.95M	1.10G	**78.8** (+4.7)	**53.7** (+2.4)	**73.6** (+0.2)	**64.6** (+3.3)	**22.4** (+1.1)
DeiT-S	22.1M	4.24G	77.2	54.6	79.4	71.1	25.8
+3DTRL	23.0M	4.33G	**80.7** (+3.5)	**61.5** (+6.9)	**79.7** (+0.3)	**72.7** (+1.6)	**27.1** (+1.3)
DeiT-B	86.6M	16.7G	76.6	51.9	81.0	70.6	27.0
+3DTRL	90.1M	17.2G	**82.8** (+6.2)	**61.8** (+9.9)	**81.2** (+0.2)	**74.7** (+4.1)	**27.3** (+0.3)

Figure 4.2: **Top**: original IN-1K samples. **Bottom**: viewpoint-perturbed IN-1K samples.

Training. We use the training recipe of DeiT [242] for training our baseline Vision Transformer model on CIFAR and ImageNet datasets from *scratch*. We performed ablations to find an optimal location in ViTs where 3DTRL should be plugged-in (Section 4.4.6). Thus, in all our experiments we place 3DTRL after 4 Transformer layers, unless otherwise stated. The configuration of our DeiT-T, DeiT-S, and DeiT-B is identical to that mentioned in [242]. All our transformer models are trained for 50 and 300 epochs for CIFAR and ImageNet respectively. Further training details and hyper-parameters can be found in the supplementary material.

Results. From Table 4.1, 3DTRL with all DeiT variants shows consistent performance improvement over the baseline on the CIFAR datasets, with only ∼2% computation overhead and ∼4% more parameters. Despite fewer training samples, 3DTRL significantly outperforms the baseline in CIFAR, showing the strong generalizability of 3DTRL when limited training data are available. We argue that multi-view data is not available in abundance, especially in domains with limited data (like in medical domain), thus 3DTRL with its ability to learn viewpoint-agnostic representations will be crucial in such domains. We also find the performance improvement on ImageNet is less than that on CIFAR. This is because ImageNet has limited viewpoints in both training and validation splits, thus reducing the

significance of performing geometric aware transformations for learning view agnostic representations. In contrast, CIFAR samples present more diverse camera viewpoints, so it is a more suitable dataset for testing the quality of learned viewpoint-agnostic representations.

Robustness on Data with Viewpoint Changes. In order to emphasize the need of learning viewpoint-agnostic representations, we further test the models trained on IN-1K on two test datasets: ObjectNet [18] and ImageNet-1K-perturbed (IN-1K-*p*). ObjectNet [18] is a *test* set designed to introduce more rotation, viewpoint, and background variances in samples compared to ImageNet. Overall, ObjectNet is a very challenging dataset considering large variance in real-world distributions. IN-1K-*p* is a *viewpoint*-perturbed IN-1K validation set by applying random perspective transformations to images, constructed by our own. Example images are shown in Figure 4.2. We note that perspective transformation on these static images is not equivalent to real viewpoint changes. Nonetheless, it is a meaningful for a proof-of-concept experiment. Results in Table 4.1 show that models with 3DTRL consistently outperform their corresponding Transformer baselines in these two test sets, suggesting 3DTRL is trained to generalize across viewpoint changes.

4.4.2 Multi-view Video Alignment

Video alignment [72, 96, 168] is a task to learn a frame-to-frame mapping between video pairs with close semantic embeddings. In particular, we consider a *multi-view* setting that aligns videos captured from the same event but different viewpoints, which could further facilitate robot imitation learning from third-person views [29, 209, 211, 230]. Here, video pairs from the same event are temporally synchronized.

Datasets We use 5 multi-view datasets from a wide range of environments: **Minecraft** (MC) – video game, **Pick**, **Can**, and **Lift** from robot simulators (PyBullet [52] and Robomimic [159]), and **Pouring** from real-world human actions [209]. Example video frames are provided in Figure 4.3. Each dataset contains synchronized videos from multiple cameras (viewpoints). There is one ego-centric camera per dataset except Pouring, which are continuously moving with the subject. These ego-centric videos make the alignment more challenging. Detail dataset statistics are available in supplementary.

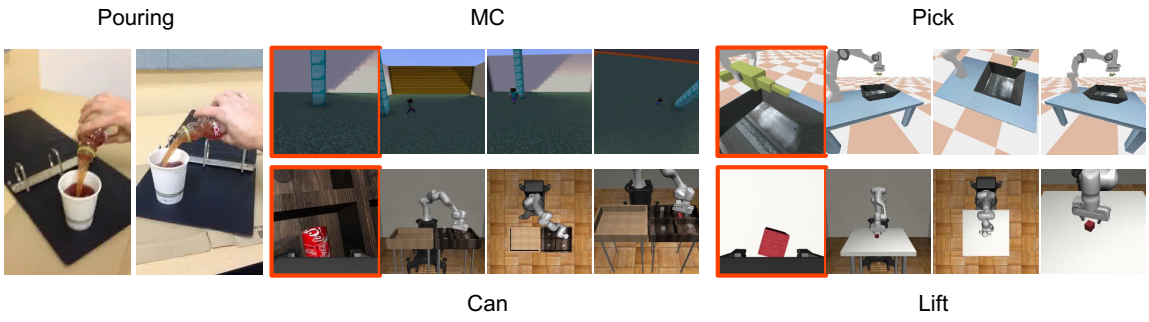


Figure 4.3: Examples video alignment datasets. Each dataset has synchronized videos of at least 2 viewpoints. All datasets except Pouring have one ego-centric view, highlighted in red boxes.

Training We follow common video alignment methods [209] to train an encoder that outputs frame-wise embeddings. We still use DeiT [242] as a baseline model (DeiT+TCN) and apply 3DTRL to it (+3DTRL), similar to image classification. During training, we use the time-contrastive loss [209] to encourage temporally closed embeddings to be similar while temporally far-away embeddings to be apart. Then, we obtain alignments via nearest-neighbor such that an embedding u_i from video 1 is being paired to its nearest neighbor v_j in video 2 in the embedding space. And similarly u_j from video 2 is paired with its nearest neighbor v_k in video 1. We use ImageNet-1K pre-trained weights for experiments on Pouring, but we train from scratch for other datasets considering that simulation environments are out of real-world distribution.

Evaluation We evaluate the alignment by three metrics: Alignment Error [209], Cycle Error, and Kendall's Tau [128]. Let the alignment pairs between two videos be (u_i, v_j) and (v_j, u_k). In brief, Alignment Error measures the temporal mismatching $|i-j|$ of (u_i, v_j). Cycle Error is based on cycle-consistency [72, 258], where two pairs (u_i, v_j) and (v_j, u_k) are called *consistent* when $i = k$. Thus, Cycle Error measures the inconsistency based on distance metric $|i-k|$. Kendall's Tau (τ) measures ordering in pairs. Given a pair of embeddings from video 1 (u_i, u_j) and their corresponding nearest neighbors from video 2 (v_p, v_q), the indices tuple (i, j, p, q) is *concordant* when $i < j$ and $p < q$ or $i > j$ and $p > q$. Otherwise the tuple is *discordant*. Kendall's Tau computes the ratio of concordant pairs and discordant pairs over all pairs of frames. Let N be the number of frames in a video, then the formal notations of the

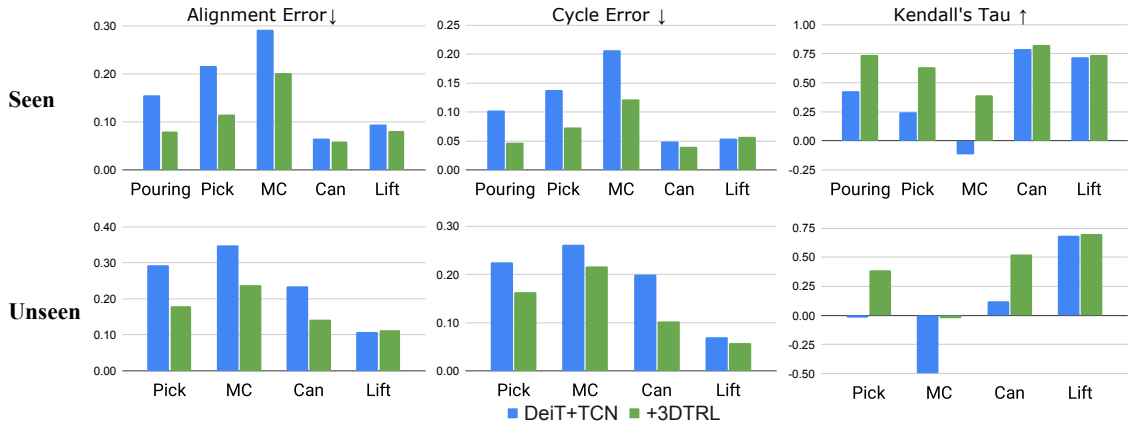


Figure 4.4: Results on video alignment in **Seen** and **Unseen** protocols. ↑ indicates a higher metric is better and ↓ is otherwise. Blue bars are for DeiT+TCN without 3DTRL and green bars are with 3DTRL. 3DTRL outperforms the baseline consistently in both settings. We note **Unseen** is not applicable to Pouring dataset because only two cameras are available in this dataset.

three metrics are:

$$\text{Alignment Error} = \mathbb{E}_i \frac{|i-j|}{N};\ \text{Cycle Error} = \mathbb{E}_i \frac{|i-k|}{N};\ \tau = \frac{\#\text{ concordant pairs} - \#\text{ discordant pairs}}{N(N-1)/2}. \tag{4.6}$$

We establish two evaluation protocols: (a) **Seen** and (b) **Unseen**. In **Seen**, we train and test models on videos from all cameras. However, in **Unseen**, we hold out several cameras for test, which is a representative scenario for validating the effectiveness of 3DTRL. Detail of experimental settings are provided in supplementary.

Results Figure 4.4 illustrates the evaluation results of 2 viewpoint settings over 5 datasets, compared to the DeiT baseline. 3DTRL outperforms the baseline consistently across all datasets. In particular, 3DTRL improves the baseline by a large margin in Pouring and MC, corroborating that 3DTRL adapts to diverse unseen viewpoints. The improvements of 3DTRL on Pick and MC also suggests the strong generalizability when learning from smaller datasets. With enough data (Lift & Can), 3DTRL still outperforms but the gap is small. When evaluating in **Unseen** setting, both methods have performance drop. However, 3DTRL still outperforms in Pick, MC, and Can, which suggests the representations learned by 3DTRL are able to generalize over novel viewpoints. MC has the largest viewpoint diversity so it is hard to obtain reasonable align results in the unseen setting for both the models.

Table 4.2: Video alignment results compared with SOTA methods. Values are alignment errors.

Method	Backbone	Input	Pouring	Pick	MC
TCN [209]	CNN	1 frame	0.180	0.273	0.286
Disentanglement [211]	CNN	1 frame	-	0.155	0.233
mfTCN [71]	3DCNN	8 frames	0.143	-	-
mfTCN [71]	3DCNN	32 frames	0.088	-	-
DeiT [242]+TCN	Transformer	1 frame	0.155	0.216	0.292
+3DTRL	Transformer	1 frame	**0.080**	**0.116**	**0.202**

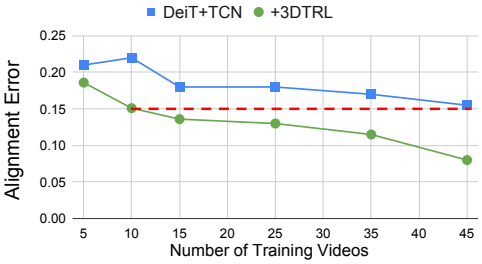


Figure 4.5: Alignment error w.r.t # training videos. Red dashed line indicates 3DTRL using 10 videos outperforms DeiT using 45 videos.

In Table 4.2, we further compare 3DTRL with previous methods on certain datasets. Note that Disentanglement [211] uses extra losses whereas others use time-contrastive loss only. We find that 3DTRL with only *single-frame* input is able to surpass the strong baselines set by models using extra losses [211] or multiple input frames [71]. We also vary the number of training videos in Pouring dataset, and results from Figure 4.5 show that 3DTRL benefits from more data. Meanwhile, 3DTRL can outperform the baseline while only using 22% of data the baseline used.

4.4.3 Quantitative Evaluations on Recovering 3D information

We perform quantitative evaluations on how well 3DTRL recovers 3D information. We emphasize once more that the 3D recovery in our approach is done without any supervision; it is optimized with respected to the final loss (e.g., object classification), without any access to the ground truth 3D information. We first discuss 3D estimations focusing on pseudo-depth estimation (Section 4.4.3), then we evaluate camera estimation (Section 4.4.3).

3D Estimation Evaluation

The key component of our 3D estimation is pseudo-depth estimation. In order to evaluate the 3D estimation capability, we compare the pseudo-depth map with ground truth depth map, using NeRF [166] dataset. We test the pseudo-depth with DeiT-T+3DTRL trained on IN-1K.

Metric: Depth Correlation. Since our estimated pseudo-depth (d') and ground truth (d) are in different scales, we measure their relative correspondence, i.e., correlation of two sets of data. We use Pearson's r: $r = \text{correlation}(d, d')$, where we regard two depth maps as two groups of data. Note that 3DTRL operates on visual Transformers, so the pseudo-depth map is at a very coarse scale (14×14). We also resize the given depth map to 14×14 to perform the evaluation. We report the average of r across all evaluation subsets.[1].

Results. Figure 4.6 shows the evaluation results. We find the estimated pseudo-depth highly correlates with the ground truth ($r \approx 0.7$). Compared to the random prediction baseline, the depth from 3DTRL shows much higher correlation to the ground truth. We also find that the model learns the estimated depth with higher correlation in the earlier stage of the training (e.g., 20 epochs) and then drops a bit in the later training epochs ($\sim$ 0.07 drop).

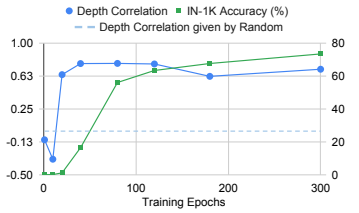


Figure 4.6: Depth correlation evaluation results. We find our estimation has a high (~0.7) correlation to the ground truth.

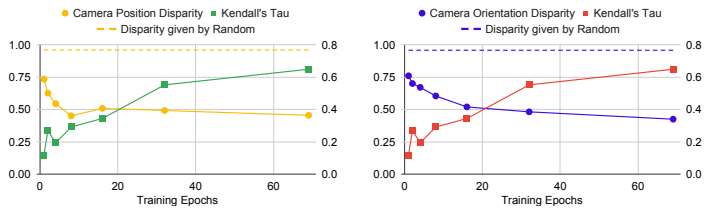


Figure 4.7: Quantitative evaluations on camera position (left) and camera orientation (right) respectively. Overall, we show the estimated cameras from 3DTRL have an acceptable mapping to the ground truth (both give < 0.5 disparity).

Camera Estimation Evaluation

In this evaluation, we mainly answer how well 3DTRL estimates camera position and orientation. To do this, we evaluate DeiT+3DTRL trained in previous video-alignment task (Section 4.4.2), using **Can** dataset. Recall that we train the model from scratch for the video alignment, without access to ground truth 3D information. We use first-person view videos for our evaluation — the camera moves

[1]To do so, we convert r to Fisher's z, take the mean value across all subsets, and convert back to Pearson's r.

together with the robot and our objective is to estimate its pose. We introduce two metrics.

Metric1: Camera Position Disparity: For each video, we get the estimated camera positions $\{\mathbf{p}'\}$ and ground truth camera positions $\{\mathbf{p}\}$, which are 3-D vectors. Since estimated and ground truth cameras are in two different coordinate systems, we need to measure how estimated camera positions map to the ground truth in a scale, translation and rotation-invariant way. The existing metrics like AUROC [120, 205] are not applicable. Therefore, we use Procrustes analysis [88] and report the disparity metric. The disparity value ranges $[0, 1]$, where a lower value indicates that two sets are more similar. We take the average disparity of all videos.

Metric2: Camera Orientation Disparity: Each camera has its orientation, i.e. "looking-at" direction, described by a 3-D vector. We note the estimated camera orientations $\{\mathbf{o}'\}$ and ground truth $\{\mathbf{o}\}$. Similar as the camera position disparity mentioned above, we use the disparity given by Procrustes analysis [88] to measure how $\{\mathbf{o}'\}$ and $\{\mathbf{o}\}$ align. We report the mean disparity from all the videos.

Results. Results are shown in Figure 4.7. Both position and orientation disparity of the final model are relatively small (< 0.5), showing that the camera estimation from 3DTRL is partially aligned with the ground truth. We also observe that the disparity decreases over the training epochs.

4.4.4 Qualitative Evaluations

We visualize some qualitative results for an intuitive understanding of 3DTRL's effectiveness. The visualization contains estimated pseudo-depth maps which are from the key intermediate step of 3DTRL. We use ImageNet-1K trained DeiT-T+3DTRL and its validation samples for visualization. In Figure 4.8, we observe a fine separation of the object-of-interest and the background in most of the pseudo-depth maps. Thus, these predicted pseudo-depth maps are sufficient to recover the tokens corresponding to object-of-interest in 3D space. For a qualitative evaluation on camera pose, please refer to Figure 4.9, where we show the images with the same/similar object pose (in different environments) result in similar estimated extrinsics regardless of the background. We believe our estimated extrinsics are doing object-centric canonicalization of images with respect to their object poses, in order to optimize the representations for the final losses.

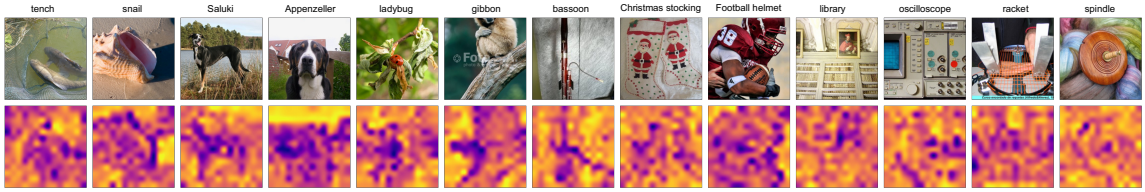


Figure 4.8: **Top**: IN-1K samples with corresponding class label. **Bottom**: Estimated pseudo-depth map, interpolated from 16×16 to 224×224 for better understanding. Depth increases from blueish to yellowish region. More examples are in Figure 4.11.

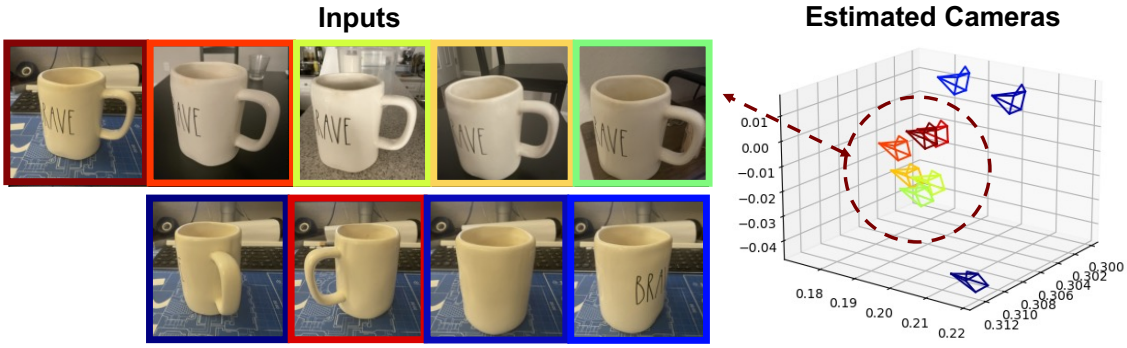


Figure 4.9: Qualitative experiment on how 3DTRL reacts to similar and different poses of the same object. Photos are taken by a regular smartphone. We use DeiT-T+3DTRL trained on ImageNet. The estimated cameras from the similar poses (first row) are clustered as shown in the red-dashed circle, while the other cameras from different poses (second row) are apart. We notice an outlier whose estimation is also introduced in this cluster, owing to the model invariance to horizontal flipping which is used as an augmentation during training.

Table 4.3: Results of using 3DTRL in more Transformer architectures, on CIFAR and multi-view video alignment datasets. Reported numbers are accuracy for CIFAR and Kendall's tau for video alignment, both are the higher the better. We show 3DTRL generally improves the performance in all tasks.

Model	CIFAR-10	CIFAR-100	Pouring	Pick
Swin-T	50.11	21.53	0.584	0.623
+3DTRL	**50.29** (+0.18)	**21.55** (+0.02)	**0.683** (+0.099)	**0.640** (+0.017)
TnT-S	81.25	54.07	0.740	0.640
+3DTRL	**82.43** (+1.18)	**56.00** (+1.93)	**0.792** (+0.052)	**0.671** (+0.031)

4.4.5 3DTRL on More Transformer Architectures

3DTRL is designed to be a plug-and-play model for Transformers. We test 3DTRL with more Transformer architectures on CIFAR and two multi-view video alignment datasets. We use Swin [152] (Tiny) and TnT [97] (Small). Results are provided in Table 4.3. We find that 3DTRL generally improves the performance of two Transformer architectures in all the datasets. This confirms 3DTRL is applicable to different Transformer architectures. The relative small improvement over the Swin backbone is due to the strong inductive bias from its local window.

4.4.6 Ablation Studies

We conduct our ablation studies on image models mostly using *CIFAR* for image classification and *Pick* for multi-view video alignment.

MLP vs. 3DTRL. 3DTRL is implemented by several MLPs with required geometric transforms in between. In this experiment, we replace 3DTRL with the similar number of fully-connected layers with residual connection, to have comparable parameters and computation as 3DTRL. Results are provided in Table 4.5. We find that MLP implementation is only comparable with the baseline performance despite the increase in parameters and computation. Thus, we confirm that the geometric transformations imposed on the token representations is the key to make 3DTRL effective.

Token Coordinates Estimation. In this ablation, we show how estimating only depth compared to estimating a set of 3 coordinates xyz differs in 3DTRL. We find

at Line 4 and 5 in Table 4.5 that depth estimation is better because it uses precise 2D image coordinates when recovering tokens in 3D space, whereas estimating xyz regresses for the 3D coordinates without any geometric constraints. Also, we find that estimating xyz hampers the performance in image classification task more than video alignment. This is because estimating 3D coordinates is harder when the training samples are in unconstrained scenarios. In contrast, video pairs in an alignment dataset share the same scene captured from different view angles, which facilitates the recovery of token in 3D space.

How to incorporate 3D positional information in Transformers? By default, we use Equation 4.5 to incorporate the 3D positional information in Transformer backbone through a learned positional embedding p^{3D}. In this experiment, we directly infuse the estimated 3D world coordinates p^{world} within the token representation by concatenating them across the channel axis. The $(m+3)$-d feature is then projected back to m-d by a MLP. We keep parameters and computation comparable to the default 3DTRL. We test this 3DTRL variant (Embedding $\rightarrow$ Concat.) and results are presented in Table 4.5. We find that the concatenation variant outperforms the default variant of 3DTRL on CIFAR-10, but comparable and worse results in CIFAR-100 and Pick. This observation substantiates the instability of using raw 3D coordinates. In comparison, the use of 3D positional embedding is generalizable to more challenging and diverse scenarios.

Comparison with perspective augmentation In this experiment, we investigate whether perspective augmentation applied to input images will help the model to learn viewpoint-agnostic representations. We test with DeiT+3DTRL on multi-view video alignment task. The training procedure is the same for all variants. Results are shown in Table 4.4. From the results we show that naively adding perspective augmentation does not improve the viewpoint-agnostic representation learning. Instead, it harms the performance compared to the DeiT baseline, since the perspective augmentation is overly artificial compared to the real-world viewpoint changes. Such augmentation does not contribute to viewpoint-agnostic representation learning.

Where should we have 3DTRL? We vary the location of 3DTRL to empirically study the optimal location of 3DTRL in a 12-layer DeiT-T backbone. In Figure 4.10, we find that inserting 3DTRL at the earlier layers (after 1/3 of the network) yields the best performance consistently on both datasets.

Table 4.4: Comparison between 3DTRL and perspective augmentation on training data. Overall, perspective augmentation shows a negative effect on all the tasks, because the perspective augmentation on image is not the real viewpoint change.

Model	Pouring	Pick	MC	Can	Lift
DeiT	0.426	0.244	-0.115	0.789	0.716
DeiT + Perspective Augmentation	0.200	-0.249	-0.419	0.342	0.486
DeiT + 3DTRL	**0.740**	**0.635**	**0.392**	**0.824**	**0.739**

Table 4.5: Ablation study results. For CIFAR, we test on models based on tiny (T), small (S) and base (B) backbones (DeiT) and report accuracy(%). For Pick we only test base model and report alignment error.

Method	**CIFAR-10** (T/S/B)	**CIFAR-100** (T/S/B)	**Pick**
DeiT	74.1/ 77.2 / 76.6	51.3 / 54.6 / 51.9	0.216
DeiT + MLP	74.2 / 77.2 / 76.5	47.9 / 54.7 / 53.4	0.130
DeiT + 3DTRL	78.8 / 80.7 / 82.8	**53.7** / 61.5 / **61.8**	**0.116**
Depth Estimation → *xyz* Estimation	76.7 / 78.2 / 77.4	48.3 / 54.1 / 52.6	0.134
Embedding → Concat.	**80.7** / **83.7** / **84.9**	53.4 / **61.8** / 60.2	0.133

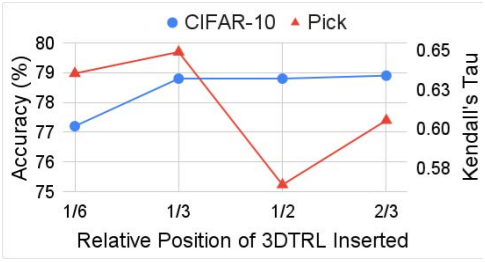


Figure 4.10: Results on inserting 3DTRL at different locations.

4.4.7 3DTRL for Video Representation Learning

In this section, we illustrate how 3DTRL can be adapted for video models. Since we aim at learning viewpoint-agnostic representations, as a natural choice we validate the effectiveness of 3DTRL on video datasets with multi-camera setups and cross-view evaluation. Consequently, we conduct our experiments on two multi-view action recognition datasets: **Toyota Smarthome** [57] (Smarthome) and **NTU-RGB+D** [210] (NTU) for the task of action classification. For evaluation on Smarthome, we follow Cross-View 2 (CV2) and Cross-Subject (CS) protocols proposed in [57], whereas on NTU, we follow Cross-View (CV) protocol proposed in [210]. In cross-view protocols, the model is trained on a set of cameras and tested on a different set of cameras. Similarly for cross-subject protocol, the model is trained and tested on different set of subjects.

Network architecture & Training / Testing TimeSformer [22] is a straightforward extension of ViT [68] for videos which operates on spatio-temporal tokens from videos, so that 3DTRL can be easily deployed to TimeSformer as well. Similar to

Table 4.6: Results on action recognition on Smarthome and NTU. *Acc* is classification accuracy (%) and *mPA* is mean per-class accuracy. In methods using Kinetics-400 (K400) pre-training, TimeSformer backbone is always initialized with pre-trained weights, and 3DTRL w/, w/o K400 denotes 3DTRL is randomly initialized and is initialized from pre-trained weights respectively.

Method	**Smarthome** (CV2)		**Smarthome** (CS)		**NTU** (CV)
	Acc	*mPA*	*Acc*	*mPA*	*Acc*
TimeSformer [22]	59.4	27.5	75.7	56.1	86.4
+ **3DTRL**	**62.9** (+3.5)	**34.0** (+6.5)	**76.1** (+0.4)	**57.0** (+0.9)	**87.9** (+1.5)
			Kinetics-400 pre-trained		
TimeSformer [22]	69.3	37.5	77.2	57.7	87.7
+ 3DTRL w/o K400	69.5 (+0.2)	39.2 (+1.7)	77.5 (+0.3)	58.9 (+1.2)	**88.8** (+1.1)
+ 3DTRL w/ K400	**71.9** (+2.6)	**41.7** (+4.2)	**77.8** (+0.6)	**61.0** (+2.3)	88.6 (+0.9)

our previous experimental settings, we place 3DTRL after 4 Transformer blocks in TimeSformer.

Results In Table 4.6, we present the action classification results on Smarthome and NTU datasets with 3DTRL plugged in TimeSformer. 3DTRL can easily take advantage of pre-trained weights because it does not change the relying backbone Transformer – just being added in between blocks. In Table 4.6, we present results for two fine-tuning scenarios: (a) 3DTRL w/o K400 and (b) 3DTRL w/ K400. For the first scenario (a), TimeSformer is initialized with K400 pre-training weights and leave the parameters in 3DTRL randomly initialized. Then in the fine-tuning stage, all the model parameters including those of 3DTRL is trained. In the second scenario (b), all parameters in TimeSformer and 3DTRL are pre-trained on K400 from scratch and fine-tuned on the respective datasets.

We find that all the variants of 3DTRL outperforms the baseline TimeSformer results. Our experiments show that although there is an improvement with 3DTRL compared to the baseline for different fine-tuning strategy, it is more significant when 3DTRL is pre-trained with K400. However, when large-scale training samples are available (NTU), 3DTRL does not require K400 pre-training. To sum up, 3DTRL can be seen as a crucial ingredient for learning viewpoint-agnostic video representations.

4.5 Conclusion

In this chapter, we have presented 3DTRL, a plug-and-play module for visual Transformer that leverages 3D geometric information to learn viewpoint-agnostic representations. Within 3DTRL, by pseudo-depth estimation and learned camera parameters, it manages to recover positional information of tokens in a 3D space. Through our extensive experiment, we confirm 3DTRL is generally effective in a variety of visual understanding tasks including image classification, multi-view video alignment, and cross-view action recognition, by adding minimum parameters and computation overhead.

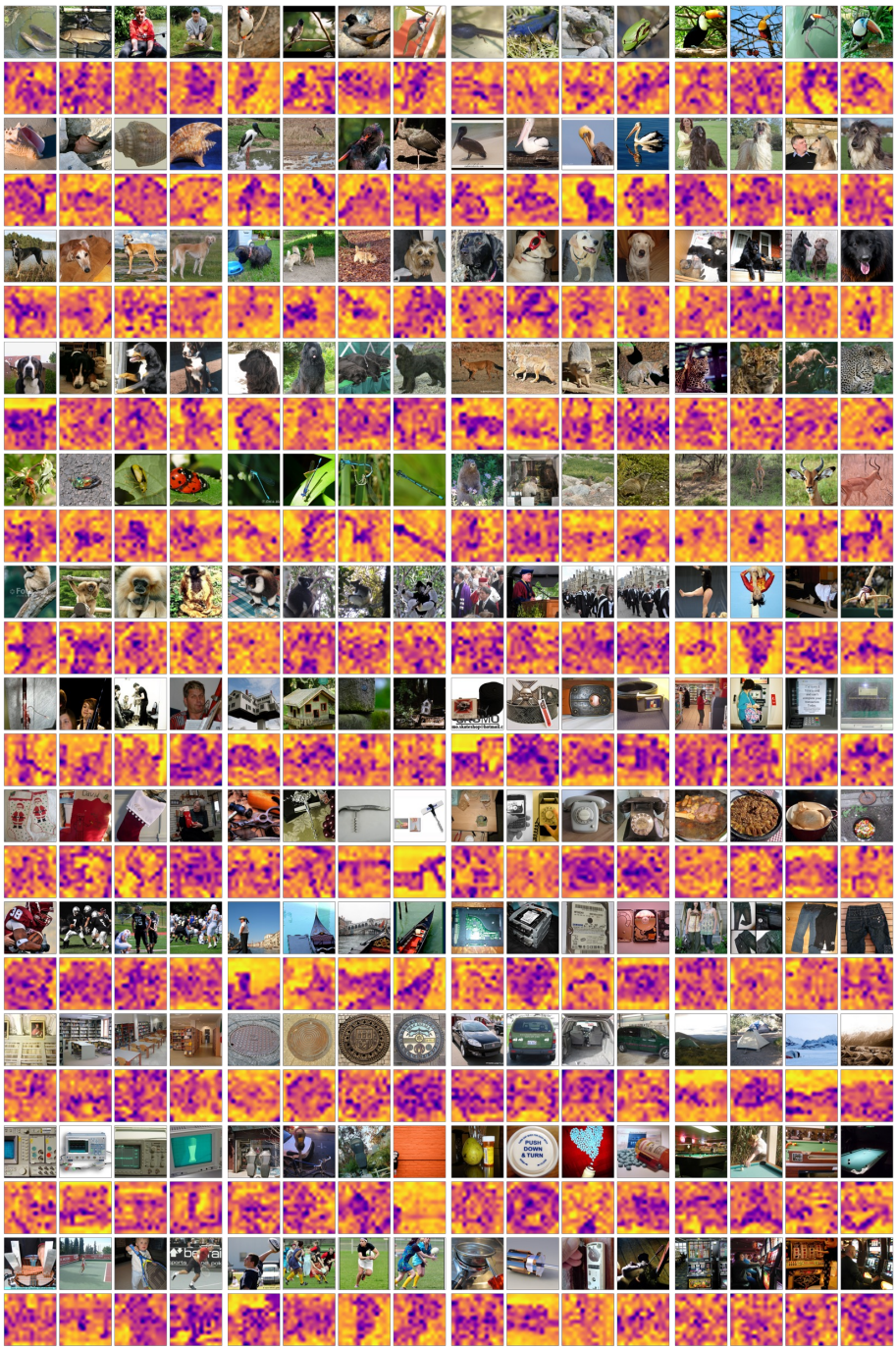


Figure 4.11: More examples of pseudo-depth maps.

Chapter 5

StARformer: Transformer with State-Action-Reward Representations for Robot Learning

5.1 Overview

Reinforcement Learning (RL) naturally operates in a sequential manner, wherein an agent observes a state from the environment, performs an action, observes the next state, and receives a reward from the environment. With the recent advances, RL has been formulated as a sequential decision-making task, and Transformer [251] architectures have become applied to this task as generative trajectory models. Given past experiences of an agent composed of a sequence of state-action-reward triplets, a model iteratively generates an output sequence of action predictions [35, 117]. This novel formulation has been shown to be quite useful, especially in terms of its capability to model long-term sequences [117] and sequence distributions [35].

In the state-of-the-art Transformer models for RL such as [35, 117], an input sequence is plainly processed through self-attention – the core modeling component of Transformers [251]. Thus, a given state, action, or reward token may attend to any of the (previous) tokens in the sequence, which allows the model to capture long-term relations. In the case of visual RL, image states are encoded using convolutional networks (CNNs) to create tokens, before processing through self-attention.

However, tokens within adjacent time steps generally exhibit strong connectivity as a result of potential causal relations. For instance, states in the recent

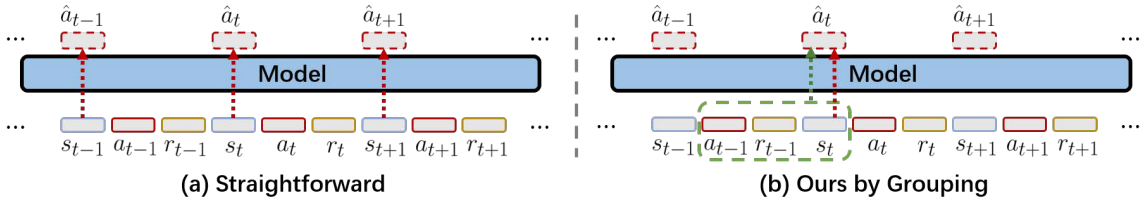


Figure 5.1: Illustration of RL as a sequence modeling task using Transformer: (a) A straightforward approach and, (b) Our proposed improvement. Our intuition is to explicitly use local features to facilitate long-term sequence modeling. Red arrows indicate actions being *autoregressively-generated* based on a state, while also considering previous tokens. Green arrows denote our explicit local representation (i.e., grouping), which further improves action generation, combined with the existing state representations. Attention maps between action and pixel state patches produced by our method (in Breakout environment) are visualized in (c). Different regions-of-interest (i.e., ball, paddle, and lower-level bricks) are focused upon each attention head. In the second attention map from the left, weights in the paddle region are directed towards right (circled in red), corresponding to the semantic meaning of the "right" action.

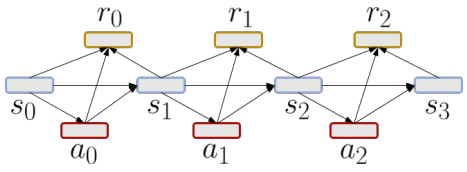


Figure 5.2: MDP view of an RL process. Only the connected pairs (denoted by directed arrows) are causally related, whereas all other elements are mutually independent. For instance, state s_1 only depends on previous state s_0 and action a_0, action a_1 is taken based on state s_1, and no direct relations exist between action a_0 and reward r_2.

past have a stronger effect on the subsequent action than those in the distant past. Similarly, the immediate-future state and the corresponding reward are the direct results of the current action. In the extreme case of the Markov Decision Process (MDP), which is used to formulate RL problems, the relations are far stronger and more restricted (see Fig. 5.2). In the aforementioned scenarios, a Transformer naively attending to all tokens naively may suffer from excess information or dilute the truly-essential relation priors, thus making the learning-process harder. This is especially critical when the input sequences are significantly large, either spatially [267] or temporally [117] dimension, and when Transformer models become heavy, i.e., contain a large number of layers [243]. Moreover, the tokenization of overall image states (as a whole) based on CNNs further restricts Transformer models from capturing detailed spatial relations, resulting in the loss of potentially critical information, particularly in RL tasks with fine-grained regions of interest.

To alleviate these issues, we propose the **St**ate-**A**ction-**R**eward Transformer (**StAR**former), a Transformer architecture that learns **St**ate-**A**ction-**R**eward-representations (i.e., **StAR**-representations) for visual RL. StARformer consists of two basic components: a Step Transformer and a Sequence Transformer. The Step Transformer learns local representations (i.e., StAR-representations) based on self-attention across state-action-reward tokens *within a window of a single time-step*. When learning StAR-representations, we use ViT-like [67] patch-wise embeddings of image states to retain fine-grained spatial information. The Sequence Transformer then combines StAR-representations with pure image state representations *from the entire sequence* to generate action predictions. Pure state representations are convolutional features of the image states. This modeling approach can explicitly capture strong short-term relationships while accounting for the overall trajectory. In our experiments, we show that StARformer outperforms the state-of-the-art Transformer-based method in both offline-RL and imitation learning settings, while being more-compliant with longer input sequences in comparison. We also find that multiple different state embeddings (such as patch-wise and convolutional) can yield better representations. Finally, we deploy our method on an actual robot which performs a human-following task, to demonstrate its potential in real-world imitation learning settings.

Our contributions are as follows: (1) we model single-step transitions explicitly, relieving model capacity to improve focus on long-term relations; (2) we use both local and global information (separately and combined) to tackle long-term sequence modeling, and (3) we verify its applicability in real-world robot learning.

5.2 Preliminary

5.2.1 Transformer

Transformer [251] architectures have diverse applications in language [64] and vision tasks [11, 67]. Given a sequence of input tokens $X = \{x_1, x_2, ..., x_n\}$, where $\forall x \in X, x \in \mathbb{R}^d$, a Transformer layer maps the sequence to an output sequence of tokens $Z = \{z_1, z_2, ..., z_n\}$, where $\forall z \in Z, z \in \mathbb{R}^d$ through Multi-head Self-Attention (MSA) [64], followed by Multi-Layer Perceptron (MLP) blocks with a residual connection [100] and Layer Normalization (LN) [16]:

$$\begin{aligned} Z' &= \text{MSA}(\text{LN}(X)) + X \\ Z &= \text{MLP}(\text{LN}(Z')) + Z'. \end{aligned} \tag{5.1}$$

A Transformer model is obtained by stacking multiple such layers. The mapping for each layer (l) is denoted by $F(\cdot)$: $Z^l = F(Z^{l-1})$. We use the notation $F(\cdot)$ to represent a Transformer layer in the remaining sections.

Self-attention [44, 147, 183, 251] is the core Transformer component that models pairwise relations between tokens. As presented in [251], an input token representation X is linearly mapped into query, key and value representations — that is $\{Q, K, V\} \in \mathbb{R}^{n \times d}$ respectively — to compute self-attention as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V. \tag{5.2}$$

The idea is to aggregate values based on pairwise similarities between queries and keys. In such a mechanism, each token can "attend to", i.e., aggregate all tokens in the sequence, with a specified weight based on learned parameters and token content.

Vision Transformer (ViT) [67] extends the same concept of self-attention to the image domain by mapping a set of non-overlapping image patches to a sequence of tokens using a fully-connected (FC) layer. Given an input image $s \in \mathbb{R}^{H \times W \times C}$, a set of n non-overlapping local patches $P = \{p_i\} \in \mathbb{R}^{h \times w \times C}$ is extracted, flattened and linearly mapped to a sequence of tokens $\{x_i\} \in \mathbb{R}^d$, through a fully-connected (FC) layer $\mathbb{R}^{hwc} \rightarrow \mathbb{R}^d$. We use a similar tokenization approach as a part of our state token embeddings.

5.2.2 RL as Sequence Modeling

We consider a Markov Decision Process (MDP), described by tuple $(\mathcal{S}, \mathcal{A}, P, \mathcal{R})$, where $s \in \mathcal{S}$ represents the state, $a \in \mathcal{A}$, the action, $r \in \mathcal{R}$, the reward, and P, the

transition dynamics given by $P(s'|s,a)$. In MDP, a trajectory (τ) is defined as the past experience of an agent, which is a sequence composed of states, actions, and rewards in the following temporal order:

$$\tau = \{s_1,\ a_1,\ r_1,\ s_2,\ a_2,\ r_2,\ \ldots,\ s_t,\ a_t,\ r_t\}. \tag{5.3}$$

RL is formulated as a sequence modeling task by considering action predictions from past experience [35, 117] according to the following equation:

$$Pr(\hat{a}_t) = p(a_t|\ s_{1:t},\ a_{1:t-1},\ r_{1:t-1}). \tag{5.4}$$

This novel scheme does not employ conventional value estimation or policy gradient methods.

A recent study [35, 117] attempted to use an existing Transformer architecture [193] for RL with the aforementioned formulation. In [35, 117], the states (s), actions (a), and rewards (r) are considered as input tokens (see Fig. 5.1a), and a causal mask is used to ensure an autoregressive output sequence generation (that is, Eq. 5.4). Here, a token could access any of its corresponding prior tokens—in time— through self-attention.

In contrast, our formulation attends tokens with (potentially) strong causal relations *explicitly*, while also attending to long-term relations as well. To achieve this, we split each trajectory into small groups of state-action-reward tuples (i.e., s, a, r) to learn local relations within the tokens of each group through self-attention (see Fig. 5.1b). We then model long-term relations in conjunction with the learned local relations. Our grouping is based on the intuition that the local causal relations between s, a and r are strong; that is, the reward r_{t-1} and the state s_t are direct results of the action a_{t-1}. Consequently, StARformer learns strong intermediate representations from local groups of (a_{t-1}, r_{t-1}, s_t) explicitly, to facilitate long-term sequence modeling. During the learning of local representations, we further split each image state into patches when learning local representations to more effectively capture fine-grained relations between the action and local state regions. We find that our scheme is similar to the divided spatial-temporal attention in [22], which is applied in the video domain with the intuition of reducing computation requirements.

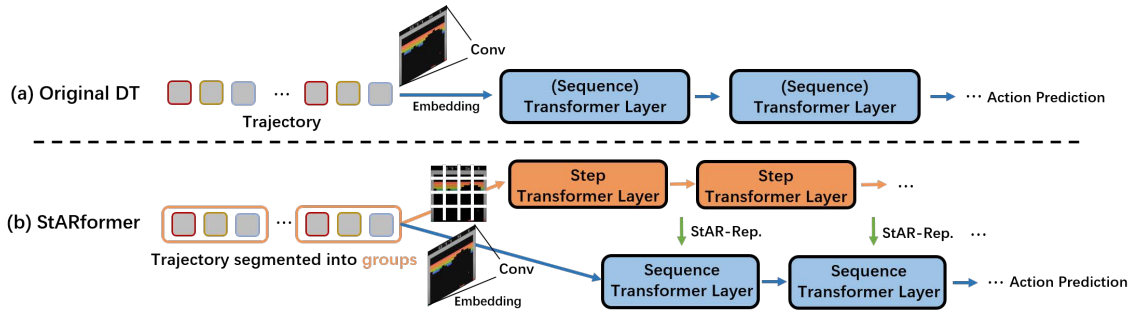


Figure 5.3: (a) Structure summary of original DT [35], where its Transformer layers act similar as our Sequence Transformer. (b) StARformer consists of Step Transformer and Sequence Transformer, to separately model a single-step and the sequence as-a-whole, respectively. Two types of layers are connected at each level via learned StAR-representations. In terms of state embedding methods, DT uses only convolution, while StARformer uses ViT-like [67] embeddings (patches) in Step Transformer and convolution in Sequence Transformer separately.

5.3 StARformer

5.3.1 Overview

StARformer consists of two basic components: Step Transformer and Sequence Transformer, together with interleaving connections (see Fig. 5.3). Step Transformer learns StAR-representations from strongly-connected local tokens *explicitly*, which are then fed into the Sequence Transformer along with pure state representations to model the whole input trajectory. At the output of the final Sequence Transformer layer, we make action predictions via a prediction head. In the following subsections, we will introduce the two Transformer components, and their corresponding token embeddings in detail.

5.3.2 Step Transformer

State-Action-Reward Embeddings

Grouping state-action-reward: Our intuition of grouping is to model strong local relations explicitly. To do so, we first segment a trajectory (τ) into a set of groups, each of which consists of a previous action (a_{t-1}), corresponding reward (r_{t-1}), and

current state (s_t)[1] (see Fig. 5.4). Each element within a group has a strong causal relationships with the other elements.

Patch-wise state token embeddings: In Step Transformer, each input image state is tokenized by dividing it into a set of non-overlapping spatial patches along its spatial dimensions, in accordance with ViT [67]. Consider an image state $s_t \in \mathbb{R}^{H\times W\times C}$, we divide it into patches $\{s_t^i\}, \in \mathbb{R}^{h\times w\times C}$, which generates a total of HW/hw state tokens per image. We use a linear projection, where the weights of which are shared by each state patch across all groups, to create token embeddings ($\mathbb{R}^{hwC} \to \mathbb{R}^d$ for image states or $\mathbb{R}^1 \to \mathbb{R}^d$ for vector states) according to the following equation:

$$z_{s_t^i} = \text{FC}(\text{Flatten}(s_t^i)) + e_i^{\text{spatial}}, \tag{5.5}$$

where $e_i^{\text{spatial}} \in \mathbb{R}^d$ represents a spatial positional embedding for each patch location, n is the number of patches, d is the embedding dimension. Note that there are no temporal positional embeddings for patch-wise spatial tokens because they are processed in agnostic of timesteps.

Our motivation for using patch embeddings is to create fine-grained state embeddings that allows the Step Transformer to model the relations between actions and rewards within local state regions. This is especially useful in RL tasks, where local regions of interest can be critical. We empirically validated this in our experiments (Sec 5.5.4).

Action and reward token embeddings: We simply embed the action tokens with a linear projection, and the reward tokens with a linear projection followed by a Tanh$(\cdot)$ activation function, as in [35]. Thus, each token is mapped to an appropriate value range for robot control.

$$z_{a_{t-1}} = \text{FC}(a_{t-1}),\ z_{r_t} = \text{Tanh}(\text{FC}(r_t)). \tag{5.6}$$

Altogether, we obtain a state-action-reward representation as the input to the initial Step Transformer layer, expressed as:

$$Z_t^0 = \{z_{a_{t-1}},\ z_{r_t},\ z_{s_t^1},\ z_{s_t^2},\ \ldots,\ z_{s_t^n}\}. \tag{5.7}$$

We obtain T groups of such token representations per trajectory that are simultaneously processed by the Step Transformer with shared parameters.

[1] We pad the trajectory with a null action and zero reward at the initial state s_1 of the trajectory (see Section 5.3.4)

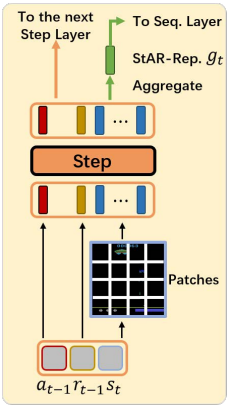


Figure 5.4: Overview of Step Transformer. Output tokens are (1) sent to the next Step Transformer layer and (2) aggregated to produce StAR-representation.

Step Transformer Layer

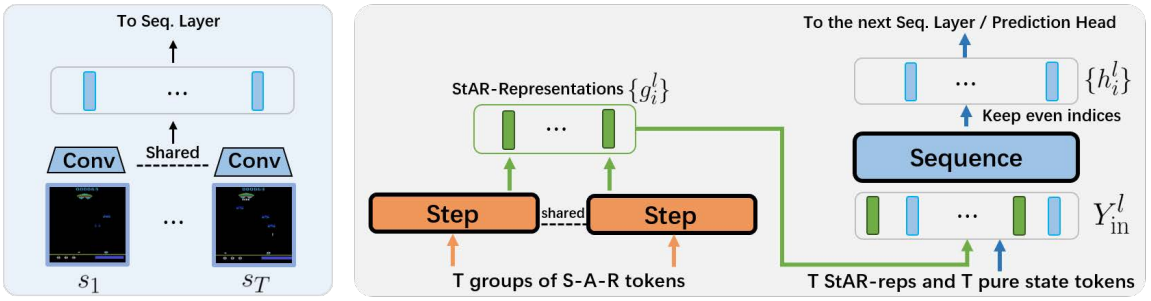
We adopt the conventional Transformer design from [251] (see Section 5.2.1) as our Step Transformer layer. Each group of tokens from the previous layer Z_t^{l-1} is transformed to Z_t^l by a Step Transformer layer with the mapping F_{step}^l:

$$Z_t^l = F_{\text{step}}^l(Z_t^{l-1}). \tag{5.8}$$

Each Step Transformer layer l outputs a State-Action-Reward-representation (StAR-representation) $g_t^l \in \mathbb{R}^D$ by aggregating the output tokens $Z_t^l \in \mathbb{R}^{n\times d}$ (see the green flows in Fig. 5.4). We concatenate the tokens of each group Z_t^l and linearly projecting ($\mathbb{R}^{nd} \to \mathbb{R}^D$), where d and D correspond to embedding dimensions of Step Transformer and Sequence Transformer, respectively.

$$g_t^l = \text{FC}([Z_t^l]) + e_t^{\text{temporal}} \tag{5.9}$$

Here $[\cdot]$ represents the token concatenation within each group, and $e_t^{\text{temporal}} \in \mathbb{R}^D$ represents the temporal positional embeddings for each timestep. Note that we add these temporal positional embeddings to g_t^l at each layer because $[Z_t^l]$ is learned agnostically with respect to time. Finally, the output StAR-representation g_t^l is fed into the corresponding Sequence Transformer layer for long-term sequence modeling.



(a) Pure State Embedding (b) Sequence Transformer and Step-to-Seq. connection

Figure 5.5: (a) Pure state embeddings are learned from shared convolutional layers. (b) Sequence Transformer takes StAR-representation and the pure state tokens and generate output tokens.

5.3.3 Sequence Transformer

Our Sequence Transformer (illustrated in Fig. 5.5) models long-term sequences by looking at the learned StAR-representations and the *pure state tokens* (introduced below) over the entire trajectory.

Pure State Token Embeddings

In contrast to the patch-wise token embeddings in Step Transformer, we embed the overall input image state s_t to generate pure state tokens h_t^0. Each such token represents a single-state representation that describes the state globally in space. We achieve this by processing each state through a CNN encoder, whose convolutional layers spatially combine the features. The convolutional encoder is obtained from [171].

$$h_t^0 = \text{Conv}(s_t) + e_t^{\text{temporal}}, \tag{5.10}$$

where $e_t^{\text{temporal}} \in \mathbb{R}^D$ represents the temporal positional embeddings exactly the same as what we add to g_t.

Sequence Transformer Layer

As in Step Transformer, we use the conventional Transformer layer design from [251] for our Sequence Transformer. The input to the Sequence Transformer layer l consists of representations from two sources: (1) the learned StAR-representations $g_t^l \in \mathbb{R}^D$ from the corresponding Step Transformer layer, and (2) the pure state representation $h_t^{l-1} \in \mathbb{R}^D$ from the previous Sequence Transformer layer.

The two types of token representations are merged to form a single sequence, preserving their temporal order, as follows:

$$Y_{\text{in}}^{l} = \{g_1^l,\ h_1^{l-1},\ g_2^l,\ h_2^{l-1},\ \ldots,\ g_T^l,\ h_T^{l-1}\}. \tag{5.11}$$

We place g_t^l before h_t^{l-1}, which originates from s_t, because g_t^l contains information pertaining to the *previous* action a_{t-1}, which comes prior to s_t in the trajectory. We also apply a causal mask in the Sequence Transformer to ensure that the tokens at time t cannot attend to any future tokens (i.e., $> t$).

Sequence Transformer computes an intermediate set of output tokens as follows:

$$Y_{\text{out}}^{l} = F_{\text{sequence}}^{l}(Y_{\text{in}}^{l}). \tag{5.12}$$

We then select the tokens at even indices of Y_{out}^{l} (where indexing starts from 1) as the pure state tokens h_t^l, which are fed into the next Sequence Transformer layer. Because the even indices correspond to the tokens originating from the pure state representations s_t (Fig. 5.5(b)), they should be used to *predict* actions from an autoregressive perspective (see Fig. 5.1). In contrast, any tokens in Y_{out}^{l} with odd indices do not propagate to the next layer.

$$\begin{aligned} Y_{\text{out}}^{l} &= \{y_{\text{out};1}^{l},\ y_{\text{out};2}^{l},\ \cdots,\ y_{\text{out};2T}^{l}\} \\ h_i^l &:= y_{\text{out};2i}^{l} \end{aligned} \tag{5.13}$$

Therefore, the overall input (StAR-representation and pure state representation) to the subsequent layer can be expressed as (by rewriting Eq. 5.11):

$$\begin{aligned} Y_{\text{in}}^{l+1} &= \{g_1^{l+1},\ h_1^l,\ g_2^{l+1},\ h_2^l,\ \ldots,\ g_T^{l+1},\ h_T^l\} \\ &= \{g_i^{l+1},\ y_{\text{out};2i}^{l}\}_{i=1}^{T}. \end{aligned} \tag{5.14}$$

Action Prediction

The output of the last Sequence Transformer layer (after selection as mentioned above) is used to generate action predictions by processing it through a prediction head $\phi(\cdot)$ linearly mapping to the action dimension (with shared weights for all timesteps): $\hat{a}_t = \phi(h_t^l)$.

5.3.4 Training and Inference

During training, we use trajectories τ with a length T, randomly sampled (and sliced) from a memory buffer. Ground-truth actions are used as labels. Cross-entropy loss is used for a discrete action space, whereas the mean-squared error

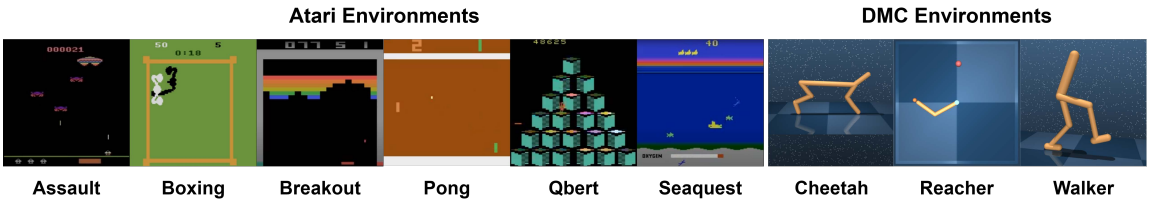


Figure 5.6: Environments used in our experiments: Atari with a discrete action space, and DMC with a continuous action space. As in prior studies, we convert RGB images to grayscale images as the model input.

(MSE) is used for a continuous action space. The overall loss term for a given training sequence is the loss averaged across all T predictions.

At inference, we initialize an input trajectory as $\tau = \{a_0, r_0, s_1\}$, where a_0 is a null action[2] and $r_0 = 0$ is the zero reward which padded at the start of each trajectory. StARformer makes an initial prediction $\hat{a}_1$ based on τ, and receives the next state s_2 and reward r_2. We append these $\hat{a}_1$, s_2 and r_2 to the trajectory τ and repeat this procedure until the end of one RL episode.

5.4 Simulation Experiment Settings

5.4.1 Settings

We use offline RL [141] and imitation learning as our experiment settings. These settings are commonly used in related studies that formulate RL as a sequence modeling task [35, 117], because ground truth labels can be obtained for actions to train a sequence model. In offline RL, we have a fixed memory buffer of sub-optimal trajectory rollouts. Offline RL is generally more challenging than conventional RL [141] due to the shifted distribution.

Imitation learning, however, is the setting that the agent is not exposed to reward signals. Therefore, we simply remove the rewards from the dataset used in offline RL. This presents a significant challenge compared to traditional imitation learning because the provided trajectories are sub-optimal (i.e., with lower rewards than true experts). The only difference in terms of the model structure is that there is now one less reward token in the input of Step Transformer in our model, and

[2]We use an additional action to represent null action in the discrete action space, and a zero vector in the continuous action space.

T less reward tokens in the baseline model (T is the number of time-steps in the input trajectory).

5.4.2 Environments

We use image-based Atari [19] (discrete action space) and DeepMind Control Suite (DMC) [247] (continuous action space) to evaluate our model's performance in different types of tasks, as shown in Fig. 5.6 with image examples. We select six games from Atari: Assault, Boxing, Breakout, Pong, Qbert, and Seaquest. As in [35] we use 1% (500k steps) of the DQN replay buffer dataset [1] to perform a thorough and fair comparison. We select five continuous control tasks in DMC [247]: Ball-in-cup-catch, Cheetah-run, Finger-spin, Reacher-easy, and Walker-walk. In DMC, we collect a replay buffer (i.e. sub-optimal trajectories) generated by training an SAC [94] agent from scratch for 500k steps in each task. This is a similar setting to "Medium-Replay" in the D4RL [79] dataset used by DT [35]. Note that these continuous control tasks involve image inputs, which the previous study [35] did not cover (originally using Gym [25]). Furthermore, learning continuous control from pixels is more challenging than directly learning from actual joint states [271], and is generally harder than Atari games due to the larger continuous action space (eg. six dimensions in Cheetah and Walker). We report the absolute value of episodic returns (i.e., cumulative rewards). The results are averaged across multiple randomly initialized runs (7 seeds in Atari and 10 in DMC), each run is evaluated by 10 randomly initialized episodes.

5.4.3 Baselines

We select Decision-Transformer (DT) [35], a SOTA Transformer-based sequence modeling method for RL. Although Trajectory-Transformer [117] exsists, it is not designed for image inputs. We use most of the same hyper-parameters used in original DT [35] for the Atari environments, without extra tuning (details in Appendix Tables 4 and 5). Because DMC environments are not covered by DT [35], we carefully tune the baseline first and then use the same set of hyper-parameters in our method. We also compare with the SOTA non-Transformer offline-RL methods, including CQL [138], QR-DQN [53], REM [2] in Atari, and BEAR [136], IQL [133], and TD3+BC [80] in DMC. For imitation (behavior cloning), we only compare with DT [35] and straightforward behaviour cloning with ViT (denoted as BC-ViT).

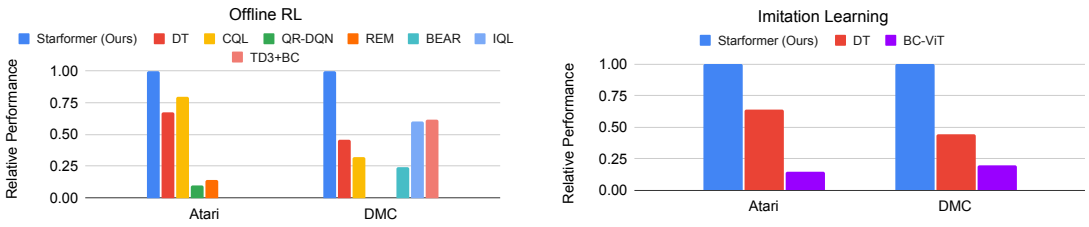


Figure 5.7: Relative performance of episodic returns. We compare StAR with DT [35] in both offline-RL and imitation settings. We also compare with previous offline-RL methods in their capable environments, that is, CQL [138], QR-DQN [53] and REM [2] in Atari (discrete action space), and CQL [138], BEAR [136], IQL [133], TD3+BC [80] in DMC (continuous action space). We introduce a baseline method BC-ViT for imitation learning, which performs behavior cloning using a ViT encoder naively. StAR consistently outperforms others in both settings. Here, the results are averaged across all environments and random seeds (7 in Atari and 10 in DMC), and normalized w.r.t. the performance of StAR. Please refer to Table 1 in the Appendix for absolute values corresponding to the above results, with more comparisons of offline-RL methods.

5.5 Results on Simulation Environments

5.5.1 Improving Sequence Modeling for RL

We first compare our StARformer (StAR) with the state-of-the-art Transformer-based RL method in Atari and image-based DMC environments, under both offline RL and imitation learning settings. We select the Decision-Transformer proposed in [35], (referred to as DT) as our baseline. Here, we maintain $T = 30$ for all environments, which is the number of time-steps (length) of each input trajectory (τ). We also compare our method to CQL [138], a SOTA non-Transformer offline-RL method. Fig. 5.7 shows that our method outperforms the baselines in both offline RL and imitation learning settings, which suggests that our method models image sequences more effectively.

5.5.2 Scaling-up to Longer Sequences

In this experiment, we evaluate how StARformer and DT perform with different input sequence lengths, specifically $T = \{10, 20, 30\}$. In Fig. 5.8, we see that StARformer achieves satisfactory performance with longer trajectories, whereas

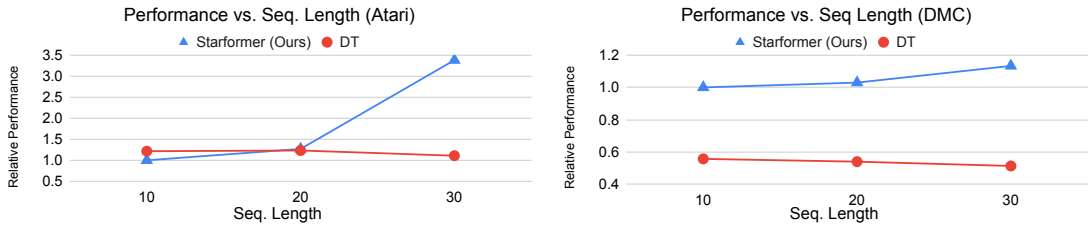


Figure 5.8: Change in performance with the length of input sequence, $T \in \{10, 20, 30\}$, in Atari and DMC (averaged across tasks). Here, the results are normalized w.r.t. StARformer performance at $T = 10$. We evaluate with offline-RL, as DT is more competitive in this setting. The results show a performance gain in StARformer with longer input sequences, whereas DT [35] exhibits a drop in performance. This validates the superior performance of our method in long-term sequence modeling. Please refer to Appendix Fig.1 for a more detailed comparison.

DT exhibits saturation as early as $T = 10$. This validates our claim that considering short-term and long-term relations separately (and then fusing) helps the model scale-up to longer sequences. Instead of learning Markovian pattern attentions [117] implicitly, we model it explicitly in our Step Transformer. This acts as an inductive bias, relieving the capacity of Sequence Transformerto place more focus on long-term relations. In contrast, DT uses the off-the-shelf language model GPT [193], that does not consider the Markov property. Although GPT [193] structure performs well for long sentences, we show here it does not adequately handle long RL sequences with image states.

5.5.3 Reward setting: Return-to-go, stepwise reward, or no reward at-all?

We also intend to determine how different reward settings affect sequence modeling, specifically, return-to-go (RTG) [35], stepwise reward, and no reward at-all. Decision-Transformer [35] originally uses RTG $\hat{R}_t$, which is defined as the sum of future step-wise rewards: $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$. Similar concepts have been studied in hindsight-related methods to facilitate RL tasks [9, 75, 123, 137, 142, 189, 229]. Stepwise reward r_t is the immediate reward generated by the environment in each step, which is generally used in most RL algorithms. Our StARformer uses step-wise reward as the default, guided by the motivation of modeling single-step transitions. The no reward setting corresponds to an imitation formulation.

The results for each reward setting are presented in Fig. 5.9. StARformer and DT

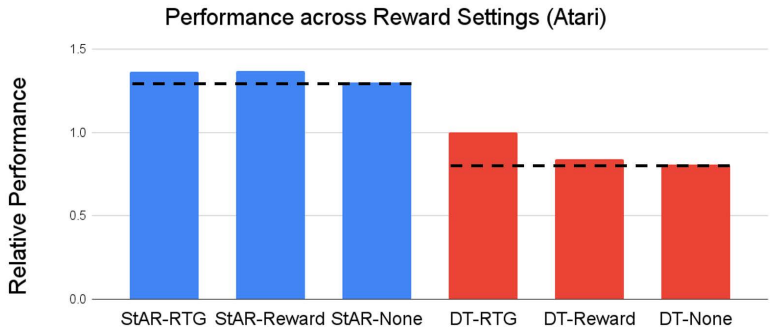


Figure 5.9: Performance in different reward settings: return-to-go (RTG), stepwise reward, and no reward at-all (labeled as 'None') settings. StARformer performs similarly in the two settings with reward, and demonstrates slight degradation when rewards are not provided. In contrast, DT shows a higher performance when used with RTG, compared to DT under either stepwise or no reward settings. Results are averaged across six Atari environments and normalized w.r.t DT-RTG.

behave differently under various reward settings. Although both methods show an increase in performance when a reward is provided, StARformer performs similarly regardless of RTG or stepwise reward, whereas DT relies more on RTG. Note that in the setting with no reward at-all, the rest of the dataset (i.e. states and actions) is still the same. Sequence modeling can still work on state-action trajectories without a reward when the model has sufficient capacity to mine relevant patterns and generate better representations. This suggests that StARformer without a reward can outperform DT with RTG.

5.5.4 Visualization

We present the attention maps between action and state patches in Step Transformer(see Fig. 5.10) at several timesteps extracted from a trajectory in Breakout. In this game, the agent should move the paddle to bounce the ball back from the bottom, after the ball falls down while breaking the bricks on the top. In the presented attention maps, the regions with a high attention score (highlighted) mainly overlap with the locations of the ball, paddle, and potential target bricks. We find the attention maps corresponding to Head #1 are particularly interesting. Here, the focused regions corresponding to the paddle show a directional pattern, associated with the semantic meaning of the actions "moving the paddle right", "left", or "stay". This affirms that Step Transformer captures the essential spatial

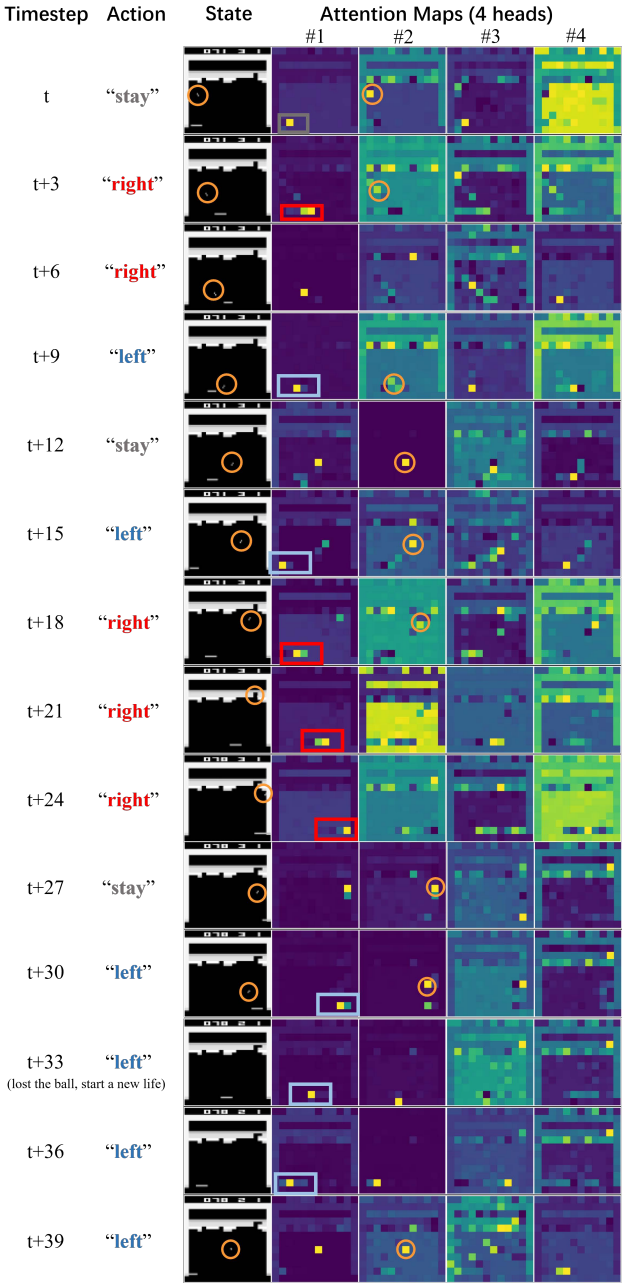


Figure 5.10: Visualization of attention maps in our Step Transformer , extracted from a clip of a Breakout trajectory, with all corresponding timesteps and actions annotated. Attention weights are computed between the action token and state patch tokens.

relations between actions and state patches, which is important for decision making. Moreover, in Head #2, we observe that the focused regions correspond to the locations of the ball, except when the ball is outside of the boundary, too-close to the paddle or indistinguishable within the bricks. Overall, these attention maps suggest that how our model obtains a basic understanding of the Breakout game.

StARformer has two major differences compared to the baseline method DT [35]: (1) it learns two types of representations, namely, StAR-representations and pure state embeddings; and (2) it employs a separate Step Transformer to merge these two types of embeddings. The following sections present ablation studies on each of these distinctions.

StAR-representations and pure state embeddings

In terms of the learned representations, our approach deviates from the baseline method, which models all the tokens from the trajectory (CNN-extracted states, actions, and rewards) directly, whereas we:

- use ViT-like [67] patch-wise image embeddings when learning StAR-representations in Step Transformer, and,
- consider pure state representations h_t (from convolutional layers) together with StAR-representations in Step Transformer.

To investigate the effect of the above changes, we vary the state embedding methods used to learn s_t in Step Transformer and h_t in Sequence Transformer. Specifically, we consider the following: (1) ViT features (patch embeddings, labeled as **P**), (2) convolutional features (labeled as **C**), and (3) none (not having the corresponding embedding, labeled as "__"). The original StARformer can be represented as **P**+**C** (patch embeddings for s_t and convolutional embeddings for h_t). Other variants include: **P**+**P**, **P**+__, **C**+**P**, **C**+**C**, and **C**+__. We also implement a variant of DT using ViT for state embedding (denoted as DT w/ ViT), to match our method in terms of having a similar embedding method and capacity (6.4M parameters vs. 5.3M parameters in our method).

When comparing StARformer with the original DT and DT w/ ViT (see Fig. 5.11(a)), we observe a performance drop in DT when combined with ViT, which suggests that replacing convolutional features with ViT-like features naively does not benefit the model despite the increased capacity (similar to ours). StARformer, however, does not benefit only from the larger capacity, but also from its better structural design, as verified in following experiments (see Fig. 5.11(c)(d)(e)). From Fig. 5.11(b), we find that **C**+__ which only uses convolutional features at

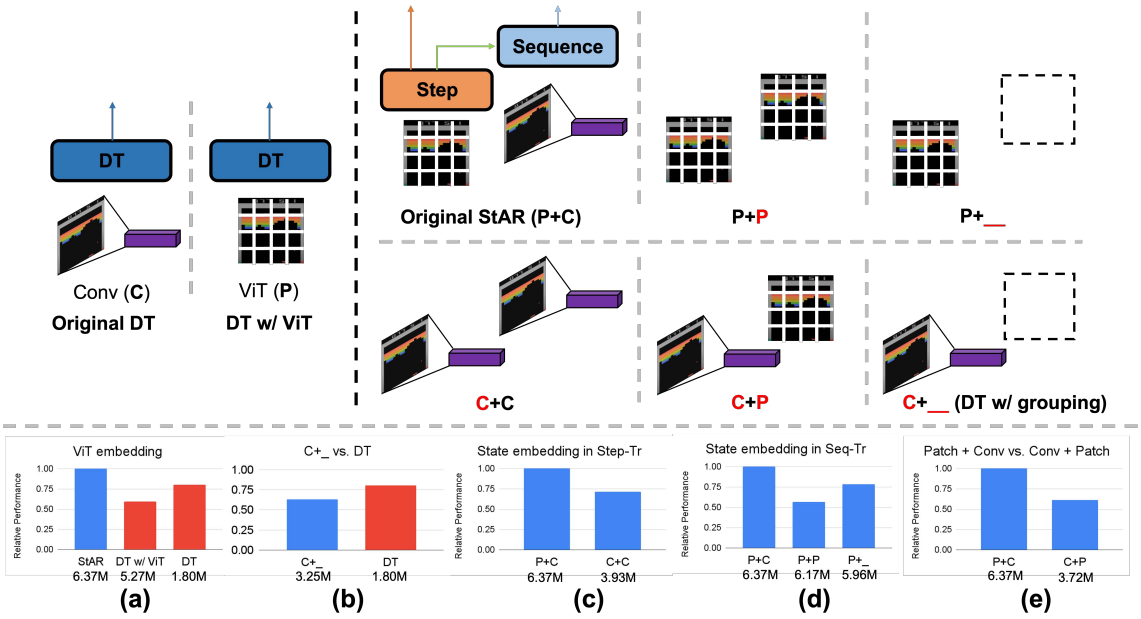


Figure 5.11: (Top): Embedding methods used in the original DT, StARformer (StAR), and their variants. We label ViT (patch embeddings) as **P**, convolution as **C**, and None (not using the corresponding embedding) as "__". (Bottom): (a-e) performance comparisons between variants. Relative performance (in offline-RL, averaged across Atari tasks) in all comparisons is normalized w.r.t. StAR. We attach the number of parameters (in millions (M)) of each method below each bar chart. These observations validate that (1) StAR benefits from the fusion of ViT and convolutional features; (2) ViT performs better in Step Transformer and convolution works better in Sequence Transformer due to the specific context (short-term and long-term, respectively). Specific results for each task are listed in Appendix Table 2.

Step Transformer, performs worse than DT. This is because convolutional features are highly abstracted, making them unsuitable for single-step transition (i.e., fine-grained) modeling.

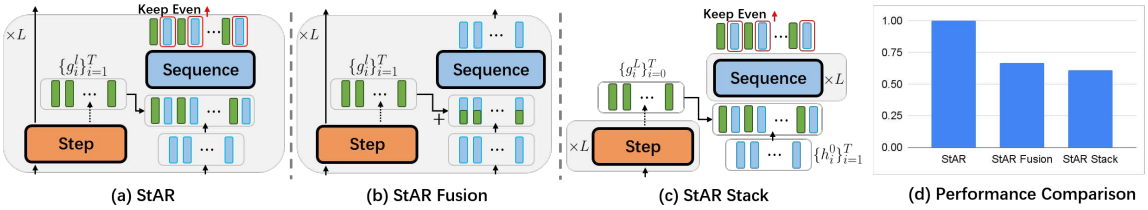
Comparing **P**+**C** with **C**+**C** (Fig. 5.11(c)), the poorer performance of **C**+**C** suggests that patches embeddings are better suited for modeling single-transitions in Step Transformer. In Fig. 5.11(d), we compare **P**+**C** with **P**+**P** and **P**+__. We find that convolutional features work best in Sequence Transformer, which indicates that they provide abstract global information that is useful for long-range modeling (coarse), in contrast to patched embeddings. The observations from these comparisons of StARformer variants suggest that our method benefits from the fusion of patches and convolutional features. This is further evaluated this by comparing **P**+**C** with **C**+**P** (Fig. 5.11(e)), where **P**+**C** performs better, confirming the "fine-grained (patches) to high-level (conv)" fusion method as a best match with our sequence modeling scheme of "single-transition followed-by long-range-context".

In summary, these observations supports two motivations on our model design (using **P**+**C**). (1) StARformer benefits from using **P** and **C** to encode a visual state from *different aspects* — **P** captures local, spatial details and **C** captures global state information. Other variants using only one kind of embeddings are relatively limited to represent a state. (2) Furthermore, the two kinds of embeddings are used *in appropriate modules*. **P** is used in Step Transformer to better fuse with action and reward tokens, and **C** is used in Sequence Transformer in order to summarize states at high level for easier temporal modeling. The variant that exchanges the usage of embeddings (**C**+**P**) has been shown worse than the desired usage.

Step-to-Sequence Layer-wise Connections

5.5.5 Ablations

In our model, we use learned StAR-representations g and pure state representation h together in the Sequence Transformer, implemented as layer-wise connections. In fact, the Step Transformer is connected to the Sequence Transformer via g_t^l in a layer-wise manner (i.e., at each corresponding layer). We investigate why such layer-wise connections are important by comparing with two other variants: (1) g_t^l is fused with h_t^l by summation (referred as StAR Fusion, see Fig. 5.12(b)), and (2) the only connection between the two is from the last layer of Step Transformer to the first layer of Sequence Transformer, in which, the Sequence Transformer is "stacked" on-top of the Step Transformer (referred as StAR Stack, see Fig. 5.12(c)). Results of these configurations are shown in Fig. 5.12(d). It shows that our layer-wise connection method is important for long-term prediction, which fuses information

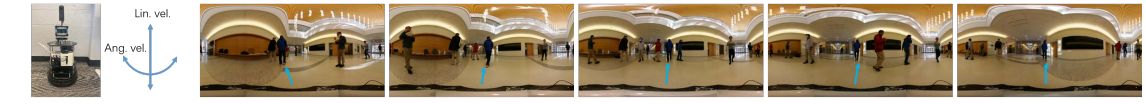


(a) StAR (b) StAR Fusion (c) StAR Stack (d) Performance Comparison

Figure 5.12: Illustration of our ablation study on Transformer connection of our model. Two variants — (b) StAR Fusion and (c) StAR Stack — are shown in comparison to our original (a) StAR model. g^l (green) is the StAR-representation from l-th layer of the Step Transformer and h^0 (blue) denotes the initial pure state embeddings. L is the number of layers. Positional embeddings are omitted for simplicity. (d) Experiments (offline-RL) find that the original structure (StAR), which is a layer-wise fusion, yields higher performance than StAR Fusion and StAR Stack connections, as seen with higher rewards. Please refer Table 3 in the Appendix for per-task results.

from StAR-representation in each layer, and is done through attention rather than direct fusion.

5.6 Real-world Robot Imitation Learning Experiment: Human-following



(a) Ground Mobility Robot System (b) Robot Action Space (c) Robot view / model input (panoramic RGB images) of the human-following task

Figure 5.13: (a) The ground mobility robot system we use. (b) The action space (control command) of the robot includes linear velocity and angular velocity. (c) A trajectory clip from the robot view (model input) of our human-following task. We highlight the locations of target person by blue arrows. In our human-following task, the robot should keep following the target person when other non-target people are actively walking, standing, and crossing.

This section demonstrates how our method can be applied to a ground mobility robot via a human-following task, in the formulation of imitation learning. Our

robot system and human-following task are illustrated in Fig. 5.13. To the best of our knowledge, this is first study to apply sequential modeling methods to real-robot imitation learning.

The reason for this is Sequential modeling (Transformer) can be a cheaper robot imitation learning method that does not require online (i.e., real-robot) policy learning, thus conserving time and human efforts. In such a formulation, learning could be done in only two steps, as in behavior cloning: (1) collecting the demonstration dataset, and (2) training the sequential model (policy). This end-to-end training scheme is also promising in the context of robotic tasks, like navigation.

The following subsections introduce our human-following task (in Sec. 5.6.1) and system settings (in Sec. 5.6.2), followed by the release of a new real-world human-following dataset (in Sec. 5.6.3), and our offline and online evaluation results (in Sec. 5.6.4 and Sec. 5.6.5, respectively).

5.6.1 Human-following Task

We set up a human-following task to collect data in the context of robot learning. In this task, the robot agent should follow a moving human target (as long as possible) without any disruptions, such as losing or changing targets, or interfering with other moving people. A similar navigation in crowded environment has been explored by Monaci et al. [172] and is shown to be very challenging. The experimental site is set up at the lobby of a university department building with up to 12 moving or standing people. First, we collect expert demonstrations (the dataset), described in Sec. 5.6.3. Subsequently, we trained a policy based on the collected dataset. After the convergence of training, the model was ready to be evaluated/used offline or online.

5.6.2 Robot System Settings

To implement the robot system for the human-following task, we construct a mobile robot platform equipped with a computer, a router, a spherical camera, a LiDAR and a projected texture stereo (PTS) camera, as shown in Fig. 5.14(a). The spherical camera captures a panoramic image with 3840×1920 resolution at a frequency of 29.97Hz. The PTS camera acquires 3-channel color and depth images with 1280×720 resolution at a frequency of 30Hz. The LiDAR sensor scans 16-channels of distance ranges in the vertical direction with a 2° interval angle for objects within the $100m$ distance.

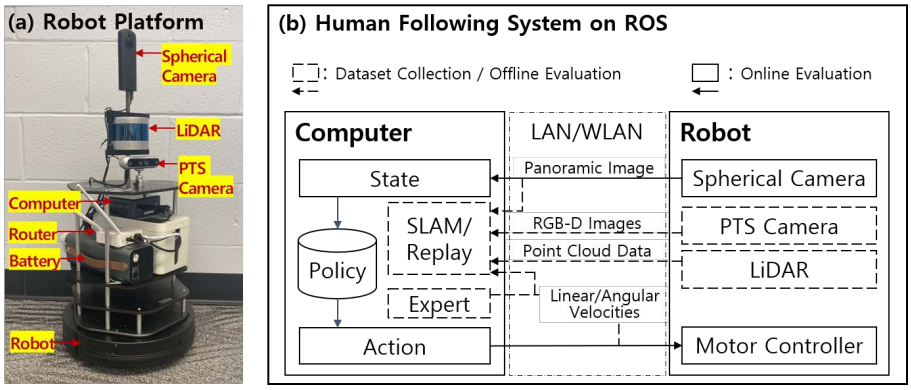


Figure 5.14: Overview of our robot system: (a) robot platform configuration and, (b) software schematics and data flow. Here, the dotted lines and boxes represent the dataset collection and offline evaluation, whereas the solid lines and boxes represent the online evaluation. Note the SLAM is only used to annotate robot global positions, which are then used as labels in the position prediction task and trajectory visualization.

In terms of software, the sensors, robot platform, and the trained model are integrated using Robot Operation System (ROS) as middleware. Shown by dashed lines and boxes in Fig. 5.14(b), the panoramic image, RGB-D image, linear and angular velocities, and point cloud data are synchronized to be saved as the dataset. The collected data are used to train the human-following model and evaluate its offline performance. For this, an expert manually controls the linear and the angular velocities of the robot by a wireless game controller, adapting it to the human-following task. Simultaneous localization and mapping (SLAM) algorithm is used to evaluate the model's performance by estimating the accurate positions and velocities of the robot using the point cloud data from LiDAR. An online evaluation of the trained model is subsequently performed, as shown by the solid lines and boxes in Fig. 5.14(b). We use FOV images as the input. The trained model generates the linear and angular velocities as actions for controlling the robot. The entire system runs at $\sim$10Hz during data collection and online evaluation.

5.6.3 Human-following Dataset

We collected a human-following expert dataset by manually controlling the robot to follow moving human targets. The experiment site was set in a department lobby

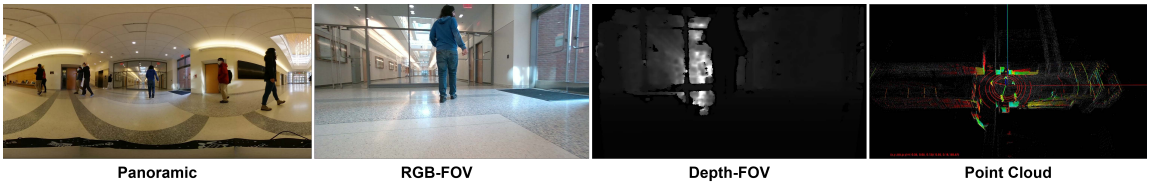


Figure 5.15: A sample frame from the collected human-following dataset. The dataset contains panoramic images, RGB and depth FOV images, and point cloud. Here, the robot is following a person (in the middle of the view) with other non-target people at the experiment site.

with a group of volunteers[3]. During data collection, the robot was continuously controlled to follow a specific person (target) continuously along a trajectory. Other volunteers were required to walk, stand, and sit randomly at the experimental site. They were also encouraged to cross the robot's path to confuse it or occlude the target.

In terms of brief statistics of the dataset, there were 2-12 volunteers simultaneously in the scene, and we collected nine valid trajectories with the robot following different targets, reaching 51,817 frames or ~86 minutes in total. We recorded stereo images, RGB-D images, LiDAR data (point clouds in local coordinates), and robot actions (linear and angular velocities). A sample frame (images and point clouds) of the collected data is shown in Fig. 5.15. Note that our study only considers image modalities as inputs. Point clouds are used to annotate the ground truth of the position prediction task and visualization via a conventional SLAM algorithm.

5.6.4 Offline Evaluation

Evaluation Task Setup

We perform offline evaluation of multiple visual input modalities and on multiple tasks. As input modalities, we consider (a) SRGB: panoramic images from the spherical camera, (b) FRGB: field-of-view (FOV) RGB images from the PTS camera, and (c) FRGB-D: FRGB input plus depth channel images from the PTS camera. We set up two evaluation tasks: policy learning and 2D position prediction.

Policy learning (action prediction): The policy learning task focuses on

[3]We provided the volunteers with free food for their contribution. All volunteers were required to wear masks whenever they were not actively eating or drinking.

predicting robot control commands, which include linear and angular velocities. Here, we consider continuous and discrete action space settings. In the continuous action space, evaluations are performed using the mean squared error (MSE) between the predicted $\hat{a}_t$ and ground truth a_t robot controls (normalized to $[-1, 1]$ range) as given by:

$$\text{error} = \frac{1}{N} \sum_{t} \|\hat{a}_t - a_t\|_2, \tag{5.15}$$

where N denotes the trajectory length. In the discrete action space, the original 2D linear-angular space is divided into finite categories. We create two discretized action spaces — 6-action and 10-action spaces — as shown in Table 5.1. Here, actions consist of combinations of (normalized) linear and angular velocities, and we re-label the original real control commands with the discrete action categories as above. Finally, we use classification accuracy as a metric to evaluate performance.

2D position prediction: The other evaluation task focuses on predicting the 2D position of the robot. We perform this task by predicting displacement $\Delta\hat{p}_i$ (in Cartesian coordinates) at each timestep, and accumulating all predicted displacements until timestep t to obtain the global position $\hat{p}_t$ given by:

$$\hat{p}_t = \sum_{0}^{t} \Delta\hat{p}_t. \tag{5.16}$$

We measure the Euclidean distance between the predicted position and the ground-truth position at the end of the trajectory as follows:

$$\text{error} = \|\hat{p}_N - p_N\|_2. \tag{5.17}$$

Note that the ground truth 2D positions are obtained from SLAM results. We ignore the z-axis (height) because the robot does not move along that axis.

Cross-validation was performed for all tasks. In each cross-validation run, we use 8 of 9 total trajectories to train and use the remaining trajectory for evaluation. The evaluation results are averaged across all 9 cross-validation runs.

Results

We compare our method with DT [35], and Behavior Cloning (BC). BC is implemented by (1) several convolutional layers (CNN) or (2) ViT [67] for the action prediction (imitation learning) tasks. For the position prediction task, we use the same CNN and ViT as those used in BC (labeled as CNN and ViT in Table 5.2).

Table 5.1: Continuous and discrete action spaces used for offline evaluation.

Action Space	Linear Velocity × Angular Velocity
Continuous	$[-1.0, 1.0] \times [-1.0, 1.0]$
6-action	$\{0, 1.0\} \times \{-1.0, 0, 1.0\}$
10-action	$\{0, 1.0\} \times \{-1.0, -0.5, 0, 0.5, 1.0\}$

Table 5.2: Offline Evaluation Results. 6-Discrete and 10-Discrete correspond to the discrete action spaces of 6 and 10 categories, respectively. We denote the evaluation metric used in each setting with ↑ / ↓ to note the preferred change. SRGB, FRGB-D, and FRGB represent for different input modalities. All listed results are are mean values among all cross-validation runs. Our method consistently outperforms the other methods in all tasks and input modalities.

Task		Metric	Method	Input		
				SRGB	FRGB-D	FRGB
Action Prediction	Continuous Action	MSE ↓	BC-CNN	0.149	0.164	0.170
			BC-ViT	0.137	0.167	0.168
			DT	0.132	0.146	0.150
			Ours	**0.124**	**0.136**	**0.142**
	6-action	Accuracy ↑	BC	58.6%	55.0%	55.4%
			BC-ViT	55.5%	49.5%	46.6%
			DT	80.6%	82.8%	82.1%
			Ours	**82.1%**	**84.0%**	**83.0%**
	10-action	Accuracy ↑	BC	39.8%	52.0%	48.2%
			BC-ViT	53.0%	46.9%	50.0%
			DT	73.6%	76.6%	75.7%
			Ours	**75.3%**	**77.0%**	**77.3%**
2-d Position Prediction		Error distance ↓	CNN	51.4	39.3	70.1
			ViT	82.5	130.5	109.9
			DT	42.9	38.7	36.8
			Ours	**32.1**	**22.8**	**23.2**

Table 5.2 presents the offline evaluation results on the collected human-following dataset. Among all tasks, our method and DT, which are sequential modeling (Transformer) methods, outperform naive behavior cloning. BC-ViT generally performs worse than BC-CNN, which indicates that the sole use of ViT [67] may be insufficient to tackle robotic tasks. From the first four rows of Table 5.2, we find that StARformer generally produced the highest performance in all action prediction tasks. This confirms that the proposed method can be generalized to real-world imitation learning problems. With regard to the 2D position prediction task, StARformer also outperforms BC methods and DT [35] by a large margin, showing that our model can be applied to other real-world prediction tasks as well.

Moreover, our method works well with all three visual modalities, demonstrating its generalizability. The panoramic input (SRGB) provides the best performance in the continuous action prediction task, as the target object is always included in the scene, which is essential for predicting fine-grained actions. Even under FRGB input, which presents a limited view, our method outperforms the best BC agent with SRGB input. In the discrete action space setting, FRGB-D input generally yields the best performance among all three methods (except for 10-action prediction by our method). For the position prediction task, FRGB-D input yields the best general performance. Inputs with the depth channel provide more informative cues for predicting displacements than pure RGB inputs. We also find that our model with SRGB outperforms other methods with FRGB-D input in the position prediction task, without relying on any depth information. These findings based on visual modalities can be useful for future research on vision-based robot learning using Transformers.

5.6.5 Online Real-world Evaluation

Evaluation Scenarios

We designed three scenarios (see Fig. 5.16) based on different trajectory shapes generated by the target's motion:

(a) moving in a "0" shape (Trajectory "0")

(b) moving in an "8" shape (Trajectory "8")

(c) moving randomly with interference from non-targets (Random + Interference)

In the first two scenarios, only the target person moves and the other people remain in place. In the third scenario, non-target people also move casually in the

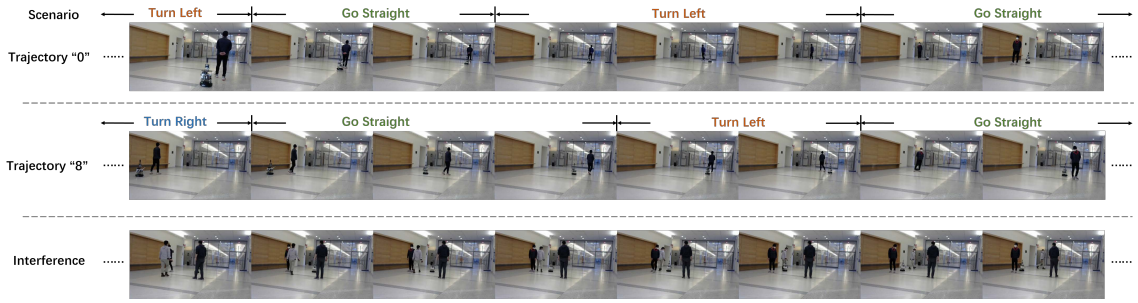


Figure 5.16: Samples of the three online evaluation scenarios. We present a chronological sequence of frames from left to right for each scenario in our experiment recording (shown in different rows). The difficulty of human-following increases from Trajectory "0" (Row 1) to Trajectory "8" (Row 2) and "Random + Interference" (Row 3). In the sample with interference, we included a non-target person moving in the robot's path.

same environment, occasionally interfering with the robot. Therefore, the difficulty increases accordingly from the first to the last scenario. Each scenario ends when the robot successfully follows the target for at least several minutes (5min, 3min, 5min for scenarios a, b, and c, respectively) or reaches a maximum of 10 trials (i.e., 10 failures). A failure is determined in real time by the experimenter when (1) the robot is about to bump into a wall, object, or other person; or (2) the robot deviates from the target person, with no indication of returning to the correct trajectory for approximate 2 seconds. We ensure fairness in the evaluation scenarios by following the same target person at the same site. The target person has an unseen appearance compared with the collected dataset. We use a panoramic image input and continuous action space in this experiment. Please check our supplementary video for a better understanding.

Evaluation Metrics

We devise two evaluation metrics for the human-following task.

Average Following Time to Failure (AFTF) : First, we measure the Average Following Time to Failure (AFTF), or the averaged duration of each trial prior to any failure[4].

Success Rate vs. Angular Error : Second, we construct a success rate-angular error curve, showing the success rate the robot gains under a certain angular

[4]The definition of a failure here in AFTF metric is the same as in Sec. 5.6.5.

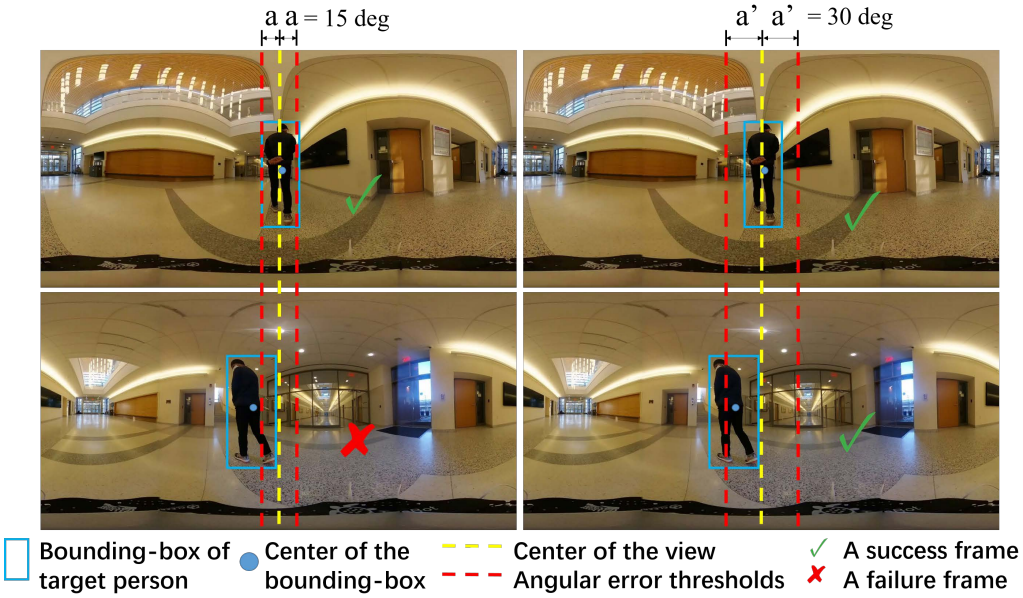


Figure 5.17: Definition of success/failure in success rate vs. angular error metrics. We show two frames in different angular error thresholds (15 and 30 degrees, for example). In the left column, with a 15-deg angular error, the upper frame suggests a success because the center of the bounding-box is within the threshold, whereas the lower frame indicates a failure. By increasing the error threshold from 15 deg to 30 deg, as shown in in the right column, both frames represent success. Note that bounding-boxes of the target person are extracted after the experiment by conventional tracking algorithms, for evaluation only.

Table 5.3: Online evaluation results measured by Average Following Time to Failure (AFTF) and the number of trials (defined in Sec. 5.6.5). We skip BC-CNN in the third scenario (random + interference), as BC fails easily and the trajectories become very short. Under our method, the robot successfully follows the person in the "0" and "8" trajectories without any failure, and gains longer AFTF compared to DT and BC in the relatively-harder "Random + Interference" scenario. Please refer to our supplementary video for better understanding.

Target Trajectory	Method	trials ↓	AFTF (s) ↑
Trajectory "0"	BC	3	101.6
	DT	**1**	**300.0**
	StAR	**1**	**300.0**
Trajectory "8"	BC	10	18.1
	DT	3	62.0
	StAR	**1**	**180.0**
Random + Interference	BC	-	-
	DT	6	53.3
	StAR	**3**	**109.3**

error threshold. More specifically, we define success/failure as the target person appearing within/falling out of a certain angular range (i.e., angular error) with respect to the center of the robot view (see Fig. 5.17. A higher success rate at a lower angular error suggests better stability in the human-following task, which means that the robot camera is continuously centered on the target. This metric is calculated using the position of the target person (center of the bounding box) extracted by a conventional tracking algorithm on panoramic images. We do not consider the distance between the robot and the target in this evaluation since our tasks do not consider a distance requirement in following.

Results

We acquire quantitative evaluation results (presented in Table 5.3 and Fig. 5.18) by comparing StARformer (StAR), DT [35], and behavior cloning (BC). We use BC-CNN here because BC-ViT is found to hard to generalized in this real world scenario. Table 5.3 lists the AFTF and the number of trials for all methods. Both Transformer-based methods outperform BC by a large margin in all scenarios. Compared with DT, our method shows better performance in the following complex patterns. In

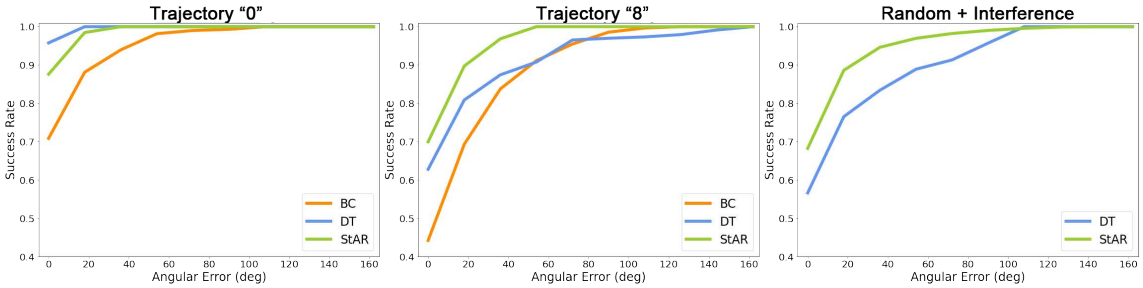


Figure 5.18: Success Rate vs. Angular Error curve of online evaluation. Because BC fails easily in the third scenario, it is omitted in the figure. Our method (StAR) shows significantly higher performance compared to DT in two of the relatively-challenging scenarios (“Trajectory 8” and “Random + Interference”), thus indicating a better stability.

the hardest scenario, where noise and randomness make the following much more challenging, both DT and our method show shorter AFTF and a higher number of trials. However, our method is more robust, with 50% fewer trials and 105% longer AFTF than DT. We also visualize the Success Rate vs. Angular Error curve, as shown in Fig. 5.18. Our method shows significantly higher success rates at lower angular-error thresholds, in two of the relatively-challenging scenarios among all algorithms, demonstrating its stability when following complex trajectories.

Fig 5.19 shows real trajectories generated from experiments on Trajectories “0” and “8”. We find that for Trajectory “0” scenario, BC fails to follow the target’s turning, whereas both DT and our method generate smooth trajectories. For Trajectory “8” scenario, BC fails more frequently and barely follows the target, as the trajectory includes sharper turns. In contrast, our method follows the target without any failure, whereas DT fails twice. In the “Random + Interference” scenario, we observe that our method can follow the original target for a longer time than the baseline methods. For instance, when a person crosses the target’s patch, causing occlusion and confusion, the robot continues to successfully follows the target. For more details concerning the “Random + Interference” scenario, please refer to our supplementary video.

5.7 Conclusion

This study introduces StARformer, which explicitly models strong local relations (Step Transformer) to facilitate the long-term sequence modeling (Sequence Trans-

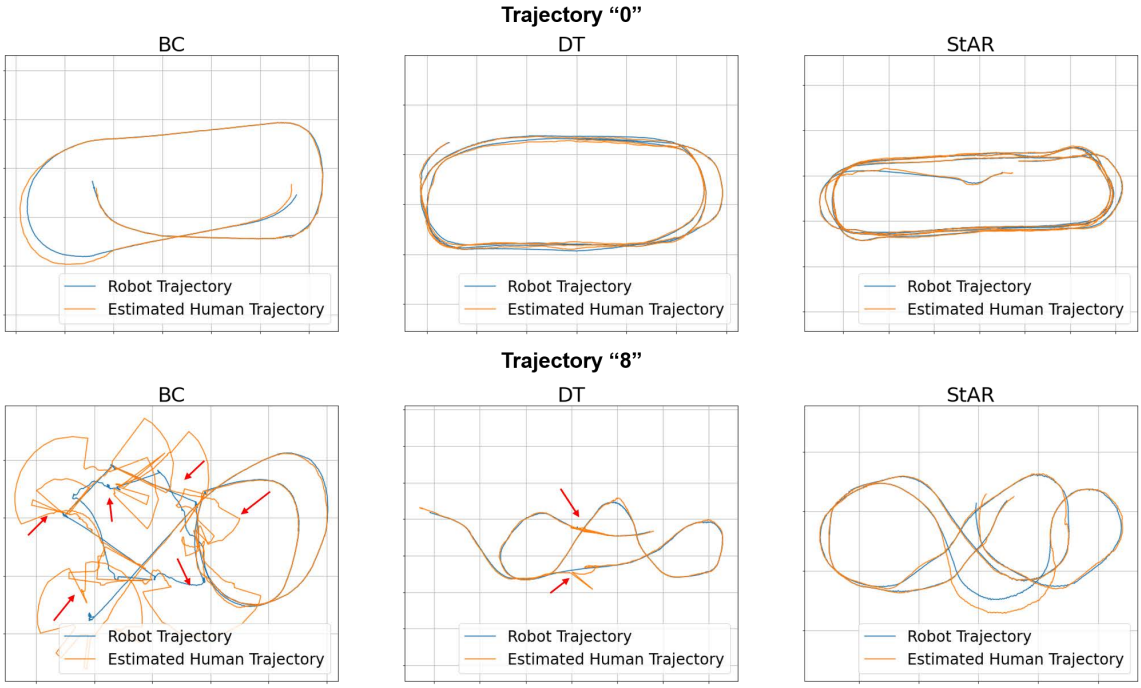


Figure 5.19: Real trajectory samples (for Trajectories “0” and “8”) from online evaluation. Robot trajectories are from SLAM. Human trajectories are estimated from robot trajectories, point clouds, and images using a conventional tracking algorithm after the experiments. Red arrows indicate where the robot fails to follow the target. When the robot is successfully following the target, the estimated human trajectories are relatively consistent with the recorded robot trajectory. In the event of failure, the estimated human trajectories deviate from the robot trajectories due to (1) target being out of view or (2) our manual reset. We find our method to have the highest success rate. We present the results as movie clips in our supplementary video to improve understanding. This supplementary video also includes the results from the “random + interference” scenario, which are hard to demonstrate in static images.

former) in Visual RL. Our extensive empirical results show how the learned StAR-representations help our model to outperform the baseline in both Atari and DMC environments, as well as both offline RL and imitation learning settings. We find that the fusion of learned StAR-representations and convolution features benefits action prediction. We further demonstrate that our designed architecture and token embeddings are essential to successfully model trajectories, with an emphasis on long sequences. We also verify that our method can be applied to real-world robot learning settings via a human-following experiment.

Chapter 6

Active Vision Reinforcement Learning under Limited Visual Observability

6.1 Overview

Although Reinforcement Learning (RL) has demonstrated success across challenging tasks and games in both simulated and real environments [5, 19, 21, 223, 246], the observation spaces for visual RL tasks are typically predefined to offer the most advantageous views based on prior knowledge and can not be actively adjusted by the agent itself. For instance, table-top robot manipulators often utilize a fixed overhead camera view [140]. While such fixed viewpoints can potentially stabilize the training of an image feature encoder [32], this form of perception is different from humans who actively adjust their perception system to finish the task, e.g. eye movements [65]. The absence of active visual perception poses challenges on learning in highly dynamic environments [160, 162], open-world tasks [78] and partially observable environments with occlusions, limited field-of-views, or multiple view angles [109, 212].

We study Active Reinforcement Learning (Active-RL) [260], the RL process that allows the embodied agent to actively acquire new perceptual information in contrast to the standard RL, where the new information could be reward signals [6, 55, 74, 135, 156, 165], visual observations [87, 91], and other forms. Specifically, we are interested in visual Active-RL tasks, i.e. ActiveVision-RL, that an agent controls its own views of visual observation, in an environment with limited visual observability [91]. Therefore, the goal of ActiveVision-RL is to learn two policies that still maximize the task return: the *motor policy* to finish the task and the *sensory policy* to control the observation.

ActiveVision-RL tasks present a considerable challenge due to the coordination between motor and sensory policies, given their mutual influence [260]. The motor policy requires clear visual observation for decision-making, while the sensory policy should adapt accordingly to the motor action. Depending on the sensory policy, transitioning to a new view could either aid or hinder the motor policy learning [32, 109, 260]. One notable impediment is the perceptual aliasing mentioned by Whitehead and Ballard [260]. An optimal strategy for sensory policy should be incorporating crucial visual information while eliminating any distractions. In the real world, humans disentangle their sensory actions, such as eye movements, from their motor actions, such as manipulation, and subsequently learn to coordinate them [160, 162]. Despite being modeled separately, these two action policies and the coordination can be learned jointly through the interaction during sensorimotor stage [70, 182, 185, 261].

Taking inspiration from human capabilities, we propose SUGARL: Sensorimotor Understanding Guided Active Reinforcement Learning, an Active-RL framework designed to jointly learn sensory and motor policies by maximizing extra intrinsic sensorimotor reward together with environmental reward. We model the ActiveVision-RL agent with separate sensory and motor policies by extending existing RL algorithms with two policy/value branches. Inspired by sensorimotor stage [182, 185, 261], we use the intrinsic sensorimotor reward to guide the joint learning of two policies, imposing penalties on the agent for selecting suboptimal observations. We use a learned sensorimotor reward module to assign the intrinsic reward. The module is trained using inverse dynamics prediction task [144, 241], with the same experiences as policy learning without additional data or pre-training.

In our experiments, we use modified Atari [19] and DeepMind Control suite (DMC) [246] with limited observability to comprehensively evaluate our proposed method. We also test on Robosuite tasks to demonstrate the effectiveness of active agent in 3D manipulation. Through the challenging benchmarks, we experimentally show that SUGARL is an effective and generic approach for Active-RL with minimum modification on top of existed RL algorithms. The learned sensory policy also exhibit active vision skills by analogy with humans' fixation and tracking.

6.2 Active Vision Reinforcement Learning Settings

Consider a vanilla RL setting based on a Markov Decision Process (MDP) described by $(\mathcal{S}, \mathcal{A}, r, P, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, r is the reward

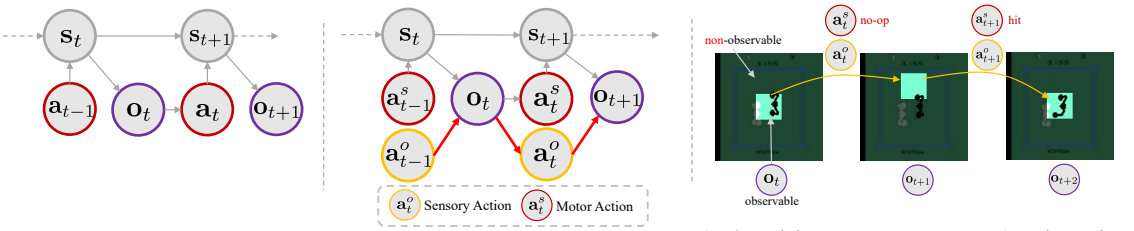


Figure 6.1: ActiveVision-RL with limited visual observability in comparison with standard RL, with exemplary process in Atari game *Boxing*. Red arrows stand for additional relationships considered in ActiveVision-RL. In the example on the right, the highlighted regions are the actual observations visible to the agent at each step. The rest of the pixels are not visible to the agent.

function, P describes state transition which is unknown and γ is the discount factor. In this work, we study ActiveVision-RL under limited visual observability, described by $(\mathcal{S}, \mathcal{O}, \mathcal{A}^s, \mathcal{A}^o, r, P, \gamma)$, as shown in Figure 6.1. $\mathcal{O}$ is the actual partial observation space the agent perceives. In particular, we are interested in visual tasks, so each observation $\mathbf{o}$ is an image contains partial information of an environmental state $\mathbf{s}$, like an image crop in 2D space or a photo from a viewpoint in 3D space. To emulate the human ability, there are two action spaces for the agent in Active-RL formulation. $\mathcal{A}^s$ is the motor action space that causes state change $p(\mathbf{s}'|\mathbf{s}, \mathbf{a}^s)$. $\mathcal{A}^o$ is the sensory action space that only changes the observation of an environmental state $p(\mathbf{o}|\mathbf{s}, \mathbf{a}^o)$. In this setting, the agent needs to take $(\mathbf{a}^s, \mathbf{a}^o)$ for each step, based on observation(s) only. An example is shown in Figure 6.1.

Our goal is to learn the motor and sensory action policies (π^s, π^o) that still maximize the return $\sum r_t$. Note that the agent is never exposed to the full environmental state $\mathbf{s}$. Both policies are completely based on the partial observations: $\mathbf{a}^s = \pi^s(\cdot|\mathbf{o})$, $\mathbf{a}^o = \pi^o(\cdot|\mathbf{o})$. Therefore the overall policy learning is challenging due to the limited information per step and the non-stationary observations.

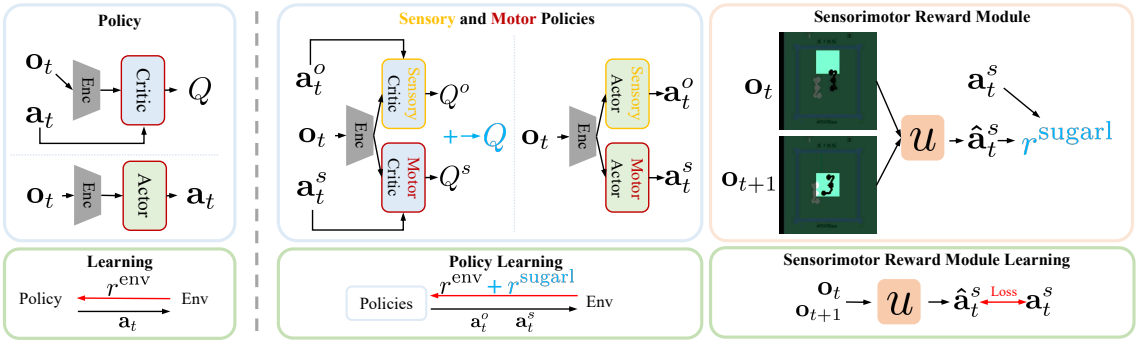


Figure 6.2: Overview of SUGARL and the comparison with original RL algorithm formulation. SUGARL introduces an extra sensory policy head, and jointly learns two policies together with the extra sensorimotor reward. We use the formulation of SAC [95] as an example. We introduce sensorimotor reward module to assign the reward. The reward indicates the quality of the sensory policy through the prediction task. The sensorimotor reward module is trained independently to the policies by action prediction error.

6.3 SUGARL: Sensorimotor Understanding Guided Active-RL

6.3.1 Active-RL Algorithm with Sensorimotor Understanding

We implement Active-RL algorithms based on the normal vision-based RL algorithms with simple modifications regarding separated motor and sensory policies (π^s, π^o), and the sensorimotor reward r^{sugarl}. We use DQN [170] and SAC [95] as examples to show the modifications are generally applicable. The example diagram of SAC is shown in Figure 6.2. We first introduce the policy then describe the sensorimotor reward in Section 6.3.2

Network Architecture The architectural modification is branching an additional head for sensory policy, and both policies share a visual encoder stem. For DQN [170], two heads output Q^s, Q^o for each policy respectively. This allows the algorithm to select $\mathbf{a}^s = \arg\max_{\mathbf{a}^s} Q^s$ and $\mathbf{a}^o = \arg\max_{\mathbf{a}^o} Q^o$ for each step. Similarly for SAC [95], the value and actor networks also have two separate heads for motor and sensory policies. The example in the form of SAC is in Figure 6.2.

Joint Learning of Motor and Sensory Policies Though two types of actions

are individually selected or sampled from network outputs, we find that the joint learning of two policies benefits the whole learning [110, 277]. The joint learning here means both policies are trained using a shared reward function. Otherwise, the sensory policy usually fails to learn with intrinsic reward signal only. Below we give the formal losses for DQN and SAC.

For DQN, we take the sum of Q-values from two policies $Q = Q^s + Q^o$ and train both heads jointly. The loss is the following where we indicate our modifications by blue:

$$\mathcal{L}_i^Q(\theta_i) = \mathbb{E}_{(\mathbf{o}_t, \mathbf{a}_t^s, \mathbf{a}_t^o)\sim\mathcal{D}}\left[\left(y_i - \left(Q^s_{\theta_i}(\mathbf{o}_t, \mathbf{a}_t^s) + Q^o_{\theta_i}(\mathbf{o}_t, \mathbf{a}_t^o)\right)\right)^2\right]$$

$$y_i = \mathbb{E}_{\mathbf{o}_{t+1}}\left[r_t^{\text{env}} + \beta r_t^{\text{sugarl}} + \gamma\left(\max_{\mathbf{a}^s_{t+1}} Q^s_{\theta_{i-1}}(\mathbf{o}_{t+1}, \mathbf{a}^s_{t+1}) + \max_{\mathbf{a}^o_{t+1}} Q^o_{\theta_{i-1}}(\mathbf{o}_{t+1}, \mathbf{a}^o_{t+1})\right)\right],$$

where L^Q is the loss for Q-networks, θ stands for the parameters of both heads and the encoder stem, i is the iteration, and $\mathcal{D}$ is the replay buffer. $\beta r_t^{\text{sugarl}}$ is the extra sensorimotor reward with balancing scale which will be described in Section 6.3.2.

For SAC, we do the joint learning similarly. The soft-Q loss is in the similar form of above DQN which is omitted for simplicity. The soft-value loss L^V and is

$$\mathcal{L}^V(\psi) = \mathbb{E}_{\mathbf{o}_t\sim\mathcal{D}}\Bigg[\frac{1}{2}\Big(V^s_\psi(o_t) + V^o_\psi(o_t) - \mathbb{E}_{\mathbf{a}_t^s\sim\pi^s_\phi, \mathbf{a}_t^o\sim\pi^o_\phi}\left[Q^s_\psi(\mathbf{o}_t, \mathbf{a}_t^s) + Q^o_\psi(\mathbf{o}_t, \mathbf{a}_t^o) - \log\pi^s_\phi(\mathbf{a}_t^s|o_t) - \log\pi^o_\phi(\mathbf{a}_t^o|o_t)\right]\Big)^2\Bigg],$$

and the actor loss $\mathcal{L}^\pi$ is

$$\mathcal{L}^\pi(\phi) = \mathbb{E}_{\mathbf{o}_t\sim\mathcal{D}}\left[\log\pi^s_\phi(\mathbf{a}_t^s|\mathbf{o}_t) + \log\pi^o_\phi(\mathbf{a}_t^o|\mathbf{o}_t) - Q^s_\psi(\mathbf{o}_t, \mathbf{a}_t^s) - Q^o_\psi(\mathbf{o}_t, \mathbf{a}_t^o)\right],$$

where ψ, ϕ are parameters for the critic and the actor respectively, and reparameterization is omitted for clearness.

6.3.2 Sensorimotor Reward

The motor and sensory policies are jointly trained using a shared reward function, which is the combination of environmental reward and our sensorimotor reward. We first introduce the assignment of sensorimotor reward and then describe the combination of two rewards.

The sensorimotor reward is assigned by the sensorimotor reward module $u_\xi(\cdot)$. The module is trained to have the sensorimotor understanding, and is used to

indicate the goodness of the sensory policy, tacking inspiration of human sensorimotor learning [185]. The way we obtain such reward module is similar to learning an inverse dynamics model [241]. Given a transition $(\mathbf{o}_t, \mathbf{a}^s_t, \mathbf{o}_{t+1})$, the module predicts the motor action $\mathbf{a}^s$ only, based on an observation transition tuple $(\mathbf{o}_t, \mathbf{o}_{t+1})$. When the module is (nearly) fully trained, the higher prediction error indicates the worse quality of visual observations. For example, if the agent is absent from observations, it is hard to infer the motor action. Such sub-optimal observations also confuse agent's motor policy. Since the sensory policy selects those visual observations, the quality of visual observations is tied to the sensory policy. As a result, we can employ the negative error of action prediction as the sensorimotor reward:

$$r^{\text{sugarl}}_t = -\Big(1 - p(\mathbf{a}^s_t|\mathbf{o}_t, \mathbf{o}_{t+1}; u_\xi)\Big). \tag{6.1}$$

This non-positive intrinsic reward penalizes the sensory policy for selecting sub-optimal views that do not contribute to the accuracy of action prediction and confuse motor policy learning. In Section 6.5.4 we show that naive positive rewarding does not guide the policy well. Note that the reward is less noisy when the module is fully trained. However, it is not harmful for being noisy at the early stage of training, as the noisy signal may encourage the exploration of policies. We use the sensorimotor reward though the whole learning.

The sensorimotor reward module is implemented by an independent neural network. The loss is a simple prediction error:

$$\mathcal{L}^u(\xi) = \mathbb{E}_{\mathbf{o}_t, \mathbf{o}_{t+1}, \mathbf{a}^s_t \sim \mathcal{D}}\Big[\text{Error}\Big(\mathbf{a}^s_t - u_\xi(\mathbf{o}_t, \mathbf{o}_{t+1})\Big)\Big], \tag{6.2}$$

where the Error$(\cdot)$ can be a cross-entropy loss for discrete action space or L2 loss for continuous action space. Though being implemented and optimized separately, the sensorimotor reward module uses the same experience data as policy learning, with no extra data or prior knowledge introduced.

Combining Sensorimotor Reward and Balancing The sensorimotor reward r^{sugarl} is added densely on a per-step basis, on top of the environmental reward r^{env} in a balanced form:

$$r_t = r^{\text{env}}_t + \beta r^{\text{sugarl}}_t, \tag{6.3}$$

where β is the balance parameter varies across environments. The reward balance is very important to make both motor and sensory policies work as expected, without heavy bias towards one side [76], which will be discussed in our ablation study in Section 6.5.4. Following the studies in learning with intrinsic rewards and rewards of many magnitudes [48, 103, 196, 245, 250], we empirically set

$\beta = \mathbb{E}_\tau[\sum_{t=1}^{T} r_t^{\text{env}}/T]$, which is the average environmental return normalized by the length of the trajectory. We get these referenced return data from the baseline agents trained on normal, fully observable environments, or from the maximum possible environmental return of one episode.

6.3.3 Persistence-of-Vision Memory

To address ActiveVision-RL more effectively, we introduce a Persistence-of-Vision Memory (PVM) to spatio-temporally combine multiple recent partial observations into one, mimicking the nature of human eyes and the memory. PVM aims to expand the effective observable area, even though some visual observations may become outdated or be superseded by more recent observations. PVM stores the observations from B past steps in a buffer and combines them into a single PVM observation according to there spatial positions. This PVM observation subsequently replaces the original observation at each step:

$$\text{PVM}(\mathbf{o}_t) = f(\mathbf{o}_{t-B+1}, \ldots, \mathbf{o}_t),$$

where $f(\cdot)$ is a combination operation. In the context of ActiveVision-RL, it is reasonable to assume that the agent possesses knowledge of its focus point, as it maintains the control over the view. Therefore, the viewpoint or position information can be used. In our implementations of $f(\cdot)$ shown in Figure 6.3, we show a 2D case of PVM using stitching, and a PVM using LSTM which can be used in both 2D and 3D environments. In stitching PVM, the partial observations are combined like a Jiasaw puzzle. It is worth noting that combination function f can be implemented by other pooling operations, sequence representations [35, 117, 215, 216], neural memories [99, 108, 199, 203], or neural 3D representations [191, 268], depending on the input modalities, tasks, and approaches.

Stitching makes PVM different from frame stacking [170]. PVM may not retain all the information, and the compression varies depending on the view selection.

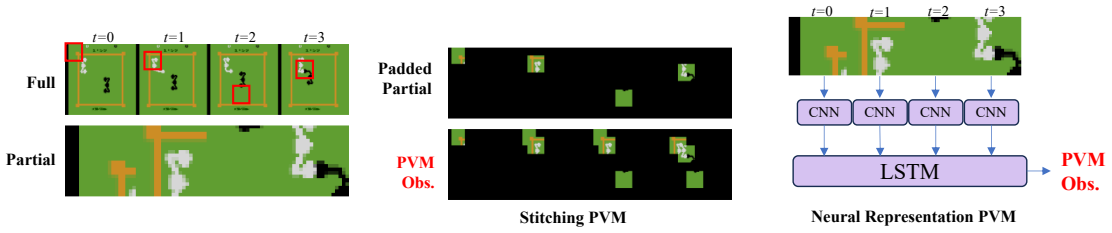


Figure 6.3: Examples for different instantiations of PVM with $B = 3$.

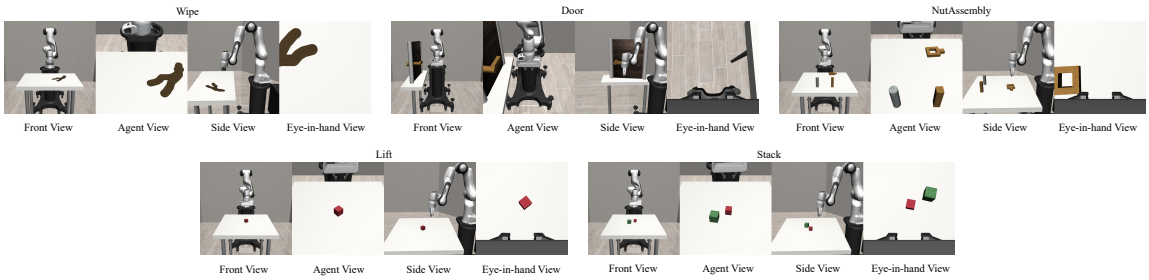


Figure 6.4: Five selected Robosuite tasks with examples on four hand-coded views.

In the given 2D example, ideally PVM combines three distinct crops that forms an effective range of Bhw measured in pixels. In a lower bound case, it combines crops from the same region and the effective observable area does not increase. PVM is light-weighted and dose not increase the RAM consumption of replay buffer by B times in contrast to frame stacking. PVM does not conflict with frame stacking and we use them together.

6.4 Environments and Settings

6.4.1 Active-Gym: An Environment for ActiveVision-RL

We present Active-Gym, an open-sourced customized environment wrapper designed to transform RL environments into ActiveVision-RL constructs. Our library currently supports active vision agent on Robosuite [284], a robot manipulation environment in 3D, as well as Atari games [19] and the DeepMind Control Suite (DMC) [246] offering 2D active vision cases. These 2D environments were chosen due to the availability of full observations for establishing baselines and upper bounds, and the availability to manipulate observability for systematic study.

6.4.2 Robosuite Settings

Observation Space In the Robosuite environment, the robot controls a movable camera. The image captured by that camera is a partial observation of the 3D space.

Action Spaces Each step in Active-Gym requires motor and sensory actions $(\mathbf{a}^s, \mathbf{a}^o)$. The motor action space $\mathcal{A}^s$ is the same as the base environment. The sensory action space $\mathcal{A}^o$ is a 5-DoF control: relative (x, y, z, yaw, pitch). The maximum

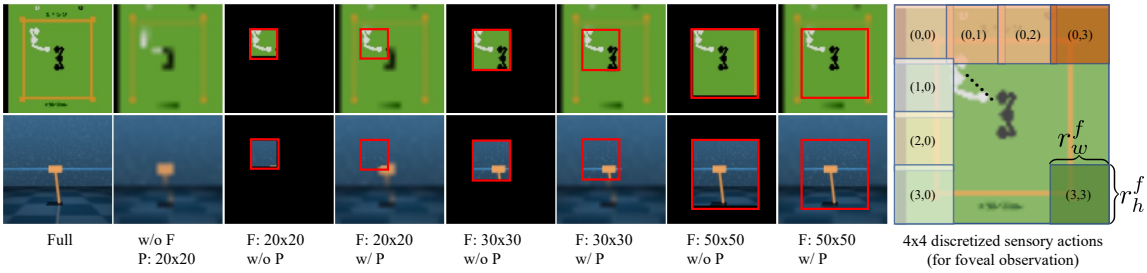


Figure 6.5: Left: observations from Active-Gym with different foveal resolutions (F) and peripheral settings (P) in Atari and DMC. The red bounding boxes show the observable area (foveal) for clarity. Right: 4x4 discrete sensory action options in our experiments.

linear and angular velocities are constrained to 0.01/step and 5 degrees/step, respectively.

6.4.3 2D Benchmark Settings

Observation Space In the 2D cases of Active-Gym, given a full observation $\mathbf{O}$ with dimensions (H, W), only a crop of it is given to the agent's input. Examples are highlighted in red boxes in Figure 6.5. The sensory action decides an observable area by a location (x, y), corresponding to the top-left corner of the bounding box, and the size of the bounding box (h, w). The pixels within the observable area becomes the foveal observation, defined as $\mathbf{o}^{\mathrm{f}} = \mathbf{o}^{\mathrm{c}} = \mathbf{O}[x : x + h, y : y + w]$. Optionally, the foveal observation can be interpolated to other resolutions $\mathbf{o}^f = \mathrm{Interp}(\mathbf{o}^{\mathrm{c}}; (h, w) \to (r_h^f, r_w^f))$, where (r_h^f, r_w^f) is the foveal resolution. This design allows for flexibility in altering the observable area size while keeping the effective foveal resolution constant. Typically we set $(r_h^f, r_w^f) = (h, w)$ and fixed them during a task. The peripheral observation can be optionally provided as well, obtained by interpolating the non-foveal part $\mathbf{o}^{\mathrm{p}} = \mathrm{Interp}(\mathbf{O} \setminus \mathbf{o}^{\mathrm{c}}; (H, W) \to (r_h^p, r_w^p))$, where (r_h^p, r_w^p) is the peripheral resolution. The examples are at the even columns of Figure 6.5. If the peripheral observation is not provided, $\mathbf{o}^p = 0$.

Action Spaces The sensory action space $\mathcal{A}^o$ includes all the possible (pixel) locations on the full observation, but can be further formulated to either continuous or discrete spaces according to specific task designs. In our experiments, we simplify the space by a 4x4 discrete grid-like anchors for (x, y) (Figure 6.5 right). Each anchor corresponds to the top-left corner of the observable area. The sensory

policy chooses to place the observable area among one of 16 anchors (**absolute** control), or moves it from one to the four neighbor locations (**relative** control).

6.4.4 Learning Settings

In our study, we primarily use DQN [170] and SAC [95] as the backbone algorithms of SUGARL to address tasks with discrete action spaces (Atari), and use DrQv2 [270] for continuous action spaces (DMC and Robosuite). All the visual encoders are standardized as the convolutional networks utilized in DQN [170]. To keep the network same, we resize all inputs to 84x84. For the sensorimotor understanding model, we employ the similar visual encoder architecture with a linear head to predict $\mathbf{a}_t^s$. Each agent is trained with one million transitions for each of the 26 Atari games, or trained with 0.1 million transitions for each of the 6 DMC tasks and 5 Robosuite tasks. The 26 games are selected following Atari-100k benchmark [125]. We report the results using Interquartile Mean (IQM), with the scores normalized by the IQM of the base DQN agent under full observation (except Robosuite), averaged across five seeds (three for Robosuite) and all games/tasks per benchmark. Details on architectures and hyperparameters can be found in the Appendix.

6.5 Results

6.5.1 Robosuite Results

We selected five of available tasks in Robosuite [284], namely block lifting (Lift), block stacking (Stack), nut assembling (NutAssembleSquare), door opening (Door), wiping the table (Wipe). The first two are easier compared to the later three. Example observations are available in Figure 6.4. We compare against a straightforward baseline that a single policy is learned to govern both motor and sensory actions. We also compare to baselines including RL with object detection (a replication of Cheng et al. [45]), learned attention [237], and standard RL with hand-coded views. Results are in 6.1. We confirm that our SUGARL works outperforms baselines all the time, and also outperforms the hand-coded views most of the time. Specifically, for the harder tasks including Wipe, Door, and NutAssemblySquare, SUGARL gets the best scores.

Designs of PVM We compare different instantiations of proposed PVMs including: Stacking: naively stacking multiple frames; LSTM: Each image is first encoded by CNN and fed into LSTM; 3D Transformation + LSTM: we use camera

Table 6.1: Results on Robosuite. We report the IQM of raw rewards from 30 evaluations. Highlighted task names are harder tasks. **Bold** numbers are the best scores of each task and underscored numbers are the second best.

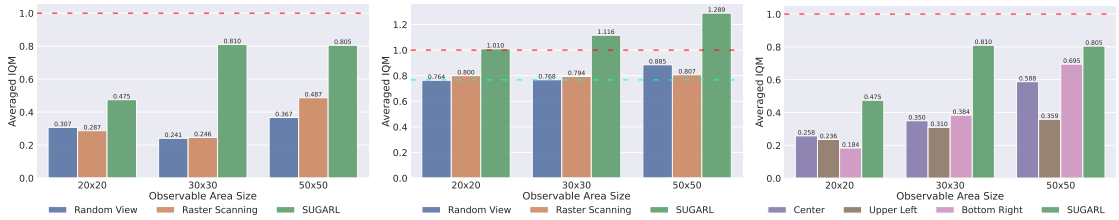
Approach	Wipe	Door	NutAssemblySquare	Lift	Stack
SUGARL-DrQ (Stacking PVM)	56.0	274.8	78.0	79.2	12.7
SUGARL-DrQ (LSTM PVM)	58.5	266.9	**108.6**	88.8	31.5
SUGARL-DrQ (3D Transformation+LSTM PVM)	**74.1**	**291.0**	65.2	87.5	32.4
SUGARL-DrQ w/o Joint Learning	43.6	175.4	58.0	107.2	12.0
SUGARL-DrQ w/o PVM	52.8	243.3	37.9	55.6	7.7
Single Policy	12.4	22.8	8.42	10.7	0.53
DrQ w/ Object Detection (DETR)	15.2	43.1	54.8	15.4	7.5
DrQ w/ End-to-End Attention	14.2	141.4	28.5	33.0	13.6
Eye-in-hand View (hand-coded, moving camera)	16.1	114.6	102.9	**233.9**	**73.0**
Front View (hand-coded, fixed camera)	49.4	240.6	39.6	69.0	13.8
Agent View (hand-coded, fixed camera)	12.7	190.3	49.9	122.6	14.7
Side View (hand-coded, fixed camera)	25.9	136.2	34.5	56.6	12.8

parameters to align pixels from different images to the current camera frame. Then an LSTM encodes the images after going through CNN. We find that 3D Transformation + LSTM works the best, because it tackles spatial aligning and temporal merging together. LSTM also works well in general.

6.5.2 2D Benchmark Results

We evaluate the policy learning on Atari under two primary visual settings: **with** and **without peripheral observation**. In each visual setting we explore three sizes of observable area (set equivalent to foveal resolution): **20x20**, **30x30**, and **50x50**. In with peripheral observation setting, the peripheral resolution is set to 20x20 for all tasks. We use DQN [170]-based SUGARL (SUGARL-DQN) and compare it against two variants by replacing the learnable sensory policy in SUGARL with: **Random View** and **Raster Scanning**. Random View always uniformly samples from all possible crops. Raster Scanning uses a pre-designed sensory policy that chooses observable areas from left to right and top to down sequentially. Raster Scanning yields relatively stable observation patterns and provides maximum information under PVM. We also provide a DQN baseline trained on full observations (84x84) as a soft oracle. In with peripheral observation settings, another baseline trained on peripheral observation only (20x20) is compared as a soft lower bound.

In Figure 6.6a and 6.6b, we find that SUGARL performs the best in all settings,



(a) Without peripheral obs. (b) With peripheral obs. (c) Static policies comparison

Figure 6.6: Results with different observation size and peripheral observation settings. The green bars are results from SUGARL. The red dashed lines stand for the DQN baseline trained on full observations using the same amount of data. We compare SUGARL against two dynamic views: Random View and Raster Scanning, and three static view baselines. In (b) with peripheral observation, we compare a baseline using peripheral observation only indicated by the cyan line.

showing that SUGARL learns effective sensory and motor policies jointly. More importantly, SUGARL with peripheral observation achieves higher overall scores (+0.01∼0.2) than the full observation baselines. In details, SUGARL gains higher scores than the full observation baseline in 13 out of 26 games with 50x50 foveal resolution (details are available in the Appendix). This finding suggests the untapped potential of ActiveVision-RL agents which leverage partial observations better than full observations. By actively selecting views, the agent can filter out extraneous information and concentrate on task-centric information.

Compare against Static Sensory Policies We also examine baselines with static sensory policies, which consistently select one region to observe throughout all steps. The advantage of this type baseline lies in the stability of observation. We select 3 regions: **Center**, **Upper Left**, and **Bottom Right**, and compare them against SUGARL in environment w/o peripheral observation. As shown in Figure 6.6c, we observe that SUGARL still surpasses all three static policies. The performance gaps between SUGARL and Center and Bottom Right are relatively small when the observation size is larger (50x50), as the most valuable information is typically found at these locations in Atari environments. Static observation is therefore

Table 6.2: Evaluation results on different conditions and algorithm backbones

(a) Action modeling

Model	20	30	50
SUGARL (abs)	**0.475**	**0.810**	0.805
SUGARL (rel)	0.367	0.745	**0.945**
Single Policy	0.132	0.222	0.171

(b) Train more steps

Steps	Model	20	30	50
1M	SUGARL	**0.475**	**0.810**	**0.805**
	Single Policy	0.132	0.222	0.171
5M	SUGARL	**1.170**	**1.121**	**1.553**
	Single Policy	0.332	0.640	1.145

(c) SUGARL with SAC

Model	20	30	50
SUGARL	**0.424**	**0.730**	**0.785**
SUGARL w/o r^{sugarl}	0.300	0.307	0.504
SAC-raster scanning	0.117	0.195	0.136
SAC-random view	0.155	0.104	0.134

(d) Different PVMs

Model	20	30	50
Stitching PVM	**0.475**	**0.815**	**0.810**
LSTM PVM	0.397	0.448	0.470

quite beneficial for decision-making. However, as the observation size decreases, an active policy becomes essential for identifying optimal observations, leading SUGARL to outperform others by a significant margin.

Sensory-Motor Action Spaces Modeling SUGARL models the sensory and motor action spaces separately inspired by the human abilities. An alternative approach is jointly modeling two action spaces $\mathcal{A} = \mathcal{A}^s \times \mathcal{A}^o$ and learning one single action policy, with environmental reward only (referred as **Single Policy**). We compare Single Policy and display the results in Table 6.2a, which reveal that modeling one large action space fails to learn the policy properly due to the expanded action space. Additionally, we examine two types of sensory action spaces: **absolute** (abs) and **relative** (rel). Absolute modeling allows the view to jump across the entire space while relative modeling restricts the view to moving only to the neighboring options at each step. Results in Table 6.2a indicate that the absolute modeling performs better in smaller observable regions, as it can select the desired view more quickly than relative modeling. In contrast, relative modeling demonstrates better performance in larger observable region setting as it produces more stable observations across steps.

Training More Steps We subsequently explore SUGARL performance when trained with more environmental steps, comparing outcomes between 1M and 5M steps. Results in Table 6.2b confirm that SUGARL continuous to improve with more steps and consistently outperforms the single policy baseline, suggesting that it does not merely stagnate at a trivial policy. However, due to the challenges posed by limited observability, the learning still proceeds slower than for the agent with full observations, which achieves the score 4.106 at 5M steps.

Generalization of SUGARL We apply SUGARL to Soft Actor-Critic (SAC) [95] and evaluate its performance on environments without peripheral observation. As before, we compare it with Random View, Raster Scanning, and SUGARL without intrinsic reward. According to Table 6.2c, we find that SUGARL-SAC outperforms the naive version without the guidance of r^{sugarl}, further emphasizing the significance of our method. Moreover, SUGARL-SAC also surpasses the random view and raster scanning policies. However, when employing max-entropy-based methods like SAC, it is necessary to reduce the weight of the entropy term to ensure consistency in the sensory policy. We will further discuss it in Section 6.5.3 based on the analysis of the learned sensory policy behavior.

DMC Tasks We also evaluate SUGARL on DMC [246] tasks based on DrQ [270], an improved version of DDPG [253]. We use six environments in DMC [246]: *ball_in_cup-catch*, *cartpole-swingup*, *cheetah-run*, *dog-fetch*, *fish-swim*, and *walker-walk*. Three foveal observation resolutions (20x20, 30x30, and 50x50) without

Table 6.3: SUGARL on DMC

Model	20	30	50
SUGARL-DrQ	**0.686**	**0.717**	**1.052**
DrQ-Single Policy	0.540	0.570	0.776
DrQ-Raster Scanning	0.609	0.566	0.913
DrQ-Random View	0.569	0.591	0.768
SUGARL-DrQ w/o PVM	0.672	0.620	0.930
SUGARL w/o Joint Learning	0.377	0.446	0.355

peripheral observation are explored. We use relative control for sensory action and decide the discrete sensory options by thresholding the continuous output. We compare our approach with baselines with joint modeling of sensory and motor action spaces (Single Policy), Raster Scanning, and Random View. Results are shown in Table 6.3. SUGARL outperforms all baselines in three foveal settings in DMC, showing it is also applicable to continuous control tasks.

6.5.3 Sensory Policy Analysis

Sensory Policy Patterns In Figure 6.7, we present examples of learned sensory policies from SUGARL-DQN in four Atari games, in settings w/o peripheral observations. These policies are visualized as heat maps based on the frequency of observed pixels. We discover that the sensory policies learn both **fixation** and **movement** (similar to tracking) behaviours [23, 275] depending on the specific task requirements. In the first two examples, the sensory policy tends to concentrate on fixed regions. In *battle_zone*, the policy learns to focus on the regions where enemies appear and the fixed front sight needed for accurate firing. By contrast, in highly dynamic environments like *boxing* and *freeway*, the sensory policies tend to observe broader areas in order to get timely observations. Though not being a perfect tracker, the sensory policy learns to track the agent or the object of interest in these two environments, demonstrating the learned capability akin to humans' Smooth Pursuit Movements. Recorded videos for entire episodes are available at our project page.

Sensory Policy Distribution We quantitatively assess the distributions of learned sensory policies. There are 16 sensory actions, i.e. the observable area options in our setup (Figure 6.5). We compare the distributions against uniform distribution using KL-divergence, across 26 games x 10 eval runs x 5 seeds. The

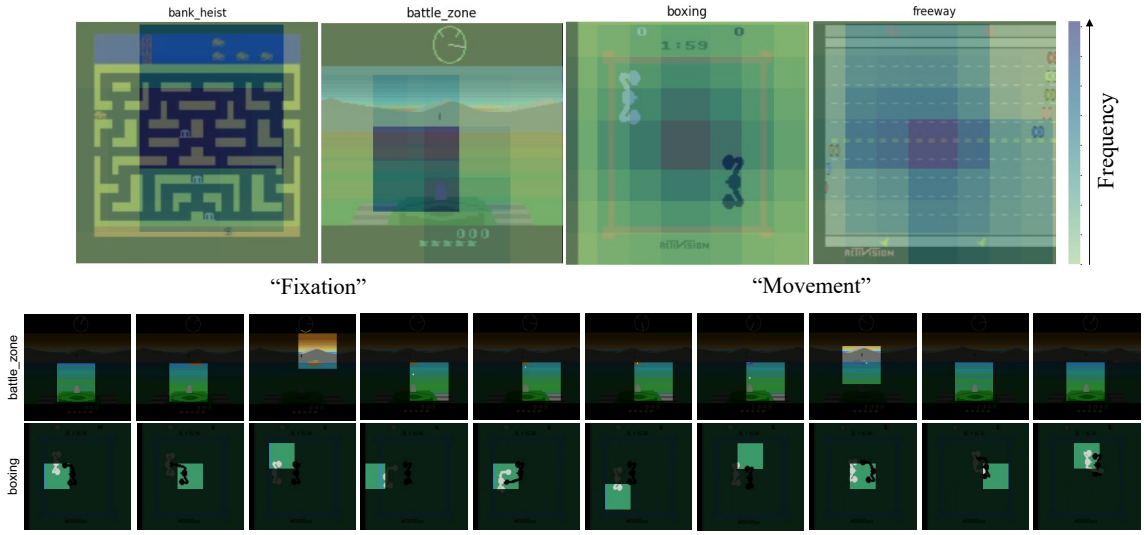


Figure 6.7: Examples of learned sensory policies from SUGARL-DQN. Top: frequency heat maps of pixels being observed. Bottom: sequences of observations showing fixation-like and tracking-like behaviors. More videos are available at this link

Table 6.4: Varing α of SUGARL-SAC.

Model	20	30	50
autotune α	0.271	0.358	0.444
fixed-$\alpha = 0.2$	**0.424**	**0.730**	**0.785**

resulting histogram are shown in Figure 6.8. We observe that the learned policies consistently deviate from the uniform distribution, suggesting that sensory policies prefer specific regions in general. The high peak at the high KL end supports the "fixation" behavior identified in the previous analysis. As the observation size increases, the divergence distribution shifts towards smaller KL end, while remaining > 0.5 for all policies. This trend indicates that with larger observable sizes, the policy does not need to adjust its attention frequently, corroborating the benefit of using relative control shown in Table 6.2a.

Pitfalls on Max-Entropy Based Methods In the previous analysis, we both qualitatively and quantitatively demonstrate that sensory policies are not uniformly dispersed across the entire sensory action space. This observation implies that

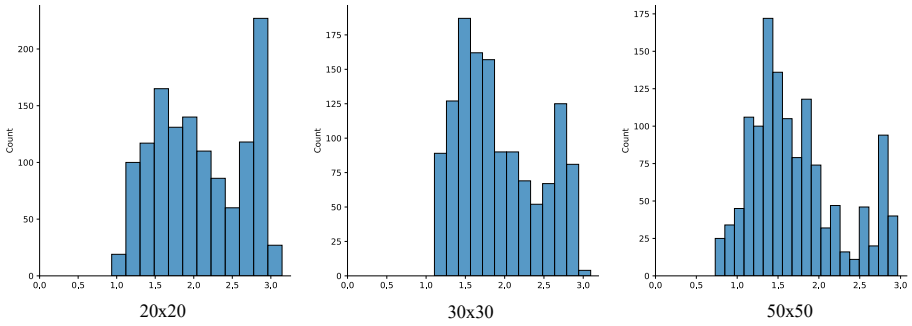


Figure 6.8: KL divergence distributions of learned visual policies from all 26 Atari games across 50 evaluation runs, in three different foveal observation sizes.

sensory policy should exercise caution when adopting the max-entropy-based methods. We conduct an experiment on varying the usage of α, the entropy coefficient in SAC in all three foveal observation size settings. Results in Table 6.4 show that simply disabling autotune and setting a small value to the sensory policy's α improves. This finding tells that max-entropy may not be a suitable assumption in modeling sensory policies.

6.5.4 Ablation Studies

We conduct ablation studies in the setting without peripheral observation and with 50x50 observation size. Five crucial design components are investigated to validate their effectiveness by incrementally adding or removing them from the full model. The results are presented in Table 6.5. From the results, we demonstrate the importance of *penalizing* the agent for inaccurate self-understanding predictions rather than rewarding accurate predictions ($r^{\text{sugarl}}(\text{positive}) \rightarrow r^{\text{sugarl}}(\text{negative})$). By imposing penalties, the maximum return is bounded by the original maximum possible return per episode, allowing motor and sensory policies to better coordinate each other and achieve optimal task performance. *Reward balance* significantly improves the policy, indicating its effectiveness in coordinating two policies as well. PVM also considerably enhances the algorithm by increasing the effective observable area as expected.

6.6 Limitations

In this work, we assume that completely independent sensory and motor actions are present in an embodied agent. But in a real-world case, the movement of the sensors may depend on the motor actions. For example, a fixed camera attached to the end-effector of a robot manipulator, or to a mobility robot. To address the potential dependence and conflicts between two policies in this case, extensions like voting or weighing across two actions to decide the final action may be required. The proposed algorithm also assumes a chance to adjust viewpoints at every step. This could be challenging for applications where the operational or latency costs for adjusting the sensors are high like remote control. To resolve this, additional penalties on sensory action and larger memorization capability are potentially needed. Last, the intrinsic reward currently only considers the accuracy of agent-centric prediction. Other incentives like gathering novel information or prediction accuracy over other objects in the environment can be further explored.

6.7 Conclusion

We present SUGARL, a framework based on existed RL algorithms to jointly learn sensory and motor policies through the ActiveVision-RL task. In SUGARL, an intrinsic reward determined by sensorimotor understanding effectively guides the learning of two policies. Our framework is validated in both 3D and 2D benchmarks with different visual observability settings. Through the analysis on the learned sensory policy, it shows impressive active vision skills by analogy with human's fixation and tracking that benefit the overall policy learning. Our work paves the initial way towards reinforcement learning using active agents for open-world tasks.

Table 6.5: Ablation study results based on DQN 50x50 foveal res w/o peripheral observation. The blank fields means there are no such modifications for that model.

	Model	r^{sugarl}	Joint Learning	PVM	Reward Balance (β)	IQM
	Random View			✓		0.367
SUGARL =	Base RL algorithm					-
	+Naive positive intrinsic reward	positive				0.281
	+Joint learning	positive	✓			0.322
	Positive → negative r^{sugarl}	negative	✓			0.360
	+PVM	negative	✓	✓		0.423
	+Reward Balance	negative	✓	✓	✓	**0.805**
	SUGARL w/o r^{sugarl}		✓	✓		0.421
	SUGARL w/o r^{sugarl} and w/o PVM		✓			0.231

Chapter 7

Conclusion and Future Work

This thesis proposal investigate several aspects of dealing with natural perceptions in embodied agent policy learning. Specifically, Chapter 3 introduces a feature disentangle method to align first-person view and third-person view observations together. The resulting viewpoint-agnostic state representations enable the third-person imitation learning problem.

In Chapter 4, we further extend the concept of viewpoint-agnostic representation to generic image and video understandings. We propose a plug-and-play Transformer plug-in that estimates the 3D coordinates of each pixel, to enrich the positional information in the representation. The outcome representation in the end exhibits viewpoint-agnostic patterns and improve image and video classification tasks.

Chapter 5 discusses StARformer, a sequence modeling approach to enhance action generation from past experience. We propose to use both short-term and long-term trajectory representation together to enhance the model capability. By doing so, StARformer gains better performance especially when the experience is long. We also test the method on a ground mobility robot and confirm its effectiveness.

In Chapter 6, we investigate a new problem setting of Active Vision Reinforcement Learning and propose our algorithm SUGARL. Instead of learning from pre-assigned views, Active Vision Reinforcement Learning allows the agent to actively adjust its view during the task. Inspired from human's ability, we design intrinsic reward to drive the sensory policy. SUGARL performs well in this ActiveVision-RL in general, and sometimes outperforms the standard RL baseline under static view or global observations.

By all these works together, we show the path of dealing with natural per-

ceptions in embodied agent policy learning: from static viewpoint agnostic representations, to an agent that is able to actively adjusting its viewpoint across time.

7.1 Future Work

We conclude this thesis proposal and discuss the future work. We propose to investigate how natural languages can be used together with visual perception and thus benefit action policies. Large Language Models (LLMs) [28, 64, 181, 244] or Vision-Language Models (VLMs) [143, 148, 194] are shown to have abundant commonsense knowledge about daily scenarios and activities. LLMs also have shown quantum performance leap on many embodied tasks, including open-world exploration in Minecraft [256], real-robot manipulation [4, 24, 26, 27, 69, 218, 276], and navigation [272].

One line of recent work is LLM-based task planning, where the user directly prompts the off-the-shelf LLMs with examples and task descriptions, and get LLM generated step-by-step plans in language space. Among all kinds of languages, programming languages becomes good proxies between LLM and actual robot control due to its unique structured but explainable nature [111, 146].

Despite the promising long-term planning capability, the generated plan simply follows the most straightforward way to complete task, which is not always optimal in terms of execution effort like distance and time cost. For example, a simple *for* loop may be returned if prompted to move a number of blocks, while the optimal way is to operate all of them in one batch. Also, moving a large group of items to another single object is harder than moving the object towards them. Motivated by these scenarios, we look into how commonsense knowledge in LLMs can be used to rewrite the LLM-generated plans to achieve better efficiency.

Specifically, we want to take advantage of equivalent operations for high-level planning optimizations. There are many sets of actions they can produce the same outcomes, by specifying different parameters for different action. For example, if there are two groups A and B of items mixed together and we want to separate. Separating A from B is exactly the same of separating B from A. We want to optimize the plan towards the case using less operations to achieve efficiency. For low-level planning, we want to use the LLM to remove redundant operations. In code generation style robot control [111, 146], LLM may use simple for loops to express the repeated action. However, there could be redundant actions in between like reset to default pose loop. We want to take advantage of LLM to unroll this

type of loop and make the plan smooth.

Overall, we propose our solution Policy Compiler to address issue in several steps:

- We plan to use High-level Compiler which is an LLM to re-write the high-level planning using equivalent operations. We give example demonstrations as prompts to LLM
- We plan to use Low-level Compiler which is also an LLM to remove redundant behaviors in the low-level planning.

The other line of work is the extension of Active Vision Reinforcement Learning using VLM. Instead of only using the inverse dynamics prediction accuracy, which is more focusing on self-awareness, there are more factors to describe to evaluate the goodness of the visual observation. One possible factor is how the current visual observation relates to the task, which we could name it task-awareness. A good visual observation should reflect what is needed for completing the task. For example, the target object should be within the field-of-view. We notice two aspects in the task-awareness: object existence and object motion. The object existence signals how the sensory policy should be driven to focus on the object, especially where the object could exist given a scene. The object motion additionally indicates how the sensory policy should cooperate with both motor policy as well as compensate the object motion, e.g. tracking. Though in our preliminary results in Atari and Robosuite, where a tracking-like behavior is shown, the model is not explicitly guided by this factor.

We propose to use VLM to summary the visual observation (history) into natural languages, and query LLMs to give a feedback on where should the sensory policy should be focusing on. The sensory policy and motor policies could be conditioned on natural languages, serving as intrinsic reward in an implicit form. We observe that such policies may need a pretraining to follow such natural languages, but it is feasible since it could be done in task-free style.

Overall, for this line of work, we propose:

- We plan to use VLM to summarize the visual observation (history) into natural languages.
- We plan to use LLM to provide language-based feedback based on summarized observation and task description.
- The sensory and action policies are pretrained to follow the conditioning language.

In conclusion, we propose Policy Compiler to improve LLM-generated robot plan for execution efficiency, and also propose VLM-based task-awareness indicator to improve Active Vision Reinforcement Learning.

Chapter 8

Bibliography

[1] R. Agarwal, D. Schuurmans, and M. Norouzi. An optimistic perspective on offline reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 104–114, July 2020. 60

[2] R. Agarwal, D. Schuurmans, and M. Norouzi. An optimistic perspective on offline reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 104–114. PMLR, 2020. 60, 61

[3] S. Agarwal, H. Arora, S. Anand, and C. Arora. Contextual diversity for active learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pages 137–153. Springer, 2020. 10

[4] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022. 5, 100

[5] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. 81

[6] R. Akrour, M. Schoenauer, and M. Sebag. April: Active preference learning-based reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II 23*, pages 116–131. Springer, 2012. 10, 81

[7] A. Andreopoulos and J. K. Tsotsos. A theory of active object localization. In *2009 IEEE 12th International Conference on Computer Vision*, pages 903–910. IEEE, 2009. 10

[8] A. Andreopoulos and J. K. Tsotsos. A computational learning theory of active object recognition under uncertainty. *International journal of computer vision*, 101:95–142, 2013. 10

[9] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 62

[10] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay, 2018. 19

[11] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Luvcić, and C. Schmid. ViViT: A video vision transformer. In *Proceedings of the International Conference on Computer Vision (ICCV)*, Oct. 2021. 8, 9, 52

[12] J. T. Ash, C. Zhang, A. Krishnamurthy, J. Langford, and A. Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*, 2020. 10

[13] N. Atanasov, B. Sankaran, J. Le Ny, G. J. Pappas, and K. Daniilidis. Nonmyopic view planning for active object classification and pose estimation. *IEEE Transactions on Robotics*, 30(5):1078–1090, 2014. 10

[14] Y. Aytar, L. Castrejon, C. Vondrick, H. Pirsiavash, and A. Torralba. Cross-modal scene networks. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2303–2314, 2017. 7

[15] Y. Aytar, T. Pfaff, D. Budden, T. L. Paine, Z. Wang, and N. de Freitas. Playing hard exploration games by watching youtube, 2018. 6, 19

[16] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization, 2016. arXiv:1607.06450. 52

[17] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos. Revisiting active perception. *Autonomous Robots*, 42:177–196, 2018. 10

[18] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, J. Tenenbaum, and B. Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32, 2019. 36

[19] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *J Artif Intell Res .*, 47(1):253–279, May 2013. 60, 81, 82, 88

[20] W. H. Beluch, T. Genewein, A. Nürnberger, and J. M. Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9368–9377, 2018. 10

[21] C. Berner, G. Brockman, B. Chan, V. Cheung, P. D**k**ebiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019. 81

[22] G. Bertasius, H. Wang, and L. Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2021. 8, 9, 45, 46, 53

[23] A. Borji and L. Itti. State-of-the-art in visual attention modeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):185–207, 2012. 94

[24] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, A. Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023. 5, 100

[25] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI gym, 2016. arXiv:1606.01540. 60

[26] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023. 2, 5, 100

[27] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. In *Proceedings of Robotics: Science and Systems*, 2023. 2, 5, 100

[28] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901, 2020. 2, 4, 100

[29] R. Burgert, J. Shang, X. Li, and M. Ryoo. Neural neural textures make sim2real consistent. *arXiv preprint arXiv:2206.13500*, 2022. 8, 36

[30] A.-Q. Cao and R. de Charette. Monoscene: Monocular 3d semantic scene completion. *arXiv preprint arXiv:2112.00726*, 2021. 8

[31] A. Casanova, P. O. Pinheiro, N. Rostamzadeh, and C. J. Pal. Reinforced active learning for image segmentation. *arXiv preprint arXiv:2002.06583*, 2020. 10

[32] E. Cetin, P. J. Ball, S. Roberts, and O. Celiktutan. Stabilizing off-policy deep reinforcement learning from pixels. In *International Conference on Machine Learning*, pages 2784–2810. PMLR, 2022. 81, 82

[33] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *The British Machine Vision Conference (BMVC)*, 2014. 7, 29

[34] C.-F. R. Chen et al. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 7, 30

[35] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, Dec. 2021. 1, 8, 9, 49, 53, 54, 55, 59, 60, 61, 62, 65, 72, 74, 77, 87

[36] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, P. Dhariwal, D. Luan, and I. Sutskever. Generative pretraining from pixels. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1691–1703, Jul. 2000. 9

[37] M. Chen, F. Wei, C. Li, and D. Cai. Frame-wise action representations for long videos via sequence contrastive learning. *arXiv preprint arXiv:2203.14957*, 2022. 8

[38] R. T. Q. Chen, X. Li, R. Grosse, and D. Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, 2018. 6, 16

[39] S. Chen, H. Cao, Q. Ouyang, X. Wu, and Q. Qian. Alds: An active learning method for multi-source materials data screening and materials design. *Materials & Design*, 223:111092, 2022. ISSN 0264-1275. 10

[40] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 7

[41] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020.

[42] X. Chen and K. He. Exploring simple siamese representation learning, 2020. 7, 17

[43] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: interpretable representation learning by information maximizing generative adversarial nets. In *Neural Information Processing Systems (NIPS)*, 2016. 7

[44] J. Cheng, L. Dong, and M. Lapata. Long short-term memory-networks for machine reading, 2016. arXiv:1601.06733. 52

[45] R. Cheng, A. Agarwal, and K. Fragkiadaki. Reinforcement learning of active vision for manipulating objects under occlusions. In *Conference on Robot Learning*, pages 422–431. PMLR, 2018. 90

[46] R. Cheng, Z. Wang, and K. Fragkiadaki. Geometry-aware recurrent neural networks for active visual recognition. *Advances in Neural Information Processing Systems*, 31, 2018. 10

[47] T.-C. Chi, M. Shen, M. Eric, S. Kim, and D. Hakkani-tur. Just ask: An interactive learning framework for vision and language navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2459–2466, 2020. 10

[48] J. Choi and S.-e. Yoon. Intrinsic motivation driven intuitive physics learning using deep reinforcement learning with intrinsic reward normalization. *arXiv preprint arXiv:1907.03116*, 2019. 86

[49] J. Choi, K. M. Yi, J. Kim, J. Choo, B. Kim, J. Chang, Y. Gwon, and H. J. Chang. Vab-al: Incorporating class imbalance and difficulty with variational bayes for active learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6749–6758, 2021. 10

[50] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, D. Belanger, L. Colwell, and A. Weller. Rethinking attention with performers. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Apr. 2020. 9

[51] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996. 10

[52] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019. 20, 36

[53] W. Dabney, M. Rowland, M. Bellemare, and R. Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 60, 61

[54] Z. Dai, H. Liu, Q. V. Le, and M. Tan. CoAtNet: Marrying convolution and attention for all data sizes. In *Advances in Neural Information Processing Systems (NeurIPS)*, Dec. 2021. 8, 10

[55] C. Daniel, M. Viering, J. Metz, O. Kroemer, and J. Peters. Active reward learning. In *Robotics: Science and systems*, volume 98, 2014. 10, 81

[56] S. Das and M. S. Ryoo. Viewclr: Learning self-supervised video representation for unseen viewpoints. *Arxiv*, 2112.03905, 2021. 29

[57] S. Das, R. Dai, M. Koperski, L. Minciullo, L. Garattoni, F. Bremond, and G. Francesca. Toyota smarthome: Real-world activities of daily living. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019. 45

[58] S. Das, S. Sharma, R. Dai, F. Bremond, and M. Thonnat. Vpn: Learning video-pose embedding for activities of daily living. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 72–90. Springer, 2020. 8

[59] S. Das, R. Dai, D. Yang, and F. Bremond. Vpn++: Rethinking video-pose embeddings for understanding activities of daily living. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 8

[60] S. Dasari and A. Gupta. Transformers for one-shot visual imitation. In *Conference on Robot Learning (CoRL)*, 2020. 9

[61] T. De Bruin, J. Kober, K. Tuyls, and R. Babu**v**ska. Integrating state representation learning into deep reinforcement learning. *IEEE Robotics and Automation Letters*, 3(3):1394–1401, 2018. 1

[62] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. 35

[63] E. Denton and V. Birodkar. Unsupervised learning of disentangled representations from video. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4417–4426, 2017. 7

[64] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding, 2019. arXiv:1810.04805. 2, 5, 9, 52, 100

[65] R. Dodge. Five types of eye movement in the horizontal meridian plane of the field of regard. *American journal of physiology-legacy content*, 8(4): 307–329, 1903. 81

[66] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015. 7

[67] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Apr. 2020. 9, 51, 52, 54, 55, 65, 72, 74

[68] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 7, 29, 30, 32, 34, 35, 45

[69] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023. 5, 100

[70] P. W. Duncan, M. Propst, and S. G. Nelson. Reliability of the fugl-meyer assessment of sensorimotor recovery following cerebrovascular accident. *Physical therapy*, 63(10):1606–1610, 1983. 82

[71] D. Dwibedi, J. Tompson, C. Lynch, and P. Sermanet. Learning actionable representations from visual observations. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1577–1584. IEEE, 2018. 6, 11, 39

[72] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. Temporal cycle-consistency learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1801–1810, 2019. 8, 36, 37

[73] A. D. Edwards, H. Sahni, Y. Schroecker, and C. L. Isbell. Imitating latent policies from observation. *arXiv preprint arXiv:1805.07914*, 2018. 6

[74] A. Epshteyn, A. Vogel, and G. DeJong. Active reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 296–303, 2008. 10, 81

[75] B. Eysenbach, X. Geng, S. Levine, and R. Salakhutdinov. Rewriting history with inverse rl: Hindsight inference for policy improvement. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 62

[76] L. Fan and Y. Wu. Avoiding lingering in learning active recognition by adversarial disturbance. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4612–4621, January 2023. 10, 86

[77] L. Fan, P. Xiong, W. Wei, and Y. Wu. Flar: A unified prototype framework for few-sample lifelong active recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15394–15403, October 2021. 10

[78] L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D.-A. Huang, Y. Zhu, and A. Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Advances in Neural Information Processing Systems*, volume 35, pages 18343–18362. Curran Associates, Inc., 2022. 11, 81

[79] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021. arXiv:2004.07219. 60

[80] S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 60, 61

[81] H. Furuta, Y. Matsuo, and S. S. Gu. Distributional decision transformer for hindsight information matching. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 9

[82] Y. Gal, R. Islam, and Z. Ghahramani. Deep bayesian active learning with image data. In *International conference on machine learning*, pages 1183–1192. PMLR, 2017. 10

[83] J. Gao, M. Chen, and C. Xu. Fine-grained temporal contrastive learning for weakly-supervised temporal action localization. *arXiv preprint arXiv:2203.16800*, 2022. 8

[84] J. J. Georgia Gkioxari, Jitendra Malik. Mesh r-cnn. *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019. 29

[85] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, et al. Challenges in representation learning: A report on three machine learning contests. In *Neural Information Processing: 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part III 20*, pages 117–124. Springer, 2013. 1

[86] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 6

[87] R. Göransson. Deep reinforcement learning with active vision on atari environments. In *Master's Thesis*. Lund University, 2022. 10, 11, 81

[88] J. C. Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, 1975. 41

[89] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou, and M. Douze. Levit: a vision transformer in convnet's clothing for faster inference. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 12259–12269, 2021. 8

[90] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020. 7, 17

[91] M. K. Grimes, J. V. Modayil, P. W. Mirowski, D. Rao, and R. Hadsell. Learning to look by self-prediction. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. 10, 11, 81

[92] S. S. Guo, R. Zhang, B. Liu, Y. Zhu, D. Ballard, M. Hayhoe, and P. Stone. Machine versus human attention in deep reinforcement learning tasks. *Advances in Neural Information Processing Systems*, 34:25370–25385, 2021. 10

[93] W. H. Guss, B. Houghton, N. Topin, P. Wang, C. Codel, M. Veloso, and R. Salakhutdinov. MineRL: A large-scale dataset of Minecraft demonstrations. *Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019. URL `http://minerl.io`. 11, 20

[94] 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 International Conference on Machine Learning (ICML), pages 1861–1870, Jul. 2018. 60

[95] 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 International Conference on Machine Learning (ICML)*, 2018. 1, 84, 90, 93

[96] I. Hadji, K. G. Derpanis, and A. D. Jepson. Representation learning via global temporal alignment and cycle-consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11068–11077, 2021. 36

[97] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang. Transformer in transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, Dec. 2021. 7, 9, 29, 43

[98] A. Haque, B. Peng, Z. Luo, A. Alahi, S. Yeung, and L. Fei-Fei. Towards viewpoint invariant 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 160–177. Springer, 2016. 8

[99] M. J. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2015. 87

[100] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, Jun. 2016. 52

[101] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 7, 29

[102] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020. doi: 10.1109/cvpr42600.2020.00975. URL http://dx.doi.org/10.1109/cvpr42600.2020.00975. 7

[103] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 86

[104] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017. 7, 17

[105] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017. 16

[106] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, Dec. 2016. 1

[107] J. Ho and S. Ermon. Generative adversarial imitation learning. *CoRR*, abs/1606.03476, 2016. URL http://arxiv.org/abs/1606.03476. 6, 7

[108] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 87

[109] K. Hsu, M. J. Kim, R. Rafailov, J. Wu, and C. Finn. Vision-based manipulators need to also see from their hands. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 1, 8, 11, 81, 82

[110] W. Huang, I. Mordatch, and D. Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pages 4455–4464. PMLR, 2020. 85

[111] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In *Conference on Robot Learning (CoRL)*, 2023. 2, 5, 100

[112] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017. 12

[113] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 28, 2015. 8

[114] S. James and A. J. Davison. Q-attention: Enabling efficient learning for vision-based robotic manipulation. *IEEE Robotics and Automation Letters*, 7 (2):1612–1619, 2022. 10

[115] S. James, K. Wada, T. Laidlow, and A. J. Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13739–13748, 2022. 10

[116] R. Jangir, N. Hansen, S. Ghosal, M. Jain, and X. Wang. Look closer: Bridging egocentric and third-person views with transformers for robotic manipulation. *IEEE Robotics and Automation Letters (RA-L)*, 2022. 8

[117] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems (NeurIPS)*, Dec. 2021. 1, 8, 49, 51, 53, 59, 60, 62, 87

[118] D. Jayaraman and K. Grauman. Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pages 489–505. Springer, 2016. 10

[119] Z.-H. Jiang, Q. Wu, K. Chen, and J. Zhang. Disentangled representation learning for 3d face shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11957–11966, 2019. 7

[120] L. Jin, S. Qian, A. Owens, and D. F. Fouhey. Planar surface reconstruction from sparse views. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12991–13000, 2021. 41

[121] M. Johnson, K. Hofmann, T. Hutton, and D. Bignell. The malmo platform for artificial intelligence experimentation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 4246–4247. AAAI Press, 2016. ISBN 9781577357704. 20

[122] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In *2009 ieee conference on computer vision and pattern recognition*, pages 2372–2379. IEEE, 2009. 10

[123] L. P. Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1993. 62

[124] K. Kahatapitiya and M. S. Ryoo. Swat: Spatial structure within and among tokens. *arXiv preprint arXiv:2111.13677*, 2021. 7

[125] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, A. Mohiuddin, R. Sepassi, G. Tucker, and H. Michalewski. Model based reinforcement learning for atari. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 90

[126] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. doi: 10.1109/cvpr.2019.00453. URL http://dx.doi.org/10.1109/CVPR.2019.00453. 7, 16

[127] S. Kasaei, J. Sock, L. S. Lopes, A. M. Tomé, and T.-K. Kim. Perceiving, learning, and recognizing 3d objects: An approach to cognitive service robots. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 10

[128] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2): 81–93, 1938. 37

[129] T. Kim, I. Hwang, H. Lee, H. Kim, W.-S. Choi, J. J. Lim, and B.-T. Zhang. Message passing adaptive resonance theory for online active semi-supervised learning. In *International Conference on Machine Learning*, pages 5519–5529. PMLR, 2021. 10

[130] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 6

[131] N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The efficient transformer. In *Proceedings of the International Conference on Learning Representations (ICLR)*, May 2019. 9

[132] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, Dec. 2000. 1, 8

[133] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 60, 61

[134] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009. 35

[135] D. Krueger, J. Leike, O. Evans, and J. Salvatier. Active reinforcement learning: Observing rewards at a cost. *arXiv preprint arXiv:2011.06709*, 2020. 10, 81

[136] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019. 60, 61

[137] A. Kumar, X. B. Peng, and S. Levine. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*, 2019. 62

[138] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:1179–1191, 2020. 60, 61

[139] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995. 7

[140] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018. 81

[141] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020. arXiv:2005.01643. 59

[142] A. C. Li, L. Pinto, and P. Abbeel. Generalized hindsight for reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 62

[143] J. Li, D. Li, C. Xiong, and S. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 12888–12900. PMLR, 2022. 2, 5, 100

[144] X. Li, J. Shang, S. Das, and M. S. Ryoo. Does self-supervised learning really improve reinforcement learning from pixels? *arXiv preprint arXiv:2206.05266*, 2022. 82

[145] Y. Li, S. Li, V. Sitzmann, P. Agrawal, and A. Torralba. 3d neural scene representations for visuomotor control. In *Conference on Robot Learning*, pages 112–123. PMLR, 2022. 11

[146] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023. 2, 5, 100

[147] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding, 2017. arXiv:1703.03130. 52

[148] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 2, 5, 100

[149] L. Liu, S. Fryc, L. Wu, T. L. Vu, G. Paul, and T. Vidal-Calleja. Active and interactive mapping with dynamic gaussian process implicit surfaces for mobile manipulators. *IEEE Robotics and Automation Letters*, 6(2):3679–3686, 2021. 10

[150] Y. Liu, A. Gupta, P. Abbeel, and S. Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018. doi: 10.1109/icra.2018.8462901. URL http://dx.doi.org/10.1109/ICRA.2018.8462901. 6

[151] Z. Liu, H. Ding, H. Zhong, W. Li, J. Dai, and C. He. Influence selection for active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9274–9283, 2021. 10

[152] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 10012–10022, Oct. 2021. 7, 9, 30, 43

[153] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021. 8

[154] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu. Video swin transformer, 2021. arXiv:2106.13230. 9

[155] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022. 8

[156] M. Lopes, F. Melo, and L. Montesano. Active learning for reward estimation in inverse reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2009, Bled, Slovenia, September 7-11, 2009, Proceedings, Part II 20*, pages 31–46. Springer Berlin Heidelberg, 2009. 10, 81

[157] W. Luo, A. Schwing, and R. Urtasun. Latent structured active learning. *Advances in Neural Information Processing Systems*, 26, 2013. 10

[158] D. Mahapatra, B. Bozorgtabar, J.-P. Thiran, and M. Reyes. Efficient active learning for image classification and segmentation using a sample selection and conditional generative adversarial network. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 580–588. Springer, 2018. 10

[159] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021. 36

[160] J. S. Matthis, J. L. Yates, and M. M. Hayhoe. Gaze and the control of foot placement when walking in natural terrain. *Current Biology*, 28(8):1224–1233, 2018. 81, 82

[161] C. Mayer and R. Timofte. Adversarial sampling for active learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3071–3079, 2020. 10

[162] M. K. McBeath, D. M. Shaffer, and M. K. Kaiser. How baseball outfielders determine where to run to catch fly balls. *Science*, 268(5210):569–573, 1995. 81, 82

[163] O. Mees, M. Merklinger, G. Kalweit, and W. Burgard. Adversarial skill networks: Unsupervised robot skill learning from video. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4188–4194. IEEE, 2020. 6, 12

[164] A. N. Meltzoff. Imitation of televised models by infants. *Child development*, 59(5):1221, 1988. 1, 12

[165] P. Ménard, O. D. Domingues, A. Jonsson, E. Kaufmann, E. Leurent, and M. Valko. Fast active learning for pure exploration in reinforcement learning. In *International Conference on Machine Learning*, pages 7599–7608. PMLR, 2021. 10, 81

[166] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis.

In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–421. Springer, 2020. 39

[167] M. B. Mirza, R. A. Adams, C. D. Mathys, and K. J. Friston. Scene construction, visual foraging, and active inference. *Frontiers in computational neuroscience*, 10:56, 2016. 10

[168] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 527–544. Springer, 2016. 36

[169] S. Mittal, M. Tatarchenko, Ö. **c**Ciccek, and T. Brox. Parting with illusions about deep active learning. *arXiv preprint arXiv:1912.05361*, 2019. 10

[170] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning, 2013. arXiv:1312.5602. 1, 8, 84, 87, 90, 91

[171] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. 57

[172] G. Monaci, M. Aractingi, and T. Silander. DiPCAN: Distilling privileged information for crowd-aware navigation. In *Robotics: Science and Systems (RSS) XVIII*, 2022. 69

[173] P. Munjal, N. Hayat, M. Hayat, J. Sourati, and S. Khan. Towards robust and reproducible active learning using neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 223–232, 2022. 10

[174] A. Nair, S. Bahl, A. Khazatsky, V. Pong, G. Berseth, and S. Levine. Contextual imagined goals for self-supervised robotic learning. In *Conference on Robot Learning*, pages 530–539. PMLR, 2020. 7

[175] A. Narr, R. Triebel, and D. Cremers. Stream-based active learning for efficient and adaptive classification of 3d objects. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 227–233. IEEE, 2016. 10

[176] D. Neimark, O. Bar, M. Zohar, and D. Asselmann. Video transformer network, 2021. arXiv:2102.00719. 9

[177] K. Nguyen and H. Daumé III. Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 684–695, 2019. 10

[178] K. Nguyen, D. Dey, C. Brockett, and B. Dolan. Vision-based navigation with language-based assistance via imitation learning with indirect intervention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12527–12537, 2019. 10

[179] D. Nilsson, A. Pirinen, E. Gärtner, and C. Sminchisescu. Embodied visual active learning for semantic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2373–2383, 2021. 10

[180] S. Nolfi and M. Mirolli. *Evolution of communication and language in embodied agents*. Springer Science & Business Media, 2009. 1

[181] OpenAI. Gpt-4 technical report, 2023. 2, 5, 100

[182] J. K. O'regan and A. Noë. A sensorimotor account of vision and visual consciousness. *Behavioral and brain sciences*, 24(5):939–973, 2001. 82

[183] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference, 2016. arXiv:1606.01933. 52

[184] X. Peng, Z. Huang, X. Sun, and K. Saenko. Domain agnostic learning with disentangled representations. In *International Conference on Machine Learning*, pages 5102–5112. PMLR, 2019. 7, 16

[185] J. Piaget. Piaget's theory, 1976. 82, 86

[186] A. Piergiovanni and M. S. Ryoo. Recognizing actions in videos from unseen viewpoints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4124–4132, June 2021. 8, 29

[187] A. Piergiovanni, V. Casser, M. S. Ryoo, and A. Angelova. 4d-net for learned multi-modal alignment. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 15435–15445, 2021. 8

[188] R. Pinsler, J. Gordon, E. Nalisnick, and J. M. Hernández-Lobato. Bayesian batch active learning as sparse subset approximation. *Advances in neural information processing systems*, 32, 2019. 10

[189] V. Pong, S. Gu, M. Dalal, and S. Levine. Temporal difference models: Model-free deep rl for model-based control. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 62

[190] D. Premack and G. Woodruff. Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(4):515–526, 1978. 12

[191] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 87

[192] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training, 2018. 9

[193] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8), 2019. 9, 53, 62

[194] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021. 2, 5, 100

[195] K. Ranasinghe, M. Naseer, S. Khan, F. S. Khan, and M. Ryoo. Self-supervised video transformer. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 8

[196] A. Rangel and J. A. Clithero. Value normalization in decision making: theory and evidence. *Current opinion in neurobiology*, 22(6):970–981, 2012. 86

[197] D. Robert, B. Vallet, and L. Landrieu. Learning multi-view aggregation in the wild for large-scale 3d semantic segmentation. *arXiv preprint arXiv:2204.07548*, 2022. 8

[198] D. Rukhovich, A. Vorontsova, and A. Konushin. Imvoxelnet: Image to voxels projection for monocular and multi-view general-purpose 3d object

detection. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2397–2406, 2022. 8

[199] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In J. W. Shavlik and T. G. Dietterich, editors, *Readings in Machine Learning*, pages 115–137, San Mateo, CA, 1990. Kaufmann. 87

[200] G. A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems*, volume 37. Citeseer, 1994. 8

[201] M. S. Ryoo, T. J. Fuchs, L. Xia, J. K. Aggarwal, and L. Matthies. Robot-centric activity prediction from first-person videos: What will they do to me? In *Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction*, pages 295–302, 2015. 11

[202] M. S. Ryoo, A. Piergiovanni, A. Arnab, M. Dehghani, and A. Angelova. TokenLearner: Adaptive space-time tokenization for videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, Dec. 2021. 8, 9

[203] M. S. Ryoo, K. Gopalakrishnan, K. Kahatapitiya, T. Xiao, K. Rao, A. Stone, Y. Lu, J. Ibarz, and A. Arnab. Token turing machines. *arXiv preprint arXiv:2211.09119*, 2022. 87

[204] S. Salter, D. Rao, M. Wulfmeier, R. Hadsell, and I. Posner. Attention-privileged reinforcement learning. In *Conference on Robot Learning*, pages 394–408. PMLR, 2021. 10

[205] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020. 41

[206] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A platform for embodied ai research. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct 2019. 11

[207] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017. 20, 26

[208] O. Sener and S. Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017. 10

[209] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE, 2018. 6, 7, 8, 11, 12, 13, 15, 16, 17, 19, 21, 22, 23, 24, 26, 27, 36, 37, 39

[210] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 45

[211] J. Shang and M. S. Ryoo. Self-supervised disentangled representation learning for third-person imitation learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 214–221, 2021. doi: 10.1109/IROS51168.2021.9636363. 1, 8, 36, 39

[212] J. Shang and M. S. Ryoo. Self-supervised disentangled representation learning for third-person imitation learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 214–221. IEEE, 2021. 11, 81

[213] J. Shang and M. S. Ryoo. Active reinforcement learning under limited visual observability. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 1

[214] J. Shang, S. Das, and M. Ryoo. Learning viewpoint-agnostic visual representations by recovering tokens in 3d space. *Advances in Neural Information Processing Systems*, 35:31031–31044, 2022. 11

[215] J. Shang, K. Kahatapitiya, X. Li, and M. S. Ryoo. Starformer: Transformer with state-action-reward representations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 8, 87

[216] J. Shang, X. Li, K. Kahatapitiya, Y.-C. Lee, and M. S. Ryoo. Starformer: Transformer with state-action-reward representations for robot learning. *IEEE transactions on pattern analysis and machine intelligence*, 2022. 8, 87

[217] J. Shields, O. Pizarro, and S. B. Williams. Towards adaptive benthic habitat mapping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9263–9270. IEEE, 2020. 10

[218] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022. 5, 100

[219] Z. Shu, E. Yumer, S. Hadap, K. Sunkavalli, E. Shechtman, and D. Samaras. Neural face editing with intrinsic image disentangling. In *Computer Vision and Pattern Recognition, 2017. CVPR 2017. IEEE Conference on*, pages –. IEEE, 2017. 7

[220] C. Shui, F. Zhou, C. Gagné, and B. Wang. Deep active learning: Unified and principled method for query and training. In *International Conference on Artificial Intelligence and Statistics*, pages 1308–1318. PMLR, 2020. 10

[221] G. A. Sigurdsson, A. Gupta, C. Schmid, A. Farhadi, and K. Alahari. Actor and observer: Joint modeling of first and third-person videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7396–7404, 2018. 8

[222] G. A. Sigurdsson, A. Gupta, C. Schmid, A. Farhadi, and K. Alahari. Charades-ego: A large-scale dataset of paired third and first person videos. *arXiv preprint arXiv:1804.09626*, 2018. 8

[223] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017. 81

[224] S. Sinha, S. Ebrahimi, and T. Darrell. Variational adversarial active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5972–5981, 2019. 10

[225] L. Smith and M. Gasser. The development of embodied cognition: Six lessons from babies. *Artificial life*, 11(1-2):13–29, 2005. 1

[226] K. Sohn, X. Yan, and H. Lee. Learning structured output representation using deep conditional generative models. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 3483–3491, 2015. 7

[227] I. Sorokin, A. Seleznev, M. Pavlov, A. Fedorov, and A. Ignateva. Deep attention recurrent q-network, 2015. 10

[228] A. Srinivas, M. Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning, 2020. 7

[229] R. K. Srivastava, P. Shyam, F. Mutz, W. Jaśkowski, and J. Schmidhuber. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019. 62

[230] B. C. Stadie, P. Abbeel, and I. Sutskever. Third-person imitation learning. *ArXiv*, abs/1703.01703, 2017. 1, 6, 8, 12, 13, 15, 36

[231] B. C. Stadie, P. Abbeel, and I. Sutskever. Third-person imitation learning. *arXiv preprint arXiv:1703.01703*, 2017. 11

[232] A. Stooke, K. Lee, P. Abbeel, and M. Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pages 9870–9879. PMLR, 2021. 1

[233] J. J. Sun, J. Zhao, L.-C. Chen, F. Schroff, H. Adam, and T. Liu. View-invariant probabilistic embedding for human pose. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 53–70. Springer, 2020. 8

[234] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, Aug. 1988. 8

[235] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. doi: 10.1109/CVPR.2016.308. 7, 29

[236] Y. Tang and D. Ha. The sensory neuron as a transformer: Permutation-invariant neural networks for reinforcement learning. *arXiv preprint arXiv:2109.02869*, 2021. 9

[237] Y. Tang, D. Nguyen, and D. Ha. Neuroevolution of self-interpretable agents. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 414–424, 2020. 10, 90

[238] G. Tesauro et al. Temporal difference learning and TD-Gammon. *Commun. ACM*, 38(3):58–68, Mar. 1995. 8

[239] J. Thomason, M. Murray, M. Cakmak, and L. Zettlemoyer. Vision-and-dialog navigation. In *Conference on Robot Learning*, pages 394–406. PMLR, 2020. 10

[240] Y. Tian, D. Krishnan, and P. Isola. Contrastive multiview coding. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 776–794, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58621-8. 7, 17

[241] F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018. 1, 82, 86

[242] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 10347–10357, Jul. 2021. 8, 29, 30, 35, 37, 39

[243] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou. Going deeper with image transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 32–42, Oct. 2021. 51

[244] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 2, 5, 100

[245] A. Tucker, A. Gleave, and S. Russell. Inverse reinforcement learning for video games. *arXiv preprint arXiv:1810.10593*, 2018. 86

[246] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. 81, 82, 88, 93

[247] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, Nov. 2020. 60

[248] T. Van de Maele, T. Verbelen, O. **c**Catal, and B. Dhoedt. Embodied object representation learning and recognition. *Frontiers in Neurorobotics*, 16:840658, 2022. 10

[249] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 23

[250] H. P. van Hasselt, A. Guez, M. Hessel, V. Mnih, and D. Silver. Learning values across many orders of magnitude. *Advances in neural information processing systems*, 29, 2016. 86

[251] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, **L**. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, Dec. 2017. 9, 49, 52, 56, 57

[252] A. Vaswani et al. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 34

[253] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017. 93

[254] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017. 8

[255] F. Wan, T. Yuan, M. Fu, X. Ji, Q. Huang, and Q. Ye. Nearest neighbor classifier embedded network for active learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10041–10048, 2021. 10

[256] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. 2, 5, 100

[257] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity, 2020. arXiv:2006.04768. 9

[258] X. Wang, A. Jabri, and A. A. Efros. Learning correspondence from the cycle-consistency of time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2566–2576, 2019. 37

[259] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, May 1992. 1, 8

[260] S. D. Whitehead and D. H. Ballard. Active perception and reinforcement learning. In *Machine Learning Proceedings 1990*, pages 179–188. Elsevier, 1990. 10, 81, 82

[261] D. M. Wolpert, J. Diedrichsen, and J. R. Flanagan. Principles of sensorimotor learning. *Nature reviews neuroscience*, 12(12):739–751, 2011. 82

[262] H. Wu, K. Khetarpal, and D. Precup. Self-supervised attention-aware reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10311–10319, 2021. 10

[263] S. Xiang. Eliminating topological errors in neural network rotation estimation using self-selecting ensembles. *ACM Transactions on Graphics (TOG)*, 40 (4):1–21, 2021. 33

[264] Y. Xie, H. Lu, J. Yan, X. Yang, M. Tomizuka, and W. Zhan. Active finetuning: Exploiting annotation budget in the pretraining-finetuning paradigm. *arXiv preprint arXiv:2303.14382*, 2023. 10

[265] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016. 8

[266] J. Yang, Z. Ren, M. Xu, X. Chen, D. J. Crandall, D. Parikh, and D. Batra. Embodied amodal recognition: Learning to move to perceive objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2040–2050, 2019. 10

[267] J. Yang, C. Li, P. Zhang, X. Dai, B. Xiao, L. Yuan, and J. Gao. Focal self-attention for local-global interactions in vision transformers, 2021. arXiv:2107.00641. 51

[268] R. Yang, G. Yang, and X. Wang. Neural volumetric memory for visual locomotion control. In *Conference on Computer Vision and Pattern Recognition*, 2023. 11, 87

[269] S. Yang, W. Zhang, W. Lu, H. Wang, and Y. Li. Cross-context visual imitation learning from demonstrations. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5467–5473. IEEE, 2020. 6

[270] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021. 90, 93

[271] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 10674–10681, May 2021. 60

[272] S. Yenamandra, A. Ramachandran, K. Yadav, A. Wang, M. Khanna, T. Gervet, T.-Y. Yang, V. Jain, A. W. Clegg, J. Turner, et al. Homerobot: Open-vocabulary mobile manipulation. *arXiv preprint arXiv:2306.11565*, 2023. 5, 100

[273] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 558–567, 2021. 8

[274] T. Yuan, F. Wan, M. Fu, J. Liu, S. Xu, X. Ji, and Q. Ye. Multiple instance active learning for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5330–5339, 2021. 10

[275] G. Zelinsky, W. Zhang, B. Yu, X. Chen, and D. Samaras. The role of top-down and bottom-up processes in guiding eye movements during visual search. *Advances in neural information processing systems*, 18, 2005. 94

[276] A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, et al. Socratic models: Composing zero-shot multimodal reasoning with language. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 5, 100

[277] T. Zhang, Y. Li, C. Wang, G. Xie, and Z. Lu. Fop: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 12491–12500. PMLR, 2021. 85

[278] Z. Zhang, L. Tran, X. Yin, Y. Atoum, X. Liu, J. Wan, and N. Wang. Gait recognition via disentangled representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4710–4719, 2019. 7

[279] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun. Point transformer. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 16259–16268, October 2021. 8

[280] Q. Zheng, A. Zhang, and A. Grover. Online decision transformer, 2022. 9

[281] A. Zhou, M. J. Kim, L. Wang, P. Florence, and C. Finn. Nerf in the palm of your hand: Corrective augmentation for robotics via novel-view synthesis. *arXiv preprint arXiv:2301.08556*, 2023. 11

[282] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. 33

[283] J.-J. Zhu and J. Bento. Generative adversarial active learning. *arXiv preprint arXiv:1702.07956*, 2017. 10

[284] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020. 88, 90

[285] K. Zolna, S. Reed, A. Novikov, S. G. Colmenarej, D. Budden, S. Cabi, M. Denil, N. de Freitas, and Z. Wang. Task-relevant adversarial imitation learning. *arXiv preprint arXiv:1910.01077*, 2019. 7