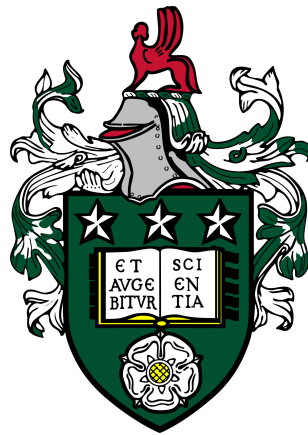


**Seeing to learn:
Observational learning of robotic manipulation tasks**

by
Leo Pauly

Submitted in accordance with the requirements for the degree of
Doctor of Philosophy



University of Leeds
July 2021

School of Civil Engineering

School of Computing

Declaration

The candidate confirms that the work submitted is his own and that appropriate credit has been given where reference has been made to the work of others. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement. The right of Leo Pauly to be identified as Author of this work has been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.

Abstract

Learning new tasks has always been a challenging problem in robotics. Even though several approaches have been proposed, from manual programming to learning from demonstrations, the field has directions which require further research and development. This thesis focuses on one of these relatively unexplored areas: observational learning.

We present O_2A , a novel method for learning to perform robotic manipulation tasks from a single (one-shot) third-person demonstration video. The key novelty lies in pre-training a feature extractor for creating an abstract feature representation for actions that we call '*action vectors*'. The action vectors are extracted using a 3D-CNN network pre-trained for action recognition on a generic action dataset. The distance between the action vectors from the observed third-person demonstration and trial robot executions is used as a reward/cost for learning of the demonstrated task.

We report on experiments in simulation and on a real robot, with changes in viewpoint of observation, properties of the objects involved, scene background and morphology of the manipulator between the demonstration and the learning domains. O_2A outperforms baseline approaches under different domain shifts and has comparable performance with an oracle (that uses an ideal reward function). We also plot visualisation of trajectories and show that our method has high reward for desired trajectories. Videos of the results, including demonstrations, can be found in our: [project-website](#).

Finally, we present a framework for extending observational learning with multi modal observations. We report our initial experiments and results in the future works.

Acknowledgement

First and foremost I thank my supervisors, Professor Raul Fuentes and Professor David Hogg, for their guidance throughout this PhD. From not missing any of our meetings to reviewing all the drafts and manuscripts, it was a great privilege to be your student. You mentored me as a researcher in many ways: inspired to come up with new ideas, helped to formulate research problems and advised with deep insights in implementation and evaluation of the solutions. You were always there for me no matter how difficult the situation was. Thank you for everything.

I thank the University of Leeds for awarding the Leeds Anniversary Research Scholarship which funded my research. Without the scholarship, I would not have been able to pursue my dream of higher studies and research. Also, I thank the university for providing such a vibrant student environment which helped me to balance my social life throughout the rigours of doing a PhD.

I'm grateful to all my friends and fellow researchers for helping with this journey. I dare not list any names for the fear the list is (too!) long and may miss a name. I had a great time with you all. Thank you for your friendship, care, guidance, advice and support.

I owe all my success to my loving parents Pauly and Mary. You always supported and encouraged me to pursue my dreams. For all the sacrifices you made to provide me with a good life and education, I love you both so much. Also special gratitude to my siblings Geyo and Leya for their closeness even though I was half the world away from home.

Finally, I express my affection to Juka for your love and care. I will miss you.

Contents

List of Figures	viii
List of Tables	xiv
1 Introduction	1
1.1 Defining observational learning	3
1.2 Motivation	5
1.3 Aim and objectives	7
1.4 List of contributions	8
1.5 Publications	9
1.6 Thesis summary and outline	9
2 Literature review	12
2.1 Introduction	12
2.2 Observation	15
2.2.1 Assisted observation	15
2.2.2 Direct observation	18
2.3 Abstract feature representation	19
2.3.1 Deep metric learning	19
2.3.2 Generative adversarial learning	21
2.3.3 Domain translation	22

2.3.4	Transfer learning	24
2.3.5	Geometrical representation learning	24
2.3.6	Action primitives	25
2.3.7	Predictive modelling	26
2.3.8	Video to text translation	27
2.4	Execution	28
2.4.1	Inverse reinforcement learning	28
2.4.2	Direct regression	30
2.4.3	Model predictive control	31
2.4.4	Action templates	32
2.4.5	Generative adversarial learning	33
2.4.6	Meta-learning	33
2.5	Discussion	34
2.6	Comparative study	35
2.7	Conclusion	39
3	Action vectors	40
3.1	Introduction	40
3.2	Pre-training with large generic datasets	41
3.3	Action vectors	42
3.3.1	Dataset	42
3.3.2	Network architecture	44
3.3.3	Pre-training objective	46
3.3.4	Pre-training and action vector extraction	48
3.4	Analysing action vectors	50
3.4.1	LMD evaluation Dataset	50
3.4.2	Clustering analysis	50
3.4.3	Class similarity scores	55
3.4.4	Visualisation	58
3.4.5	Discussion	62
3.4.6	Pre-training accuracy and transfer performance	62
3.5	Conclusion	64

4	One-shot observational learning	65
4.1	Introduction	65
4.2	O ₂ A overview	65
4.3	Simulation experiment	68
4.3.1	DDPG (Deep Deterministic Policy Gradient) RL algorithm	68
4.3.2	Experimental setup	74
4.3.3	Oracle and baselines	75
4.3.4	Correlation of rewards	77
4.3.5	Trajectory maps	80
4.3.6	Limitations	90
4.3.7	Reach - Push	91
4.4	Real robot experiment	93
4.4.1	Stochastic Trajectory Optimisation (STO)	93
4.4.2	Experimental setup	95
4.4.3	Results	98
4.5	Conclusion	98
5	Conclusion and Future Work	100
5.1	Conclusion of the thesis	100
5.2	Limitations	103
5.3	Future works	104
5.3.1	Multi-modal (one-shot) observational learning	104
5.3.2	Pre-training objectives	105
5.3.3	Reducing number of trial robot executions	106
5.3.4	O ₂ A beyond manipulation tasks	107
5.3.5	Observational learning of long horizon manipulation tasks	107
6	Appendix	109
6.1	Recent Action datasets	109
6.2	UCF101 Dataset	109
6.2.1	Action catagories	110
6.3	Feature selection	111
6.4	3D visualisation	111

6.5	Clustering analysis after PCA	116
6.6	Simulation experiment environment designs	116
6.6.1	Reach	116
6.6.2	Push	117
6.7	DDPG RL algorithm	119
6.8	MIME dataset task categories	120
	Bibliography	121

List of Figures

1.1	Imitation learning by (a) teleoperation [1] and (b) kinesthetic teaching [2]	2
1.2	Observational learning consists of observing the demonstrations, extracting an abstract feature representations and executing the demonstrated task.	4
1.3	Thesis summary	11
2.1	Different approaches used for learning robotic manipulation tasks. Observational learning approaches are used when demonstrations with domain shifts are provided for learning tasks.	13
2.2	Organisational structure of remaining sections in the chapter.	14
2.3	Motion capture system. Note the inconvenience for the demonstrator in wearing on body sensors. Figure from [3]	16
2.4	LED trackers used for recording the positions and velocities of the manipulated objects. Figure from [4].	16
2.5	Visual detectors used for assisting observations. Figure from [5]	17
2.6	Full body skeleton tracking of the task demonstrator. Figure from [6].	17

2.7	Training time-contrastive networks with triplet loss for feature extraction from demonstration videos. The anchor and the positive images are encouraged to be closer in the feature space, while being further away from the negative image. Figure from [7].	20
2.8	Basic generative adversarial learning framework. Figure from [8].	21
2.9	Domain translation. Images are translated from the demonstration domain (top row) to the observers domain (bottom row). Figure from [9].	22
2.10	Domain translation method using generator-discriminator network. Fig. from [10].	24
2.11	Representing insertion manipulation task in terms of geometrical primitives (a): points (coloured: blue) and lines (coloured: red, yellow, pink) and their associations(b). Figure from [11].	25
2.12	Decomposing actions into action primitives for task of (a) Pick and place (b) Pushing away (c) Opening bottle. Colour bars indicates the time duration of the action primitives in the task video. Figure from [12].	26
2.13	Textual descriptions from demonstrations. Figure from [13].	27
2.14	Values from a learned reward function for videos depicting (a) successful and (b) failed execution of the pouring task. It can be seen that the reward value drops, as the task fails to execute in (b). Figure from [13].	30
2.15	Illustration of a general MPC system . Figure from [14].	31
2.16	MAML algorithm illustration. Figure from [15].	34
3.1	An illustration of the conceptual action vector space. Action vectors from videos of a task are closer to each other irrespective of the domain settings in which they are recorded and further away from other classes.	43
3.2	Selected few classes from UCF101 action dataset illustrating the diversity in terms actions and domain settings. Figure from [16].	45
3.3	(a) 2D convolution (b) 3D convolution. Figure adapted from [17].	46
3.4	3D pooling	46

3.5	Downsampling training video before feeding into the network. Down-sampled size (N_f) set to 5 frames for ease of illustration.	48
3.6	Testing accuracy (on UCF101 test set) per epoch during action recognition pre-training with UCF101 dataset	49
3.7	Tasks of (a) reaching (b) pushing and (c) reach-push from LMD	51
3.8	Diversity in the data collected for the task of pushing. (a) Normal pushing and pushing with changes in: (b) viewpoint of observation (c) object properties (d) scene background and (e) morphology of the manipulator. Similar diversity can be observed in other classes as well.	52
3.9	Visualising LMD using action vectors for Baseline-R (features from pool5 layer of NN:UCF101 with randomly initialised weights are used)	58
3.10	Visualising LMD using action vectors from (a) pool5 and (b) fc6 layers of NN:UCF101	59
3.11	Visualising LMD (after merging reach and push classes) using action vectors for Baseline-R (features from pool5 layer of NN:UCF101 with randomly initialised weights are used).	60
3.12	Visualising LMD (after merging reach and push classes) using action vectors from (a) pool5 and (b) fc6 layers of NN:UCF101	61
3.13	ARI scores when NN:UCF101 network is pre-trained for higher action recognition accuracies on UCF101 dataset.	63
4.1	Overview of O ₂ A method. A 3D-CNN action vector extractor is used to extract action vectors \mathbf{X}_D and \mathbf{X}_R from the video clips of the demonstration and robot trial execution respectively. A reward function is used to compare \mathbf{X}_D and \mathbf{X}_R in the action vector space, generating a reward signal (r) based on their closeness. The RL algorithm then iteratively learns an optimal control policy by maximizing this reward signal, thus enabling observational learning.	66
4.2	Sample environments available in OpenAI Gym	68
4.3	Custom designed environments for the tasks of (a) reaching and (b) pushing in simulation experiments	70

4.4	Critic network architecture	71
4.5	Actor network architecture	72
4.6	Snapshots of the demonstration and the execution of corresponding learned policies in the simulation experiment for selected domain shifts. (Results shown for action vectors extracted from pool5 layer of NN:UCF101 network).	75
4.7	Task completion measures for the task of (a) reaching and (b) pushing in the simulation experiment. O ₂ A outperforms both the baselines and has performance comparable to the oracle under all domain shifts. The oracle score is shown only once since it is unaffected by the domain shifts (refer to Table 4.3 for domain shift definitions).	79
4.8	Selected few examples (execution by human and robotic manipulator) from MIME action dataset. The tasks are (clockwise from top-left) stirring, pouring, stacking, wiping, opening a bottle and passing. Figure from [18]	80
4.9	Testing accuracy (on MIME test set) per epoch during pre-training the action vector extractor on MIME dataset	82
4.10	Trajectory maps obtained during task learning under identical domain settings for (a) reaching (left) (b) pushing (right). High rewards are obtained for desired trajectories in all the cases.	83
4.11	Trajectory maps obtained during task learning with changes in viewpoint of observation for (a) reaching (left) (b) pushing (right). High rewards are obtained for desired trajectories when the NN:UCF101 network is used.	84
4.12	Trajectory maps obtained during task learning under identical domain settings for (a) reaching (left) (b) pushing (right). High rewards are obtained for desired trajectories in all the cases.	85
4.13	Trajectory maps obtained during task learning with changes in viewpoint of observation and object properties for (a) reaching (left) (b) pushing (right). High rewards are obtained for desired trajectories when the NN:UCF101 network is used.	86

4.14	Trajectory maps obtained during task learning when background clutter is introduced in to the background for (a) reaching (left) (b) pushing (right). High rewards are obtained for desired trajectories in all the cases.	87
4.15	Trajectory maps obtained during task learning when demonstration is provided by a human hand for (a) reaching (left) (b) pushing (right). High rewards are obtained for desired trajectories only when NN:UCF101 network is used.	88
4.16	Trajectory maps obtained during task learning for (a) reaching and (b) pushing. New layout for the learning environment is generated by moving target region to the left. O ₂ A was unable to identify desired trajectories for the domain shift.	90
4.17	Trajectory maps obtained during task learning for (a) reaching and (b) pushing. New layout for the learning environment is generated by moving target region to the right. O ₂ A was unable to identify desired trajectories for the domain shift.	91
4.18	Snapshots of the video samples of the reach-push task collected. It includes a demonstration (D) and video samples showing varying degrees of task completion (A-E).	92
4.19	Normalised reward values for video pairs D-A to D-E. Higher rewards are obtained when task moves towards completion, showing O ₂ A can successfully model the more complete reach-push task as well.	92
4.20	Real robot experiment tasks. From left-right: pushing, hammering, sweeping and striking	93
4.21	Real robot experimental setup overview	95
4.22	(a) Execution system experimental setup built for running real robot experiments. It consists of a local laptop workstation (A), custom made camera holder (B), camera (C) and the end effector (D) attached to the UR5 robot (E). (b) View of the robot.	96

4.23	Snapshot of the demonstration and execution of the corresponding optimal control sequences obtained for selected domain shifts from the real robot experiment (Results shown for action vectors extracted from the pool5 layer of the NN:UCF101 network).	98
4.24	Task completion measures for the task of pushing, hammering, sweeping and striking in the real robot experiment. The result shows that O ₂ A performs well under different domain shifts on a real robot. . .	99
5.1	Overview of pre-training sound vector extractors using UrbanSound8k dataset and its usage during observational learning.	106
5.2	Modified LunarLander RL environment	107
6.1	Examples from LMD_Sup dataset for the task classes (a) Move and (b) Displace.	111
6.2	ARI scores when different subsets of features from pool5 and fc6 layers are used as the action vector. High ARI scores when only the top 20% of features (based on ANOVA F-values) are used, indicate the existence of a subset of features more significant than others in representing manipulation tasks. Tags at each point give the corresponding number of features selected.	112
6.3	Visualising LMD using action vectors for Baseline-R (features from pool5 layer of NN:UCF101 with randomly initialised weights) . . .	113
6.4	Visualising LMD using action vectors from pool5 layer of NN:UCF101	114
6.5	Visualising LMD using action vectors from fc6 layer of NN:UCF101	115
6.6	DDPG RL algorithm	119

List of Tables

1.1	Differences between imitation learning and observational learning .	3
2.1	Advantages and disadvantages of different methods used for assisted observation	18
2.2	Observational learning methods in existing literature are compared, based on approaches used in different stages.	37
2.3	Observational learning methods in existing literature are compared. O ₂ A requires only a single demonstration to learn new tasks. It does not use any robot data for training the action vector extractor. And also works well under different all domain shifts.	38
3.1	Publicly available action datasets. UCF101 dataset was the most suited at the time for pre-training, in terms of number action classes and video samples per class.	44
3.2	Network architecture for the action vector extractor	47
3.3	Details of action recognition pre-training on the UCF101 dataset . .	49
3.4	Leeds Action Dataset details	51
3.5	K-means algorithm parameters used	53
3.6	ARI scores (higher the better). Results show that the features from layers pool5 and fc6 of the NN:UCF101 network are best suited to be used as action vectors.	54

3.7	ARI scores when features from different layers are concatenated and used as the action vector. Results do not show any significant improvement in the performance.	55
3.8	Class similarity scores. The intraclass similarity (diagonal values) are greater than the rest of the values, indicating adequate task-discrimination and domain-invariance.	57
3.9	Transfer performance evaluation when action vector extractor (NN:UCF101) is pre-trained for higher action recognition testing accuracies.	63
4.1	Task definitions and completion measures	67
4.2	Comparing OpenAI gym and rllab RL frameworks. It shows our criteria for selecting the RL framework for the simulation experiment. '★' symbol shows a better performance. OpenAI Gym is a clear choice for us satisfying all the requirements.	69
4.3	Domain shifts used in our experiment	76
4.4	DDPG parameters settings	76
4.5	Pearson correlation coefficients between the rewards from the oracle, and from O2A and two baselines. The coefficients are highest in most of the cases and positive for O ₂ A rewards compared to baseline approaches.	78
4.6	Robotic manipulation tasks video datasets. MIME is the largest available robotic manipulation tasks video dataset containing both human and robotic demonstrations. However, building ImageNet[19] scale robotic manipulation datasets is crucial for future research.	81
4.7	Details of pre-training the action vector extractor on the MIME dataset	82
5.1	Inter-class and intra-class distances for LSD in the action vector space. Class-1 is 'hammering on target' and Class-2 is 'hammering on table'.	105
6.1	Recently released larger action datasets	109
6.2	UCF101 action recognition dataset details	110
6.3	ARI scores after applying PCA dimensionality reduction on features from different layers of NN:UCF101	116

"Research means that you don't know, but are willing to find out."

– Charles F. Kettering

The twenty-first century has seen a considerable growth in robotic technologies. From room-cleaning to advanced medical applications, robots are making their way into human lives [20, 21, 22]. But one of the major challenges faced by even the most advanced robotic technologies lies in the inability to learn tasks from demonstrations like human beings. The traditional method of teaching tasks to robots is by manual step by step programming [23, 24]. But this approach cannot be applied in real world scenarios especially for consumer robotics, where the robotic operators need not be programmers. A practical solution for this problem is to adapt the human approach of learning from demonstrations. For example, a 3 year old child is taught new tasks not by algorithmic programming but rather by showing simple demonstrations. Hence Learning from Demonstration (LfD) has been a top priority in robotics research for the past few decades. Comprehensive surveys of the methods and techniques developed for LfD over the years can be found in [25, 26, 27, 28, 29, 30, 31].

Even though LfD has been studied in robotics for a very long time, the major share of the research works fall into the category of imitation learning. In imitation learning, the robotic system is made to learn from first person demonstrations. The demonstrations can be provided by teleoperation [1] or kinesthetic teaching [2] as shown in Figure 1.1. In teleoperation, the demonstrator guides

the robotic system indirectly using teleoperation devices to perform a task. During teleoperation, the robotic system records its joint trajectories and/or visually observes its own actions, to be used as the demonstrations for learning the task. Whereas in kinesthetic teaching, the demonstrator directly guides the robotic system to perform the task. The robotic joint trajectories obtained during this process are recorded and used as the demonstrations.

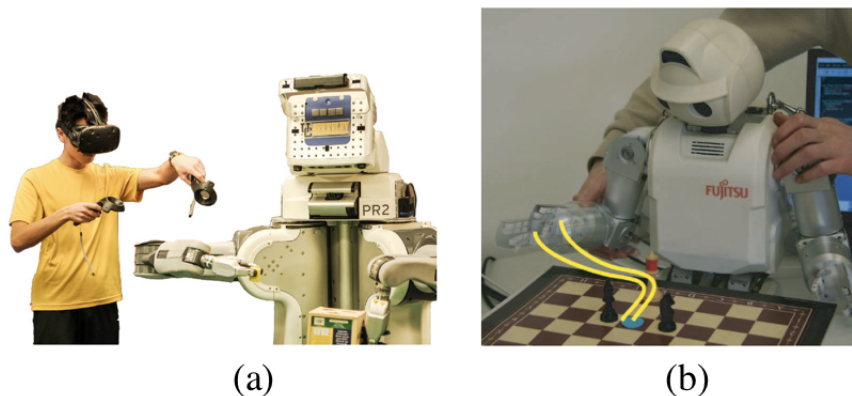


Figure 1.1: Imitation learning by (a) teleoperation [1] and (b) kinesthetic teaching [2]

Several approaches have been developed for implementing imitation learning in robotics with high degrees of success [1, 2]. However, imitation learning methods suffer from a key limitation in terms of the demonstrations that can be used for learning a task. The demonstrations have to be provided as first person observations, with access to robot joint trajectories during demonstration. This limits the ability of robotic systems to learn from demonstrations which occur naturally. For example, a search for task demonstrations on the web invariably retrieves videos mostly recorded as third person observations with no access to the corresponding robotic joint trajectories during demonstrations.

Observational learning [31, 32, 33] methods aim at overcoming this challenge. The origin of observational learning can be traced back to works published in the early 1990s [34, 35, 36]. It differs from imitation learning in that the demonstrations are observed as a third person without access to robotic joint trajectories. Observing demonstrations from a third person also give rise to domain shifts between the demonstration and learning environments. The domain shifts can be

changes in viewpoint of observation, object properties, scene background and/or morphology of the manipulators. This makes observational learning even harder to solve. The differences between imitation learning and observational learning are summarised in Table 1.1. A plain imitation of the demonstration will not be sufficient to achieve the desired behaviour. The robotic system will have to first learn a domain-agnostic abstract feature representation of the task demonstrated and then perform an execution which will have a similar representation.

Table 1.1: Differences between imitation learning and observational learning

	Imitation learning	Observational learning
Domain shift	No domain shift. The demonstrations are performed in the same environment in which robots are learning.	Domain shifts can occur as changes in viewpoint of observation, object properties, scene background and/or morphology of the manipulator.
Access to joint trajectories	Robotic system has access to it's own joint trajectories during the demonstration.	Does not have access to robotic joint trajectories.

Observational learning is also refereed to as Learning by Watching[35], Teaching by Showing[37], Plan from Observation [36], Visual Imitation Learning [38], Third person Imitation Learning [39], Imitation from Observation [40], Imitation Learning with Domain shift [41] and Third person Visual Imitation Learning [10]. Even though the definitions vary slightly between articles, the core concept of observational learning remains the same. In the next section we explain the concept of observational learning and its stages in detail.

1.1 Defining observational learning

It is worth noting that the concept of observational learning has long been investigated in the fields of cognitive and behavioral psychology. American psychologists Bandura and Walters [42, 43, 44], postulated that people learn from one another via observation, mimicking and modeling the demonstrator's behaviors and coined the term 'observational learning' or 'social learning' to describe this be-

behaviour. They suggested that observational learning has four parts: attention, retention, reproduction and motivation. Attention is observing what is happening, retention refers to memorising and creating a mental model of the task, reproduction is the execution of the task inspired by an incentive referred to as motivation. A detailed interdisciplinary overview of this line of work on observational and imitation learning can be found in the book ‘Imitation in Animals and Artifacts’ [45].

Inspired by the concepts in psychology and existing robotics literature, observational learning in robotics can be decomposed into three stages: observation, abstract feature representation and execution as illustrated in the Fig 1.2. We describe each of these in the following subsections.

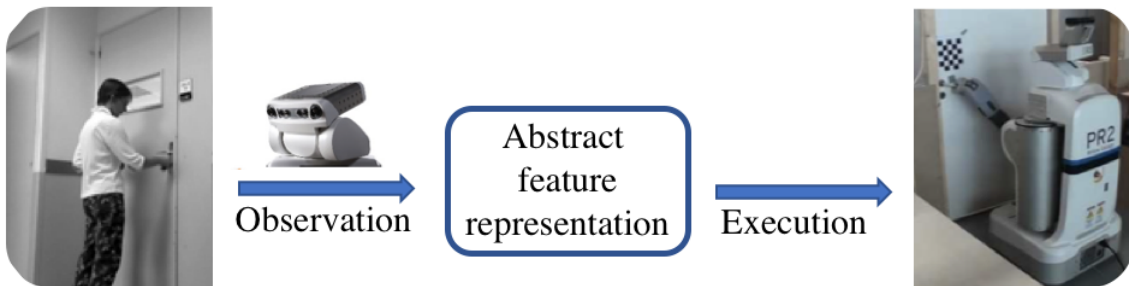


Figure 1.2: Observational learning consists of observing the demonstrations, extracting an abstract feature representations and executing the demonstrated task.

Observation

In the ‘Observation’ stage, the robotic system views the demonstrator performing the task. Demonstrations can be observed directly, i.e as raw images/videos or with assistance using trackers or visual detectors.

Abstract feature representation

In this stage, abstract feature representations of the demonstrations invariant to domain shifts are extracted. For example (referring to Figure 1.2), the representations for the task of opening a door should be identical, irrespective of the angle from which it was observed. The features are extracted frame by frame or from the video as a whole.

Execution

Execution is the last stage of observational learning where the robotic system performs the demonstrated task. It involves finding robotic controls to execute the task from the abstract representations of the demonstrations. The robotic system can obtain the controls for execution either by trial and error or using direct mapping techniques.

Given the breadth of the observational learning problem, in this thesis we limit the scope to simplify the problem with the following assumptions.

- Only manipulation tasks are considered.
- The tasks are non-collaborative, i.e the tasks are demonstrated by a single demonstrator.
- Demonstrations are viewed with a single RGB video camera.
- The demonstrations are provided by a human using one hand to perform a task on one or more objects, directly or with the use of a hand-held tool (e.g. a gripper).
- Once the demonstrations are given, the demonstrator does not provide any further assistance.

1.2 Motivation

A key challenge in observational learning approaches until recently was the difficulty to directly observe the demonstrations. The robotic systems had to be pre-programmed ‘what to observe’ in the demonstrations. This was implemented using trackers, visual detectors or motion capture mechanisms (covered in chapter 2). However, this limited the scope of observational learning to demonstrations that used pre-defined objects and also the demonstrators had the inconvenience of attaching on-body sensors or trackers. It was with the emergence of deep learning [46, 47] and its applications [48, 49], sidelining classical computer vision methods

[50, 51, 52], that direct observation of demonstrations became possible. In deep learning feature representations can be extracted directly from raw demonstration videos.

Deep neural network (DNN) feature extractors have been widely used in observational learning methods in the last few years as detailed in Chapter 2. However, the current methods lack generalisation to unseen manipulation tasks. The feature extractors are trained in a way that is dependent on the tasks. For each new task to be learned, the feature extractors have to be re-trained. This requires a large number of demonstrations for each new task to be learned. It is very time consuming to collect hundreds of demonstrations for each task especially in real world robotic application scenarios. This thesis addresses the problem of learning a task from a single demonstration. We propose a task-independent abstract feature representation extraction method that can generalise to unseen manipulation tasks without re-training.

Another challenging aspect of observational learning is the domain shifts between the demonstration and learning environments. As described earlier, domain shift can be changes in viewpoint of observation, object properties, scene background or morphology of manipulator. Humans have a remarkable ability to handle domain shifts. For example, we can learn cooking from internet videos, even if the learner has different types of utensils or cooking apparatus from that of the demonstrator. This ability comes from an abstract level understanding of the task by humans, which is independent of variables such as the object properties or morphology of the manipulator. However, equipping robotic systems with this ability is technically challenging. The existing methods handle domain shifts in constrained settings. For example, Liu et al. [53] used tools to handle domain shift in morphology of the manipulators. Similarly work by Sermanet et al. [54] used the assumption that the viewpoint of observation remains the same. A tabular review of domain shifts handled by different methods is given in chapter 2. In this thesis we aim to extract more robust domain-agnostic abstract feature representations that can handle unconstrained domain shifts.

Also, the existing methods [40, 55] perform frame by frame extraction of features from the demonstration videos. These features are then concatenated to represent the demonstrated task. A consequence of such representations is that it

places higher emphasis in the path taken to perform the task than the task itself. This restricts the freedom of the robotic system to follow a different trajectory than what is demonstrated. For example, in a task such as pushing an object, the task can be considered complete if the object is pushed into the target region irrespective of the path taken.

Similarly, the demonstration might contain states that are not achievable by the learner as pointed out by Lee et al. [56]. Forcing to follow the demonstrated path in such cases will prevent task learning. In these scenarios, the robotic system should have the freedom to deviate from the trajectory followed by the demonstrator.

1.3 Aim and objectives

Based on the issues and challenges outlined in the previous section, we define the aim of our research as: *‘Develop an observational learning method that requires only a single demonstration (one-shot) of the task to be learned’*. Note that, one-shot does not refer to the number of trial and error executions by the robot during learning from that single demonstration. The objectives for achieving this are outlined below:

- **Develop an abstract feature representation extraction method that can generalise to unseen manipulation tasks without re-training.** This will eliminate the need for re-training the feature extractor for each new task to be learned. The features are intended to be extracted from the video as a whole, rather than individually from each frame.
- **Develop a domain-agnostic feature representation method for the tasks.** A domain-agnostic feature representation of the tasks is essential to handle the domain shifts between demonstrator’s and learner’s environments.
- **Integrate the developed feature representation method into a learning framework for robotic manipulation tasks.** The feature representation of demonstrated tasks can be used to generate a reward/cost to be optimized during task learning.

- **Design and implement experimental setups and evaluation methods.** Evaluations are to be performed for: (A) understanding generalisation of the feature representation method to unseen manipulation tasks and (B) testing the performance of the overall observational learning system developed.

1.4 List of contributions

With respect to the aim and objectives laid out in section 1.3, we present our research contributions as follows:

- We developed O_2A (**O**ne-shot **O**bservational learning with **A**ction vectors), a method that learns new manipulation tasks from a single third-person demonstration. O_2A works by extracting an abstract feature representation of tasks from the demonstration videos. We call it the '*action vector*'. Action vectors are extracted with the novel approach of using a 3D convolutional neural network (CNN) pre-trained on a generic action dataset for action recognition. The pre-trained action vector extractor generalises to unseen manipulation tasks by learning the shared underlying visual dynamics during pre-training. Action vectors are domain-agnostic which handles the domain shifts between the demonstrator's and the learner's environment. The distance between the action vectors from the observed third-person demonstration and trial robot executions is used as a reward/cost for learning of the demonstrated task.
- We collected an evaluation dataset of manipulation tasks, the Leeds Manipulation Dataset (LMD), for evaluating the performance of the pre-trained action vector extractor on unseen manipulation tasks. We conducted clustering analysis, calculated class similarity scores and also visualised the action vectors from LMD. We also designed novel experiments for testing O_2A both in simulation and with a real robot. We used the tasks of reaching and pushing (in simulation) and pushing, hammering, sweeping and striking (with the real robot). We evaluated O_2A by plotting trajectory maps, measuring task completion measures and calculating correlation with oracle rewards.

1.5 Publications

Our research has been published/presented at the following venues.

Journal

L. Pauly, W. C. Agboh, D. C. Hogg, and R. Fuentes, "O₂A: One-shot Observational learning with Action vectors", at *Frontiers in Robotics and AI (Accepted)*, 2021[57]

Workshops

"One-shot observation learning", at *IROS Workshop: Examining Sensing Modalities for Robust and Dexterous manipulation*, IROS, 2018.

"One-shot observation learning using visual activity features", at *3rd UK robotics manipulation workshop*, 2019.

"One-shot observational learning", at *AI @ Leeds workshop*, 2019.

1.6 Thesis summary and outline

The thesis summary is illustrated in Figure 1.3 and an outline of each chapter is given below:

Chapter 1: Introduction

The chapter introduces and defines the concept of observational learning. Research motivation, aim and objectives, list of contributions and outline of the thesis are also presented in the chapter.

Chapter 2: Literature review

In this chapter we present a comprehensive survey of existing observational learning literature. The introductory section explains when observational learning can be used. Subsequent sections describe methods and techniques used in the different stages of observational learning.

Chapter 3: Action vectors

In this chapter we explain the concept of action vectors. We present the action

vector extractor network architecture, the generic action dataset used and other pre-training details. We also present details of LMD and report on experiments conducted.

Chapter 4: One-shot observational learning

In this chapter we present O_2A in detail. We explain how the action vectors are used in the one-shot observational learning method. We also report on experiments conducted both in simulation and on a real robot.

Chapter 5: Conclusion and future work

The final chapter presents a summary of the research. It also discusses the limitations and future directions of the presented research.

**

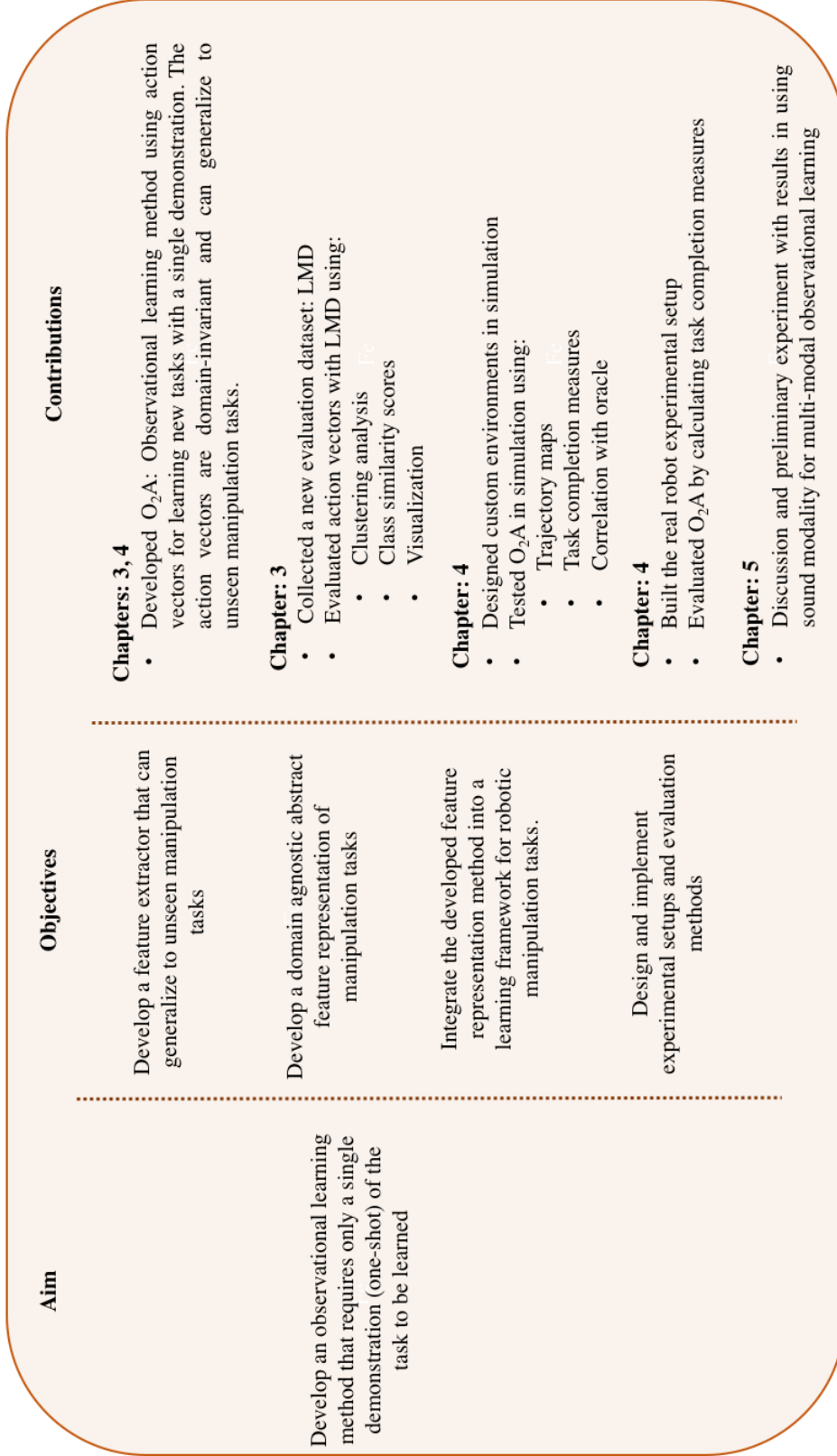


Figure 1.3: Thesis summary

Observational learning of manipulation tasks has been at the center of robotics research for a very long time. In this chapter we provide an extensive review of existing approaches for observational learning of manipulation tasks in the literature.

2.1 Introduction

Robotic systems can learn manipulation tasks in many different ways. The methods will vary based on the mode of supervision and guidance they use. Figure 2.1 presents the different approaches in the literature for learning manipulation tasks and when to use them in the form of a flow chart. Standard learning methods are used when a reward (or a cost) function is available [58, 59, 60, 61, 62, 63]. If the reward function is not available the next option is to look for the availability of task demonstrations. In the absence of demonstrations, external human assistance (for example, binary goal completion queries) is requested [64, 65, 66]. If the demonstrations are available, Learning from Demonstration (LfD) [27] can be used. LfD approaches fall under two paradigms: imitation learning and observational learning. The nature of the demonstrations available determines which of these two methods is to be used. If first person demonstrations are available, imitation learning methods are to be used [67, 68, 69, 70, 71]. Imitation learning approaches can be broadly classified into Behavioral Cloning (BC) [67, 68, 70] and

Inverse Reinforcement Learning (IRL) [69, 72] methods. In BC a control policy is learned directly from the demonstrations. In IRL, a reward function is first deduced from the demonstrations and then a control policy is learned by optimizing this deduced reward function. However, a direct imitation of the demonstration is insufficient if the observations are made from a third person point of view. Third person observations are identified by a domain shift between the demonstration and the robot's learning environment. Observational learning [31, 32, 33] approaches are used in such scenarios. These approaches handle the domain shift by extracting domain-agnostic abstract feature representations which are used for learning the demonstrated task.

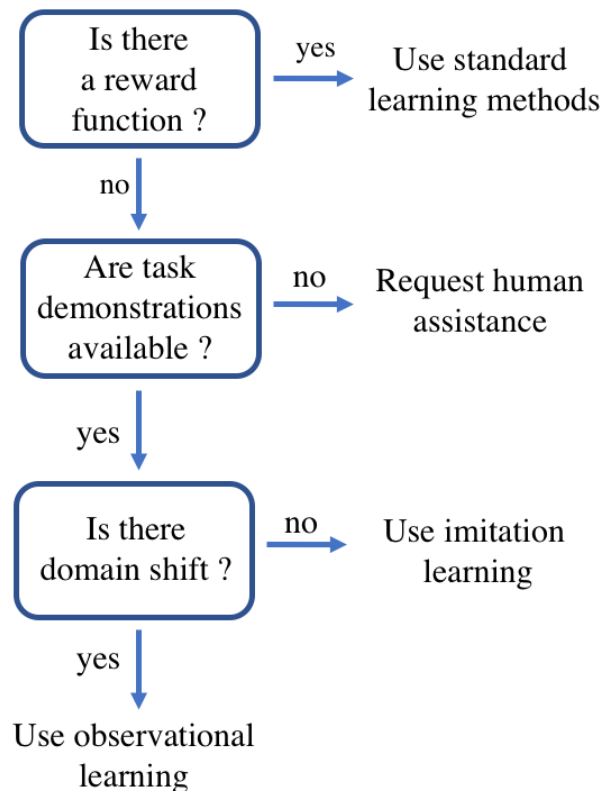


Figure 2.1: Different approaches used for learning robotic manipulation tasks. Observational learning approaches are used when demonstrations with domain shifts are provided for learning tasks.

In the upcoming sections of this chapter, we present a comprehensive review of the existing approaches in observational learning. We analyse different methods and techniques used at different stages of observational learning: observation, abstract feature representation and execution. Figure 2.2 presents this overall organisational structure for the chapter. In section 2.2, the first stage of observational learning is surveyed. The section discusses two ways of observing demonstrations: assisted and direct. Section 2.3, discusses in detail the different approaches of feature representations and extraction from observed video demonstrations. And in Section 2.4, different approaches for executing the learned task are discussed. In Section 2.5, we discuss how our method fits into the existing literature and Section 2.6 presents a qualitative comparison of the existing methods with O₂A. We compare in terms of number of demonstrations required for learning a new task and types of domain shifts handled. A direct quantitative comparison is difficult due to the diversity of approaches used and lack of standardised evaluation metrics. The chapter is summarised and concluded in section. 2.7

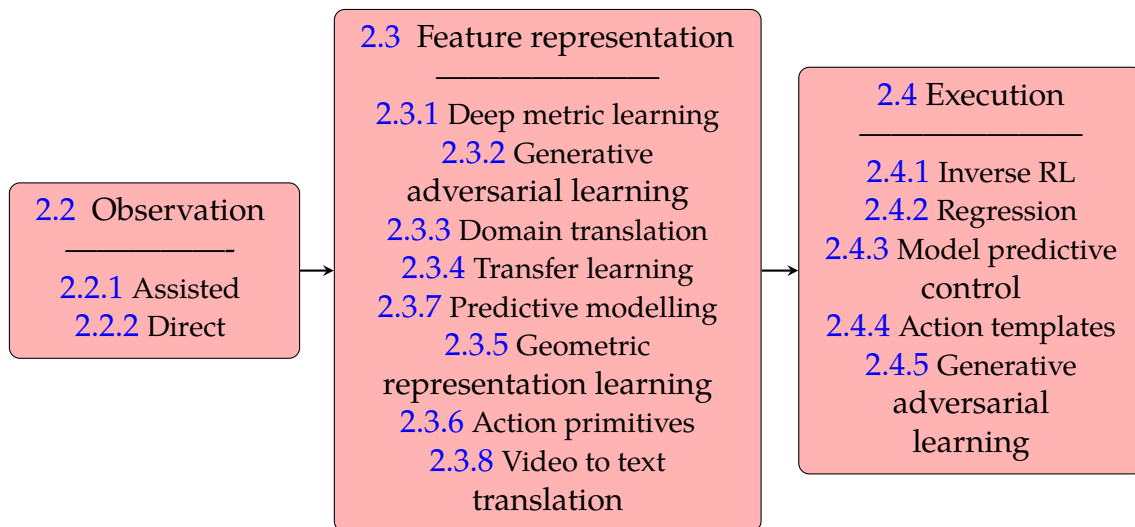


Figure 2.2: Organisational structure of remaining sections in the chapter.

2.2 Observation

Observation is the process of observing the demonstration. It includes observing items (objects and manipulators) and their interactions which are relevant in understanding the demonstrated task. Observations can be of two types: assisted and direct. In assisted observations the robotic systems are explicitly provided with assistance in identifying relevant items and interactions to observe. Whereas in direct observation, videos are observed without any assistance. The feature representations are then extracted from these raw videos.

2.2.1 Assisted observation

Assisted observation is the classical way of observing demonstrations. The assistance is provided with sensing techniques such as trackers, visual detectors, motion capture systems, skeleton tracking or a combination of the above. Each method is explained briefly below and a comparison is shown in Table 2.1.

Using trackers

Motion capture [3, 73, 74, 75] : Motion capture (illustrated in Figure 2.3) records motion of bodies (objects, humans or animals) using different sensors (inertial, electro-mechanical or acoustic) placed on the body to be tracked. It has wide applications in fields such as entertainment, sports, medical treatments, computer vision, graphics and robotics. It was one of the most widely used methods for assisting observation in the earlier days of research into observational learning due to its high accuracy. However, the requirement of expensive body-worn instrumentation and inability to work on demonstration videos from the internet are major drawbacks for this method.

Trackers[4,76]: In this method special tags or markers (for example, QR codes) are used to identify and locate the objects to detect and track in a demonstration video. Gupta et al.[4] use LED trackers (shown in Figure 2.4) for recording the positions and velocities of the manipulated objects and in [76] magnetic trackers are used for recording object movements in the demonstrated videos. Tracker based methods are comparatively easier to implement and have relatively higher

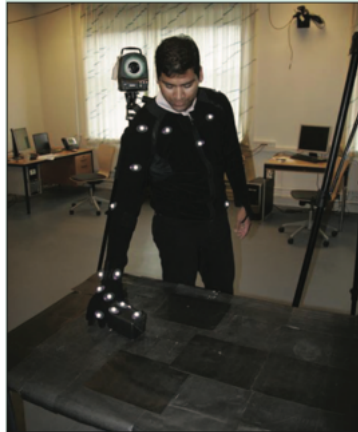


Figure 2.3: Motion capture system. Note the inconvenience for the demonstrator in wearing on body sensors. Figure from [3]

accuracy compared to the other methods.

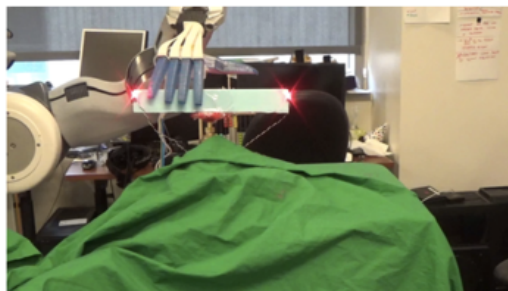


Figure 2.4: LED trackers used for recording the positions and velocities of the manipulated objects. Figure from [4].

Visual Detection

Visual detectors [5, 77, 78, 79, 80, 81, 82]: Here, pre-defined visual detectors using computer vision algorithms (as shown in Figure 2.5) are used for assisting observations. Simplicity of usage, implementation easiness and compatibility to use with demonstration videos from the internet make this method popular. Also the accuracy of the visual detectors have increased considerably with implementation of deep learning based object detectors [83, 84] compared to the classical methods based on hand-crafted features [85, 86].



Figure 2.5: Visual detectors used for assisting observations. Figure from [5]

Skeleton tracking [6, 37, 87]: Skeleton tracking is used for tracking human motion and pose estimation. It does not require any on-body sensors and instead uses computer vision algorithms for tracking human motion. An example of full body skeleton tracking is shown in Figure 2.6. Like visual detectors, skeleton tracking has also seen considerable increase in performances by the use of deep neural networks [88, 89, 90]. Also recently, more sophisticated approaches that can track even partial body parts (for example hand with fingers [91]) with higher accuracy levels have been developed.

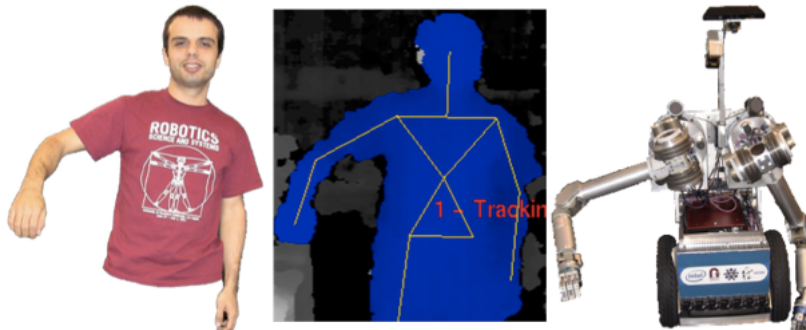


Figure 2.6: Full body skeleton tracking of the task demonstrator. Figure from [6].

Hybrid methods

Combination of the above [92]: Here a combination of two or more of the above techniques is used to assist the robotics systems in the process of observation. In [92] a combination of motion capture techniques with visual detectors is used for

observing demonstrations. Such combinations could improve overall efficiency and help to overcome the shortcomings of using a single method.

Table 2.1: Advantages and disadvantages of different methods used for assisted observation

Method	Advantages	Disadvantages
Motion capture	<ul style="list-style-type: none">• Very high accuracy	<ul style="list-style-type: none">• Expensive• Require specialised instrumentation and settings• Demonstrator has to wear on-body sensors or suits
Visual detectors	<ul style="list-style-type: none">• Easy to implement as detection is performed by the computer vision algorithms• Low cost of implementation• Does not require additional hardware or devices	<ul style="list-style-type: none">• Moderate precision and accuracy
Trackers	<ul style="list-style-type: none">• Can be set up with low cost trackers like QR codes and colour tags	<ul style="list-style-type: none">• Moderate precision and accuracy• Inconvenience of wearing additional tags and markers by the demonstrator
Skeleton tracking	<ul style="list-style-type: none">• Easy to implement• Additional hardware or devices not required	<ul style="list-style-type: none">• Can be used only with human demonstrations• Moderate precision and accuracy
Hybrid methods	<ul style="list-style-type: none">• Higher accuracy• Can overcome drawbacks of using methods individually	<ul style="list-style-type: none">• Implementing an hybrid pipeline for observation will require expertise with multiple methods

2.2.2 Direct observation

One major drawback for the assisted observation methods is that its scope is limited to pre-defined objects and manipulators. The items (objects and manipulators) to be observed have to be known beforehand and only manipulation tasks using these items can be learned. For instance, a robotic system equipped with the hand skeleton tracking method cannot be used to learn tasks demonstrated with tools, as the tracking system will fail. So for learning each class of tasks, the

observation process has to be customised. This limits the scope of observational learning methods in real world applications. A solution for this is to observe the demonstrations directly without any assistance. However, this will make the observational learning problem harder. It will require intelligent feature extractors, which can extract relevant features directly from the raw demonstration videos.

Observational learning with direct observation is a new branch of research with one of the introductory works published in 2017 [39]. It was the emergence of deep learning [46, 47], that made direct observation of demonstrations possible. DNNs [93, 94, 95] can extract meaningful hierarchical features with different levels of abstraction directly from images and videos [96, 97, 98]. In section 2.3 we review how different observational learning approaches extract feature representations directly from video demonstrations.

2.3 Abstract feature representation

Abstract feature representation is the process of extracting abstract representation of tasks from the demonstration videos. These representations should be invariant to domain specific aspects such as viewpoint of observation, object properties, scene background and morphology of manipulators. At the same time, they should capture key aspects of the tasks such as intent and goal. A variety of deep learning networks and techniques have been used individually and in combination in literature. This section provides a review of different abstract feature representations used.

2.3.1 Deep metric learning

Deep metric learning [99] can be defined as learning the distance metric between pairs of examples in some embedded feature space, using a metric loss function. The characteristics of the metric function and feature space learned will depend on the metric loss function used. Some examples of metric loss functions are: contrastive [99], triplet [100], lifted-structured [101], N-pair [102], angular [103] divergence [104] and cross-entropy [105] loss functions.

The time-contrastive network (TCN) [7] uses deep metric learning with triplet

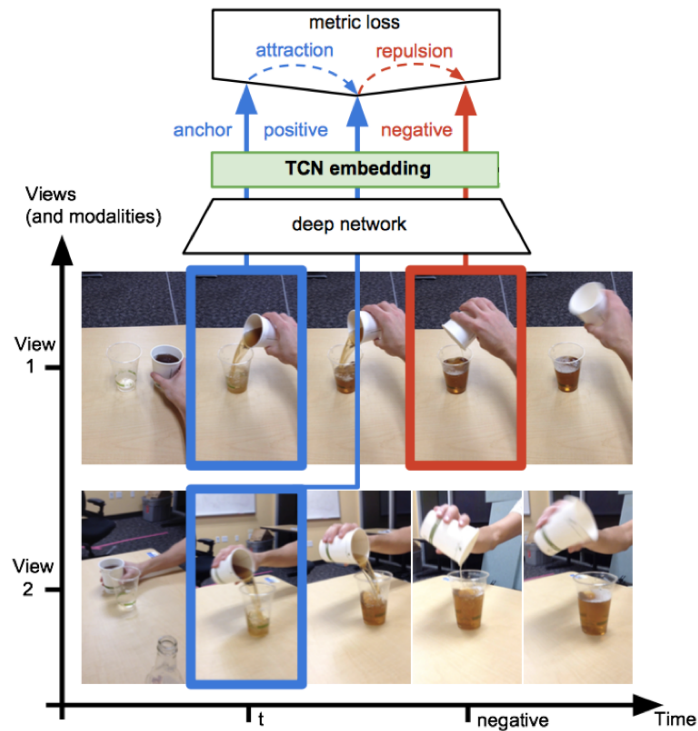


Figure 2.7: Training time-contrastive networks with triplet loss for feature extraction from demonstration videos. The anchor and the positive images are encouraged to be closer in the feature space, while being further away from the negative image. Figure from [7].

loss function to learn frame by frame feature representation of the demonstration video. The triplet loss function takes in a triplet of images called anchor, positive and negative images respectively. The anchor and positive examples with similar labels are encouraged to be close to each other in the embedding space and the negative example is forced to be further away. Here the anchor and positive images are taken from the same time-point from different viewpoints and the negative image is taken at a different time-point from the same demonstration as show in Figure 2.7. The frame by frame TCN embeddings from the demonstration video are concatenated and used as the feature representation.

2.3.2 Generative adversarial learning

Generative adversarial learning is a machine learning framework introduced by Goodfellow et al. [106]. The basic structure is illustrated in Figure 2.8. It consists of two competing functions (generally represented with DNNs) called generator (G) and discriminator (D) trained in an adversarial manner. The generator generates synthetic data (X') and the discriminator compares it with the real data (X). The generator is trained to generate synthetic data realistic enough to fool the discriminator, while the discriminator is trained to distinguish between the real and synthetic data. Upon convergence the generator learns to produce synthetic data which has a similar distribution as that of the real data.

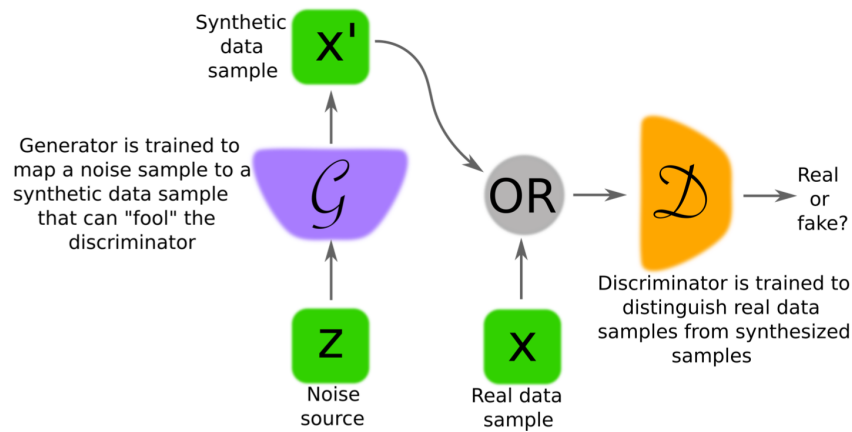


Figure 2.8: Basic generative adversarial learning framework. Figure from [8].

Stadie et al. [39] used domain confusion loss [107] with the generative adversarial framework to extract domain-agnostic features from the demonstration video. The domain confusion loss is generated by a classifier that attempts to distinguish samples coming from two different domains. Here the generator is the feature extractor and the discriminator is the domain classifier. The feature extractor tries to generate domain agnostic features from the demonstrations and the domain classifier tries to distinguish between observations coming from different domains and to maximise domain confusion loss. Consequently the feature extractor will learn to produce domain invariant features from the demonstrations.

2.3.3 Domain translation

In domain translation methods, feature extractors are trained for frame by frame translation of demonstrations from the domain of the demonstrator to that of observer. Figure 2.9 illustrates the concept of domain translation. Each frame in the coffee making demonstration video is translated from the demonstrator’s domain into observer’s (robot’s) domain at the pixel level. Note that the method accounts for the changes in the domain settings (such as manipulator morphology) that exist between the demonstration and the observer’s domain. Training to translate from one domain to the other, enables feature extractors to learn to extract domain independent features.



Figure 2.9: Domain translation. Images are translated from the demonstration domain (top row) to the observers domain (bottom row). Figure from [9]

Different approaches have been developed for feature extraction with domain translation using autoencoder networks [40], cyclic generative adversarial networks (CycleGAN) [9] and generator-discriminator networks [10]. Each of these approaches are reviewed in detail below.

Autoencoder

Autoencoders [108] are one of the most commonly used DNN structures for feature extraction. An autoencoder consists of two components: encoder and decoder. The encoder converts the input into an encoded feature vector representation. The inputs could be videos, images or even a 1D audio clip. The decoder

converts this feature vector back into the decoded form. The properties and the characteristics of the feature representation generated by the encoder can be customised by placing different constraints in the structure of encoder-decoder network and customising the loss functions used for training the autoencoder [108]. In [40], an autoencoder is trained to convert demonstration videos from demonstration domain into the observers domain. After training, the latent representations produced by the encoder network are used for learning the task execution.

CycleGAN

CycleGAN [109] is used for translating images between domains. The advantage of this method is that it does not require paired images from the source and the target domain and hence can be trained in a unsupervised manner. It learns two mappings, one from the source to the target domain and the other from the target to the source domain. Two discriminators are used to distinguish between the original and the translated images for each of these mappings. The mappings are then learned by optimising a combined loss function of these two discriminator losses and a cycle consistency loss [109]. Smithe et al. [9], uses the cycleGANs to translate demonstrations between the demonstrator domain to the observers domain. Contrary to the autoencoder based approach, the domain translated demonstration image is used in the execution stage.

Generator-discriminator network

In this approach a generator-discriminator network is used for feature extraction with domain translation. The generator-discriminator network is trained in a adversarial manner as described in section 2.3.2. Unlike the previous domain translation methods, the change in the demonstration video frames are translated instead of translating the entire image between the domains. The generator network is trained to translate the current frame to the next in the robot domain, the same way the corresponding frames in the demonstration video is translated. Thus the generator is forced to learn the pixel movements instead of having to learn to generate entire pixel distributions. Figure 2.10 illustrates the presented approach in [10]. A U-Net [110] based network architecture is used as the generator. The

discriminator loss with an additional $L1$ reconstruction loss [108] is used for training. The reconstruction loss ensures that correct frames are generated, while the discriminator loss ensures generation of realistic images.

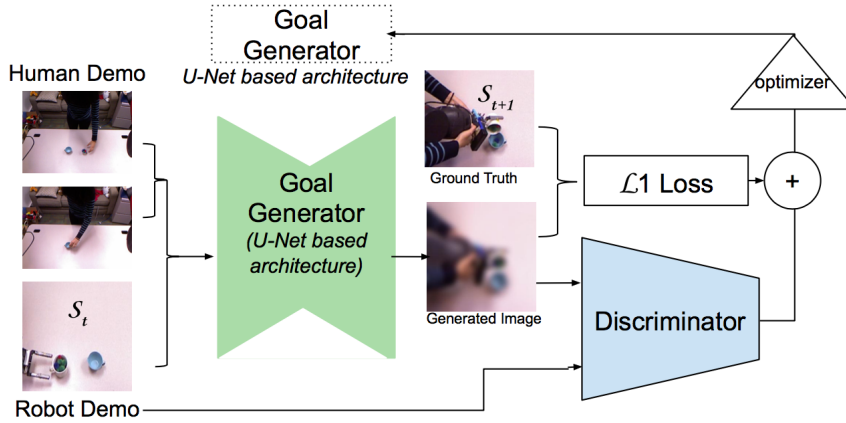


Figure 2.10: Domain translation method using generator-discriminator network.
Fig. from [10]

2.3.4 Transfer learning

Transfer learning [111] is used for solving problems in one domain by using knowledge gained from another closely related domain. Transfer learning has been widely used for feature extraction with DNNs. Pre-trained image feature extractors have been reused for generic visual feature extraction, in domains like robotic vision [112, 113], where large datasets are scarce or not available.

Sermanet et al. [54] use the Inception network [114] pre-trained for ImageNet classification [115] as the feature extractor. Similarly, pre-trained models from [116] are used by Sharma et al. [117], to extract features from demonstration videos.

2.3.5 Geometrical representation learning

Geometric representation learning is motivated by the human ability to represent manipulation tasks as a combination of geometric primitives. Geometric primitives can be defined as the simplest or irreducible geometric objects (with the attributes and characteristics of a physical object) that a system can handle [118, 119]. Points, lines, splines, planes and arcs are examples of geometric primitives.

Geometric representation learning methods infer geometric primitives and their association relationships from the demonstration video and use that to learn a feature representation of the task as illustrated in Figure 2.11.

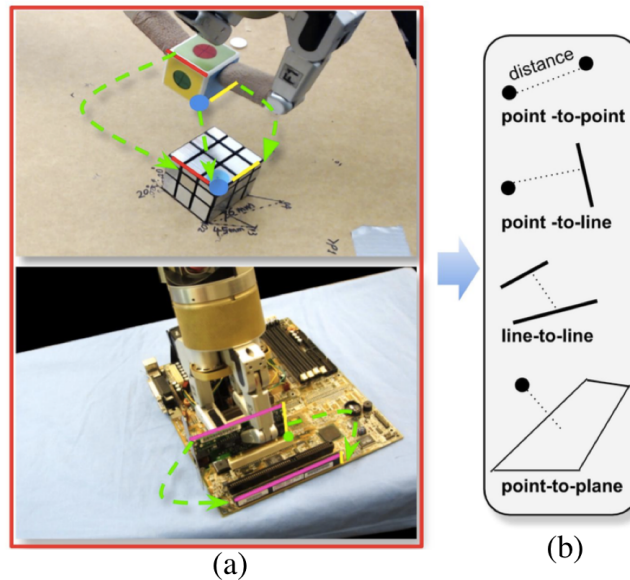


Figure 2.11: Representing insertion manipulation task in terms of geometrical primitives (a): points (coloured: blue) and lines (coloured: red, yellow, pink) and their associations(b). Figure from [11].

In [11, 38] constraint association [120] between the geometric primitives are learned with graph structured kernel functions. These graph kernels called *geometrical skill* kernels are composite function structures that describe the association constraint between geometric primitives. The graph nodes represent the geometric primitives (points, lines etc.) and graph edges represent the association between them. These learned primitive constraint associations are then combined in parallel and sequentially to represent the complex manipulation task in the demonstrated video.

2.3.6 Action primitives

Action primitives can be defined as the elementary building blocks of an action or an activity [3]. Every task can be broken down into these predefined action primitives. Figure 2.12 shows a few examples of different tasks (pick and place,

pushing and bottle opening) broken down into its pre-defined component action primitives.

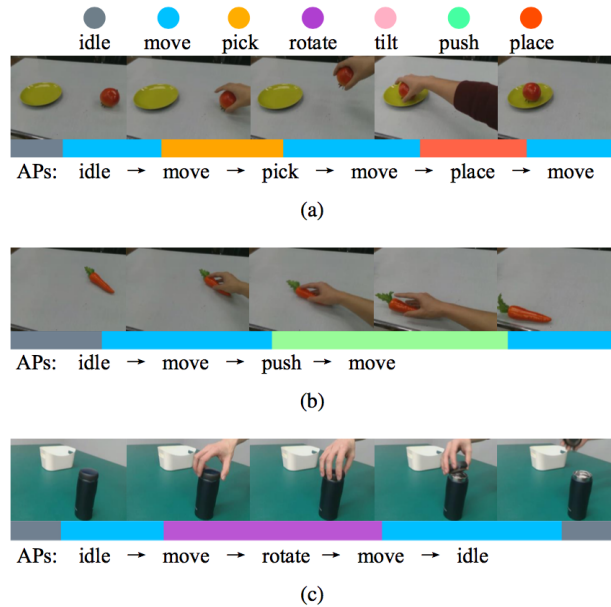


Figure 2.12: Decomposing actions into action primitives for task of (a) Pick and place (b) Pushing away (c) Opening bottle. Colour bars indicates the time duration of the action primitives in the task video. Figure from [12].

Jia et al.[12] use the concept of action primitives. Demonstrated tasks are decomposed into pre-defined action primitives for creating task representations. The pre-defined action primitives used in this work are: idle, move, pick, place, push, tilt and rotate, which are selected manually based on observing every day human activities. They are identified in the demonstration videos with an action primitive recognition network, built by stacking an LSTM [121] network on top of pre-trained CNN architectures.

2.3.7 Predictive modelling

Future-frame video prediction is a widely studied problem in computer vision [122, 123, 124]. It can be used as an unsupervised learning technique to learn meaningful representation from videos. It leverages the vast collection of available videos (example: internet videos) to learn features that can be used for several

applications like action recognition [124]. Recently video prediction has been extended to robotic task learning as action-conditioned video prediction [125, 126]. In action-conditioned video prediction, predicted future frames are a function of the current frame and also the action taken by an agent at the current time. This action-conditioned prediction can then be used to model the environment dynamics in reinforcement learning and other robot learning approaches. The proposed method in [127] uses video prediction to model the demonstration and action-conditioned video prediction to model the robot environment. Predicted frames by these models are then used to learn to execute the demonstrated task.

2.3.8 Video to text translation

In this approach, natural language descriptions of the video demonstrations are used to represent the demonstrated task. For that, first the problem of feature extraction is cast as a video captioning problem [128, 129]. Video captioning methods generates textual descriptions of the demonstrated task as shown in Figure 2.13, which can be converted into commands for a robotic planner [130]. A variety of video captioning approaches have been proposed recently utilising the advances in the field of deep learning [131, 132].

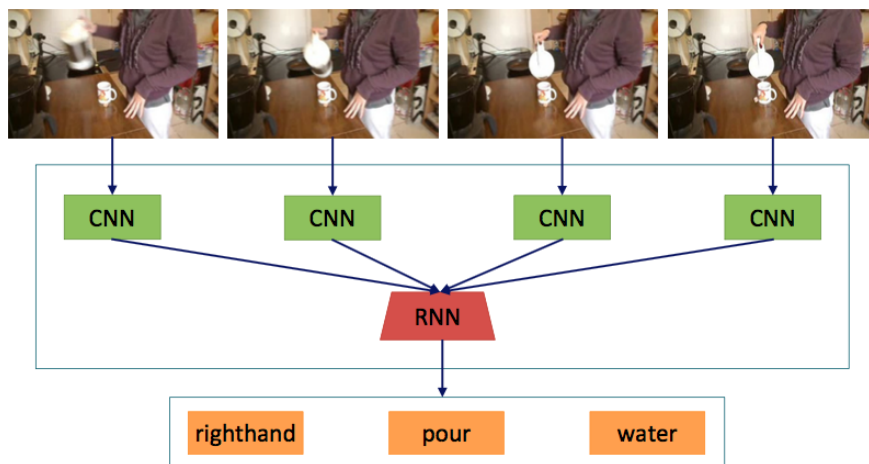


Figure 2.13: Textual descriptions from demonstrations. Figure from [13]

Nguyen et al.[13] use a modified video captioning method to generate captions for the video demonstrations. Contrary to the normal video caption methods, this

approach generates video captions in a grammar free form instead of the natural language form. The video caption network architecture is based on the encoder-decoder architecture [133] adapted from the machine translation sequence-sequence [134] network structure. The video caption system is trained using a custom dataset, the 'video-to-command' (V2C) dataset. The dataset is created by using videos taken from the 'Breakfast' dataset [135]. These videos (2-3min long) are segmented into short clips (10-15 seconds long) and annotated with a caption that describes the human action in the clip. These annotated videos are then used for training the video captioning system in a supervised manner.

Yang et al.[136] further improve on this video captioning approach to make it more adaptable to robotic manipulation tasks. This method extracts both global and local features from the demonstrated videos, and then fuses them together and use for video caption generation. The global features are extracted directly from each frame and local features are extracted from the 'video difference map' (VDM) obtained by subtracting first and last frames of the demonstration video.

2.4 Execution

Execution is the last stage of observational learning. It involves finding robotic manipulator controls to execute the demonstrated task by utilising feature representations obtained in the previous stage. Controls are obtained with learning approaches like reinforcement learning, motion planning methods like model predictive control or even by direct regression. In this section we review these different approaches employed in existing literature for task execution.

2.4.1 Inverse reinforcement learning

IRL [137] is a commonly used approach for learning controls in robotic manipulation tasks. IRL formulates observational learning as finding the solution to a Markov decision process (MDP)[138] using reinforcement learning (RL). An MDP is represented with a tuple (S_t, A_t, S_{t+1}, r) , where S_t is the state of the environment at time t , A_t the action taken on the environment (by the robotic manipulator), r is the reward obtained for taking the action and S_{t+1} is the new state.

In the observational learning scenario, solving the MDP means to find a function i.e. a control policy ($\pi(S|A)$) to control the robotic manipulator to perform the demonstrated task.

RL methods have been successfully used to solve MDPs [139], given a reward function. In IRL the reward (r) function is inferred from the task demonstration videos. The reward function may be learned [54] or engineered [55] and may not be the actual reward that the demonstrator was trying to optimize while providing the demonstrations. The inferred reward function is only an estimate of the actual reward. These rewards can then be used with a standard reinforcement learning algorithm to solve the MDP (i.e. to learn the task). Based on the methods used for generating rewards IRL methods can be divided into reward engineering and reward learning methods.

IRL with reward engineering

In this approach, a user-defined reward function is used to learn the demonstrated task. In [40], the reward function is the negative of the euclidean distance between extracted features from the demonstration and the robot state (as observed by the robot) at each time step. Sermanet et al. [55] further enhance this reward formulation by adding a Huber-style loss [140] to the squared euclidean distance. The euclidean distance will provide the RL algorithm with stronger gradients when the features are further away at the beginning of the learning process. The Huber-style loss comes into play when the features get close to each other as the learning progresses, ensuring fine-tuning of the learned motion at the end.

IRL with reward learning

In reward learning, a learning approach is used to learn the reward functions, instead of manually engineering them. In [54], Maximum Entropy (MaxEnt) IRL [141] method is applied to learn rewards as a function of deep visual features extracted from the demonstrations. The demonstration videos are first divided into segments, where each segment refers to an intermediate step or a sub-goal. The number of segments is treated as a hyper-parameter in this work. However, unsupervised video segmentation methods [142, 143] can also be used. Then a

quadratic reward function is fit into each of these segments that provides rewards as a function of the visual features extracted from the video segment. Figure 2.14 shows reward values from a learned reward function, for a successful and failed pouring task execution.

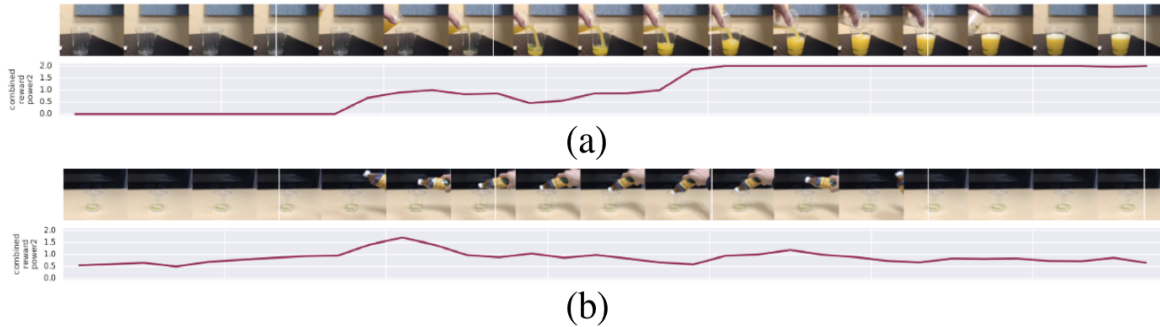


Figure 2.14: Values from a learned reward function for videos depicting (a) successful and (b) failed execution of the pouring task. It can be seen that the reward value drops, as the task fails to execute in (b). Figure from [13]

2.4.2 Direct regression

Direct regression refers to the direct mapping of the feature representations from demonstration videos to the robotic controls. The challenging aspect of this approach is that it requires a dataset with third person demonstrations and corresponding first person robotic controls for executing the task, to learn this mapping. These datasets are scarce and practically hard to collect.

Sharma et al.[117] collect such a dataset. It consists of 8260 videos of human-robot demonstrations over 20 tasks ranging from the simple pushing tasks to complex stacking tasks. The dataset consists of task demonstration videos observed from different viewpoints, and corresponding first person kinesthetic trajectories for the robot executions. This dataset is then used to learn a direct mapping of visual features from the demonstrations to the robot trajectories using an LSTM based network architecture in a supervised manner.

The direct regression approach can further be extended to develop end-end learning methods (referred as ‘pixel to torque’ methods [112]) for observational

learning. This will eradicate the need for the intermediate feature representations and relevant features will be extracted implicitly from the video demonstrations.

2.4.3 Model predictive control

Model predictive control (MPC) [144] is an optimal control approach from control theory used for motion planning in robotic manipulation tasks [145, 146, 147]. MPC methods works on the assumption of a known world model. The model of the world can be numerically derived [148], modelled in simulation [149] or learned interactively [150]. A general framework of MPC systems is illustrated in Figure 2.15. Model predictive control first uses the known world model to perform offline planning for a certain number of steps into the future, called ‘imaginary rollouts’. The cost/reward for each of these imaginary rollouts is calculated and an optimal control sequence is selected. MPC then executes the first step in this selected optimal imaginary rollout. Upon receiving the observation/measurements of the new state, MPC performs the imaginary rollouts again. This process of re-planning using imaginary rollouts and executing single step actions goes on until the robotic system has obtained an optimal trajectory for executing the desired manipulation task.

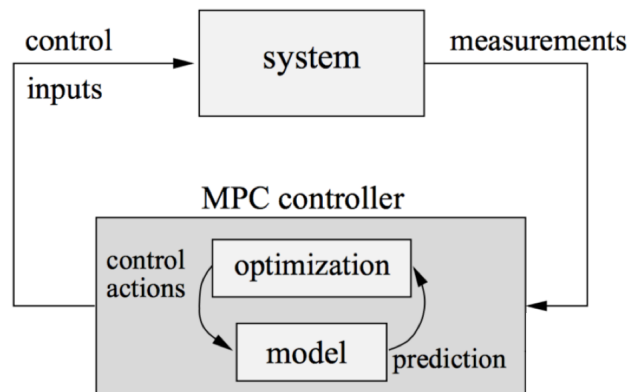


Figure 2.15: Illustration of a general MPC system . Figure from [14]

In [9], a learning based approach with the features extracted from the demonstrations is applied for task modelling. A probabilistic temporally-structured latent variable model learned with amortized variational inference [151] is used

here. This model is utilised by the latent-space MPC [152] algorithm with a sampling based optimisation procedure (the ‘Cross Entropy Method’ [153]), to search for the optimal sequence of actions.

2.4.4 Action templates

In this approach the demonstrated task is performed by invoking one or more combinations of pre-defined action templates (actions or action primitives), which the robotic system already knows how to execute. Action templates only require details like location or orientation of the objects to be manipulated to execute the corresponding action [154].

In [13], textual descriptions in the grammar-free form are extracted from the demonstration videos. These textual descriptions are directly mapped into robotic commands, which are used to invoke corresponding action templates. The relevant objects and their affordance are detected using a AffordanceNet framework [155]. A similar approach of generating textual robot commands is also used in Yang et al.[136]. A grasp detection network is applied to provide grasp solutions (grasp rectangle center and orientation) for the objects to be manipulated. Task templates are executed using these grasp solutions of the objects. An action primitive recognition network is employed in [12] to directly recognise action templates from the demonstration videos. Here a Mask-RCNN [156] network is used to detect objects and extract their binary segmentation masks from the video frames. Object poses to be manipulated are then extracted from these binary object masks.

These action template based approaches for task execution, can further benefit from the rich literature of knowledge representation based robotic manipulation methods [157]. It can facilitate integration of symbolic and geometric planning [158] for observational learning of more complicated multi-step tasks. However, this category of execution approaches suffers from the limitation that only pre-defined action templates can be executed by the system. In scenarios where a task that cannot be represented with the action templates is demonstrated, the system fails to execute them. Also this approach cannot be generalised for executing tasks with unseen objects during the training of object, affordance or grasp detection

networks. For example, failure cases are reported by Jia et al. [12] when the object detectors failed to detect objects in the scene to be manipulated and also when the action primitives extracted from the demonstration are incompatible with the pre-defined action templates.

2.4.5 Generative adversarial learning

Basics of generative adversarial learning approach is already explained in section 2.3.2 of this chapter. It involves training a generator and a discriminator in an adversarial manner. This concept is applied to learn control policies by [39]. The generator here is a policy gradient method (TRPO [159]), that generates control policies to execute the demonstrated task. The discriminator tries to distinguish between the demonstrator (expert) and robots (novice) trajectories. The trajectories here refers to the sequence of domain-agnostic task aware features extracted from the videos of the demonstration and of the robot executions. Upon convergence the policy gradient method will generate an optimal policy that can successfully execute the demonstrated task.

2.4.6 Meta-learning

Meta-learning is the process of learning to learn tasks [160]. It has been used widely in robotic manipulation learning problems to learn new tasks with fewer examples by leveraging previous task learning experiences [161]. Recently, Finn et al. developed the Model Agnostic Meta-learning (MAML) [15] algorithm, a general meta-learning approach compatible with any other gradient descent model (hence model agnostic). MAML aims to find a set of highly adaptable parameters θ , such that the parameters are closer to the optimal value θ_i , for a gradient step with respect to task i , as shown in Figure 2.16.

Yu et al. presented Domain Adaptive Meta-Learning (DAML) [162], by extending MAML method into observational learning settings. In [163], Yu et al. further extends DAML to observational learning of long horizon tasks. Here, the task to be learned is decomposed into component tasks called, primary skill tasks, which the robot had already seen during the meta-training phase. Each of these

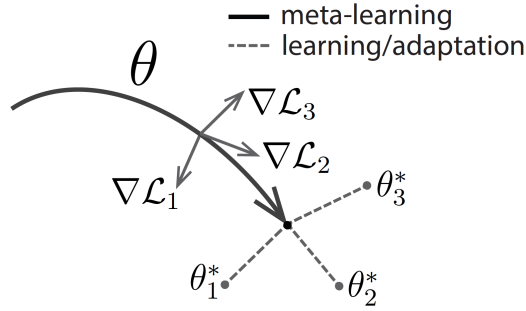


Figure 2.16: MAML algorithm illustration. Figure from [15]

primary skill tasks are then learned sequentially to learn the demonstrated long horizon (multi-step) task.

2.5 Discussion

In this section we discuss how the proposed method, O_2A , connects with existing literature. Works by Liu et al. [40] and Sermanet et al. [55] can be considered as the closest predecessors to O_2A . We adopt the general framework presented in these works for implementing O_2A . Here the demonstrations and the trial robot executions are projected into a perceptual feature space and the distance between them are used as the reward/cost to be optimised. The primary differentiator in our method is that it requires only a single demonstration, while the other approaches require large number of demonstrations per task as shown in Table 2.3. Additionally our method can handle a variety of domain shifts, without the need for constraining the domain shift, for example by using tools as in [40].

Furthermore, our feature extraction method closely relates to the transfer learning methods used in [117] and [54]. The key difference lies in using a generic human action dataset as the pre-training dataset and a 3D CNN architecture for the feature extractor. This allows for the capture of the spatio-temporal relations in a task demonstration video, which is crucial for representing the visual dynamics of a manipulation task. The existing approaches use an image-based transfer of features, applied frame by frame to demonstration videos, which cannot capture the temporal connections. Finally, O_2A employs an IRL with reward engineering ap-

proach [40] [55] for learning the robotic controls for executing the demonstrated tasks. As mentioned above, the distance between the demonstration and the trial robot executions in the feature representation space is used as the reward. This reward is then used by a standard RL algorithm for learning controls. However, it would be an interesting future work to learn the reward function (instead of engineering them) as used in [54].

2.6 Comparative study

Even though several observational learning methods have been proposed in literature, a direct comparison of these methods to O₂A is difficult due to the following reasons:

- The field's wide scope and diversity in the approaches used. For example, it is hard to compare the TCN feature representation [7] with the textual representation of tasks [13], as one is a numerical feature vector and the other being a textual sequence.
- Lack of standardised metrics and evaluation protocols make a direct quantitative comparison not possible. For example, the task of reaching can be evaluated as the ratio of the initial and final distances between the manipulator and the target region [40]. However, the task completion can also be evaluated by checking if the manipulator reaches within a certain radius around the target [10]. A set of standard evaluation protocols are necessary to enable a fast and direct comparison of different methods, without the need for re-implementing each one of them from scratch.
- Moreover, different approaches use different robotic hardware, simulation environments and programming frameworks. This makes reproducibility of works by a third party all the more difficult. An interesting future work would be to build a cloud based evaluation task-suite with a leaderboard where different observational learning methods can be deployed and compared under identical robotic manipulators and environments.

Lynnerup et al. present a detailed survey [164] of such issues in deep reinforcement learning for robotics in general, which are also applicable to observational learning. Hence in Tables 2.2 and 2.3, we present a qualitative study of observational learning methods that are comparable to our method, O₂A. Table 2.2 shows approaches used in each stage of observational learning. And in Table 2.3 (column 2), the comparison is based on the number of demonstrations required for learning a new task, including the demonstrations required for training feature representation extractors. It shows that it will require far greater number of demonstrations if we were to use the existing approaches instead of O₂A. The requirement of substantially larger number of demonstrations required in the existing methods is because the feature extractors need to be trained separately for each task. Whereas our method has already been pre-trained on a generic dataset. We believe that what we show is more general than direct comparison to one method only. We also show different domain shifts tackled by each method. Our method does not use any robot data for training the feature extractor and also works well under different domain shifts.

Table 2.2: Observational learning methods in existing literature are compared, based on approaches used in different stages.

Reference	Observation	Abstract feature representation	Execution
[55]	Direct	Metric learning	Inverse reinforcement learning
[39]	Direct	Generative adversarial learning	Generative adversarial learning
[40]	Direct	Domain translation	Inverse reinforcement learning
[9]	Direct	Domain translation	Model predictive control
[10]	Direct	Domain translation	Direct regression
[117]	Direct	Transfer learning	Direct regression
[54]	Direct	Transfer learning	Inverse reinforcement learning
[12]	Direct	Action primitives	Action templates
[127]	Direct	Predictive modelling	-NA-
[136]	Direct	Video to text translation	Action templates
[162]	Direct	Meta-learning	Meta-learning
[163]	Direct	Meta-learning	Meta-learning
O ₂ A	Direct	Transfer learning	Inverse reinforcement learning

Table 2.3: Observational learning methods in existing literature are compared. O₂A requires only a single demonstration to learn new tasks. It does not use any robot data for training the action vector extractor. And also works well under different all domain shifts.

Reference	Number of video demonstrations required per task (Including to train the feature extractor/s)	Is robot data required for training the feature extractor/s ?	Viewpoint invariant?	Invariant to changes of object properties ?	Invariant to changes in scene background ?	Invariant to changes of morphology of the manipulator ?
[55]	~40 min of human demonstrations + ~20 min random robot manipulation data	✓	✓	✓	✓	✓
[39]	An expert policy is used instead of direct demonstrations	✓	✓	✓	✓	✓
[40]	~60-3000 human demonstrations using additional tools	✗	✓	✓	✓	✗
[9]	~20-30 human demonstrations + ~300-500 random human and robot images	✓	✗	-NA-	-NA-	✓
[10]	~230 human demonstrations + corresponding robotic joint angle data	✓	✓	✓	-NA-	✓
[117]	~200-400 human demonstrations + corresponding robotic joint angle data	✓	✓	✓	-NA-	✓
[54]	~12 human demonstrations	✗	✗	✓	-NA-	✓
[12]	~50-100 human demonstrations	✗	✓	✓	✓	✓
[127]	Uses both human and robot task demonstrations (exact numbers unknown)	✓	✓	-NA-	-NA-	✓
[136]	~2990 human demonstrations	✗	✓	✓	✓	✓
[162]	1 (But uses closely related supplementary task demonstrations. Requires ~600-1200 robot and ~600-1200 human demonstrations per task)	✓	✓	✓	✓	✓
[163]	1 (But requires large number of action primitive demonstrations. ~600-1200 robot and ~600-1200 human demonstrations per action primitive)	✓	-NA-	✓	✓	✓
O ₂ A	Only 1 demonstration (human demonstration with or without using additional tools)	✗	✓	✓	✓	✓

2.7 Conclusion

In this chapter we have reviewed previous and ongoing research in the field of observational learning. Methods and techniques used in observation, abstract feature representation and execution stages are discussed. In the introductory section, we present the motivation for using observational learning. Observational learning is used when third person demonstrations are available for learning a task.

In the second section, we discuss different approaches for observing demonstrations: assisted and direct observation. In assisted observation, external assistance with sensing techniques like motion capture and visual detectors are used. However, these methods have the disadvantage that their scope is limited. Direct observation of demonstrations can overcome this drawback.

The third section discusses different approaches for extraction of feature representations directly from raw video demonstrations. Even though several techniques have been proposed ranging from domain translation to geometric representation learning, there is still room for improvement. Reducing the number of demonstrations needed to learn a new task and ability for handling different domain shifts are two areas to be further investigated.

In the fourth section, we review different methods for task executions. It involves exploring and finding the controls for the robotic manipulator to execute the demonstrated task. IRL is the commonly used approach in this stage. Direct regression, pre-defined action templates, generative adversarial learning and planning approaches have also been used.

In the final section, we present a qualitative comparison of existing methods for observational learning and O_2A . While existing approaches need multiple video demonstrations to learn a new task, O_2A requires only a single demonstration. Also, our method can handle different domain shifts. We achieved this by devising a novel approach of using a 3D CNN pre-trained for action recognition on a generic action dataset, for extracting an abstract feature representation of tasks from the demonstration videos.

**

3.1 Introduction

Neuroscience studies [165, 166] in monkeys show that there exists a type of neurons called ‘mirror neurons’ in the premotor cortex (area F5) of the brain. These neurons were found to be activated when an individual observes its own action execution as well as when it observes the same action performed by another individual. Further studies [167] have given evidence for the existence of mirror neurons in human beings and that the action perception and action execution share common neural structure involving these mirror neurons. Italian neurophysiologist Rizzolatti suggests that this visual-motor interconnection plays an important role in human observational learning and action recognition [168]. He further explains that the activation of motor neuron system for mere action recognition is not unexpected, even though it may sound odd. Understanding only the visual aspects without the underlying motor concepts will not provide knowledge about what it means to execute the observed action and the links to other similar actions. Inspired by these studies in neuroscience we asked ourselves the question: "Can we use action recognition methods to aid observational learning in robotic systems?". In answering this we have developed O₂A , a one-shot observational learning method that utilizes a feature extractor pre-trained for action recognition.

Action recognition from videos has seen a surge in performance with the advent of DNN architectures [169, 170]. These DNNs extract powerful perceptual

features from the action videos that abstractly represent the action and are generally discriminative (in terms of actions) and agnostic (to domain settings such as viewpoint of observation, object properties, scene background and morphology of manipulators [17]). In the presented method we use such an action recognition network pre-trained on a generic action dataset to extract an abstract perceptual representation of tasks (the ‘*action vector*’) from the demonstration videos. The necessary task-discriminative and domain-agnostic properties come from pre-training for action recognition on a range of diverse actions and domain settings which share common underlying visual dynamics. Using these action vectors, a reward is generated that directly reflects the similarity of the actions performed by the demonstrator and by the robot. This reward is then used for reinforcement learning of an optimal robotic control policy for carrying out the demonstrated task. The overall working of O₂A is further detailed with diagrammatic illustrations (Figure 4.1) in the next chapter (Chapter 4). The concept of the action vector and its extraction is explained in section 3.3. The evaluation experiments with results are given in section 3.4. Finally section 3.5 summarises the chapter.

The key innovations of the presented method lie in achieving one-shot learning with: (a) an action vector extractor pre-trained for action recognition and (b) using a generic action dataset for pre-training.

3.2 Pre-training with large generic datasets

Pre-training on large generic datasets has become common in the fields of computer vision and natural language processing. Models are first pre-trained on a large generic dataset(s) in a supervised or unsupervised manner. After pre-training, the models are used to solve downstream problems with minimum/no fine-tuning. Generic language models such as ELMo [171], GPT [172, 173, 174], BERT [175] have shown success in solving several downstream language processing problems. Similarly, ImageNet models [176], Image-GPT [177], BiT models [178], SEER [179] have demonstrated that this approach can be applied for computer vision problems as well. We introduce this concept into visual robotic manipulation.

3.3 Action vectors

An action vector is as an abstract task-discriminative and domain-agnostic perceptual representation of the task captured in a video. Task-discriminative and domain-agnostic means that the action vectors from the videos depicting different instances of the same task should be closer to each other irrespective of the domain in which they are recorded and further away to action vectors from other classes (as shown in Figure 3.1). Action vectors capture the underlying meaning of a task at a level of abstraction and is invariant to domain shifts. Ideally, there will be the right emphasis on both the end goal and the path followed during task execution. In our method the action vector extractor is pre-trained for action recognition. A generic action dataset depicting wide variety of actions under different domain settings is used for the pre-training. The action vector extraction is based on the following two assumptions:

(1) The spatio-temporal features generated by the final layers of an action vector extractor pre-trained for action recognition on a generic action dataset, are task-discriminative and domain-agnostic. The assumption is reasonable since the same layer outputs are used to recognise different actions, independently of camera angles of recordings, varying scene backgrounds, illumination conditions, actors / manipulators, object appearances, interactions, pose and scale.

(2) The action vector extractor pre-trained on a generic dataset can generalise to unseen manipulation tasks used in robotic observational learning. The intuition is that the underlying visual dynamics between the generic action dataset and the manipulation tasks are the same. For example, it is the same physical laws of dynamics governing object interactions, both for a cricket shot as well as a robot striking cubes.

Section 3.4 shows the experiments and results that validate these critical assumptions.

3.3.1 Dataset

We use the UCF101 action recognition dataset [16] as the generic action dataset for pre-training. The dataset was selected considering the diversity in the variety

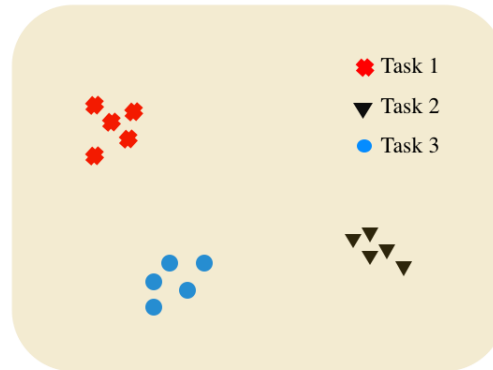


Figure 3.1: An illustration of the conceptual action vector space. Action vectors from videos of a task are closer to each other irrespective of the domain settings in which they are recorded and further away from other classes.

of actions and domain settings of video samples present. Sufficient diversity in terms of actions and domain settings is crucial for the action vector extractor to generalise to unseen manipulation tasks used for observational learning and to extract task-discriminative and domain-agnostic action vectors from them. Also in Table 4.6, we present a comparison of the commonly used action datasets in literature. UCF101 dataset was most suited for pre-training at the time, in terms of number action classes and video samples per class. However, recently more advanced datasets with fine-grained action classes are available which are given in Appendix 6.1 and could be used in future research.

The UCF101 action dataset consists of 13320 real world action videos from YouTube, having 101 action categories. The dataset has a large diversity (as shown in Figure 3.2) in terms of domain settings such as variations in camera motion, recording angle, object appearances, pose and scale, scene background and illumination conditions. The range of actions depicted also includes different kinds of interactions between various objects and a range of body articulations and shapes. The videos are collected at a fixed 25fps frame rate, 320x 240 resolution and have a mean clip length of 7.21 seconds. The training and validation sets are selected with a random split function. Further details of the dataset with the full list of 101 action categories are given in Appendix 6.2.

Table 3.1: Publicly available action datasets. UCF101 dataset was the most suited at the time for pre-training, in terms of number action classes and video samples per class.

	Release year	No: of action classes	Video samples per class	Resource
Hollywood2[180]	2009	12	61-278	Movies
UCF50[181]	2010	50	> 100	YouTube
HMDB51[182]	2011	51	> 101	Movies, YouTube, Web
ASLAN[183]	2012	432	~8	YouTube
UCF101[16]	2012	101	~132	YouTube

3.3.2 Network architecture

A 3D convolution [17, 184] based neural network (3D CNN) is used for action recognition in our research. This architecture was selected based on the ease of implementation and training. However, any DNN based architecture such as LSTM network [129], Two stream CNN [185], 2D CNN [186], Temporal segmentation network [187] or even a custom designed network can be used.

3D convolution and 3D pooling

3D convolutions are 2D convolutions extended to a third dimension, to extract temporal information from 3D data structures such as videos or volumetric CT scans as illustrated in Figure 3.3. In Figure 3.3.a, a 2D data structure with height H , width W is convolved with 2D kernel of size $k \times k$, where k is the kernel size. And in Figure 3.3.b, a 3D data structure with height H , width W and temporal length L is convolved with 3D kernel of size $d \times k \times k$, where k and d are the kernel spatial size and temporal depth respectively.

Similarly, 2D pooling is also extended to 3D pooling [17], where pooling is applied to input data within a 3D pooling cube as shown in Figure 3.4. Here a $d \times k \times k$ input data block is mapped (by pooling) into a single point in the output.



Figure 3.2: Selected few classes from UCF101 action dataset illustrating the diversity in terms actions and domain settings. Figure from [16]

Network architecture

The network architecture is modelled based on the original VGGNet [19] architecture, with convolution layers arranged in five blocks and with fully connected layers following them. The layer wise network architecture along with the kernel sizes, input and output dimensions are given in Table 3.2. Variable ‘NC’ denotes the number of classes. The network consist of eight 3D convolutional layers, five 3D maxpooling layers and three fully connected layers. The ReLU [188] activation function is used for all the convolutional and fully connected layers except the final layer, where a linear activation function followed by a Softmax function is used. A dropout of .5 is used for the fully connected layers during training to

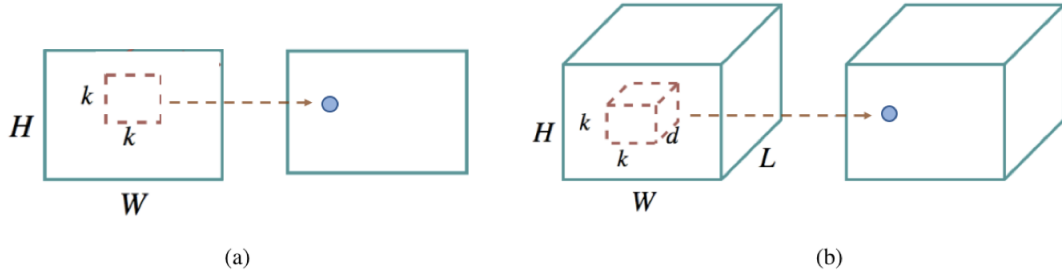


Figure 3.3: (a) 2D convolution (b) 3D convolution. Figure adapted from [17]

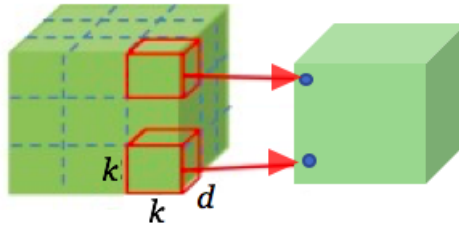


Figure 3.4: 3D pooling

avoid overfitting. We also use a zero-padding layer between the last convolution and pooling layers to control shrinkage of dimensions.

3.3.3 Pre-training objective

As mentioned, our pre-training objective is action recognition. We use the cross-entropy loss [189] function, $L(y, \bar{y})$ as given below:

$$L(y, \bar{y}) = - \sum_{i=0}^{NC-1} y_i \log(\bar{y}_i) \quad (3.1)$$

where, \bar{y}_i is the i^{th} value in the predicted output and y_i is the corresponding one-hot encoded ground truth value, both of which are expressed as probabilities. NC is the number of classes and for UCF101 dataset $NC=101$.

Table 3.2: Network architecture for the action vector extractor

Layer	Type	Kernel size	Input size	Output size
conv1	Conv3D	(3, 3, 3)	(16, 112, 112, 3)	(16, 112, 112, 64)
pool1	MaxPooling3D	(1, 2, 2)	(16, 112, 112, 64)	(16, 56, 56, 64)
conv2	Conv3D	(3, 3, 3)	(16, 56, 56, 64)	(16, 56, 56, 128)
pool2	MaxPooling3D	(2, 2, 2)	(16, 56, 56, 128)	(8, 28, 28, 128)
conv3a	Conv3D	(3, 3, 3)	(8, 28, 28, 128)	(8, 28, 28, 256)
conv3b	Conv3D	(3, 3, 3)	(8, 28, 28, 256)	(8, 28, 28, 256)
pool3	MaxPooling3D	(2, 2, 2)	(8, 28, 28, 256)	(4, 14, 14, 256)
conv4a	Conv3D	(3, 3, 3)	(4, 14, 14, 256)	(4, 14, 14, 512)
conv4b	Conv3D	(3, 3, 3)	(4, 14, 14, 512)	(4, 14, 14, 512)
pool4	MaxPooling3D	(2, 2, 2)	(4, 14, 14, 512)	(2, 7, 7, 512)
conv5a	Conv3D	(3, 3, 3)	(2, 7, 7, 512)	(2, 7, 7, 512)
conv5b	Conv3D	(3, 3, 3)	(2, 7, 7, 512)	(2, 7, 7, 512)
zeropad5	ZeroPadding3D	(0, 1, 1)	(2, 7, 7, 512)	(2, 8, 8, 512)
pool5	MaxPooling3D	(2, 2, 2)	(2, 8, 8, 512)	(1, 4, 4, 512)
flatten1	Flatten	-	(1, 4, 4, 512)	(8192)
fc6	Dense	(4096)	(8192)	(4096)
fc7	Dense	(4096)	(4096)	(4096)
fc8	Dense	(NC)	(4096)	(NC)

3.3.4 Pre-training and action vector extraction

For pre-training, we first uniformly downsample the UCF101 videos in time into $N_f=16$ frames as illustrated in Figure 3.5 for providing a fixed-length representation for each video clip. We also resize video frames (with bilinear interpolation) into 112×112 pixels to standardize the size. These downsampled and resized videos are then used for training the network for action recognition. The action vector extraction network trained with the UCF101 dataset is referred to as ‘NN:UCF101’ hereafter. The training details are given in Table 3.3 and testing accuracies (on UCF101 test set) during training are plotted in Figure 3.6. We apply the same pre-processing steps of down-sampling and resizing to videos of demonstrations and robot trial executions for action vector extraction during observational learning.

However, obtaining high testing accuracies on source datasets alone cannot guarantee better performance on target datasets/downstream tasks in transfer learning [190]. When to stop training on the source dataset to obtain the most useful transfer features is still an open area of research. We further evaluated transfer performances of models with varying testing accuracies (on source UCF101 dataset) as reported in Section 3.4.6. The results show that the transfer performance has indeed plateaued.

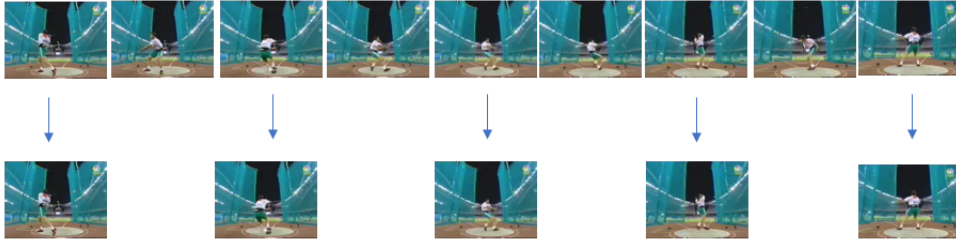


Figure 3.5: Downsampling training video before feeding into the network. Downsampled size (N_f) set to 5 frames for ease of illustration.

After training, we use features from one of the final layers of NN:UCF101 as the action vector. Our experiment (reported in Section 3.4) shows that the features from layers pool5 (size: 8192) and fc6 (size: 4096) are best suited to be used as the action vector. We report results, both when the features from pool5 and fc6 layers are used as the action vector in this thesis.

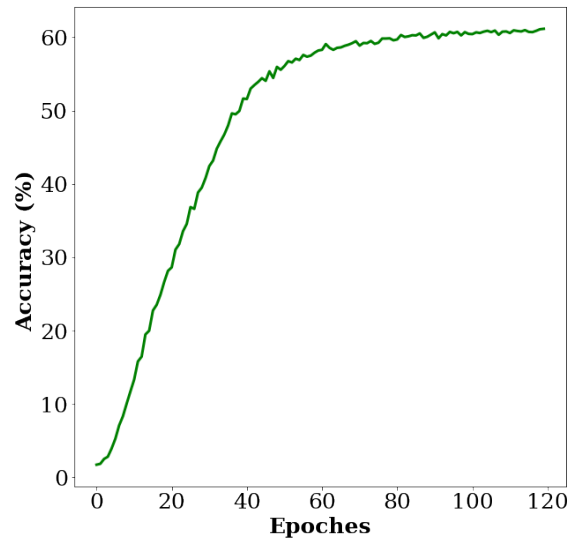


Figure 3.6: Testing accuracy (on UCF101 test set) per epoch during action recognition pre-training with UCF101 dataset

Table 3.3: Details of action recognition pre-training on the UCF101 dataset

	NN:UCF101
Number of classes (NC)	101
Batch size	16
Input size	(16, 16, 112, 112, 3)
Output size	(16, 101)
GPUs used	2 x Nvidia K80
Training time	48 hrs
Optimizer	ADAGRAD [191]
Learning rate	0.001
Number of training examples	9,990
Number of testing examples	3,330
Total number of trainable parameters	78,409,573
Number of epochs	119
Best testing accuracy	60.72%

3.4 Analysing action vectors

In this section, we aim to validate the assumptions for our action vector extraction method explained in Section 3.3. First we collect a manipulation tasks dataset, the LMD (detailed in Section 3.4.1). Note that LMD is only used for evaluation and not used in any way during pre-training of the network for action recognition. Using this dataset we conduct the clustering analysis to identify which one of the final layers of NN:UCF101 provides the best action vector for manipulation tasks. We also calculate the class similarity scores and do visualisation for the action vectors from LMD. In the next sections we detail the experiments and discuss the results obtained.

3.4.1 LMD evaluation Dataset

LMD consists of videos of three different manipulation tasks: reach, push and reach-push, examples of which are shown in Figure 3.7. The task videos are collected directly with a human hand and by using tools resembling robotic manipulators/end effectors. Each class consists of 17 videos with variations in viewpoint of observation, object properties, scene background and morphology of manipulator as illustrated in Figure 3.8. Note that very similarly looking task classes were carefully selected and the same set of objects and manipulators were used across tasks for collecting the videos. These choices are deliberate to make the task differentiation more challenging. Under these circumstances, only an efficient action vector extractor can produce task-discriminative and domain-agnostic action vectors for different task classes in LMD. The details of the dataset are summarised in Table 3.4. Note that audio/sound is not recorded during data collection. This could be a future extension for multi-modal robotic manipulation learning as described in chapter 5.3.1.

3.4.2 Clustering analysis

We conduct the clustering analysis to evaluate which one of the final layers of NN:UCF101 network provides the best action vector for manipulation tasks. We use the quality of the clusters in the action vector space as a measure to under-

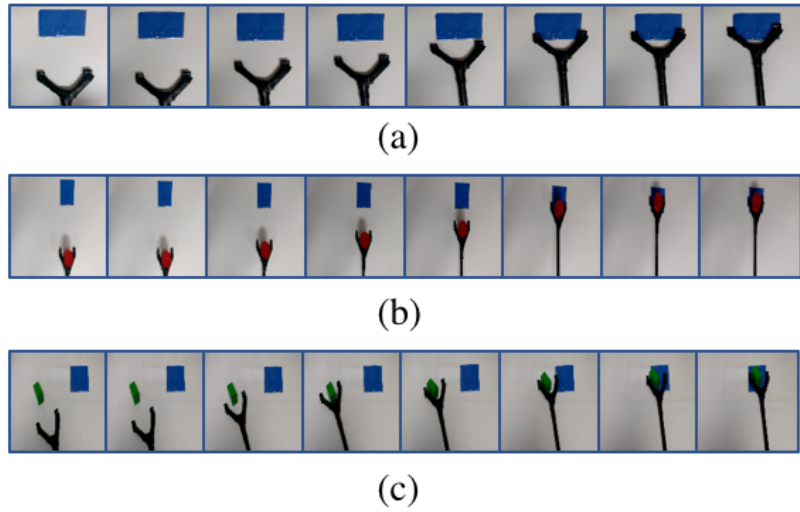


Figure 3.7: Tasks of (a) reaching (b) pushing and (c) reach-push from LMD

Table 3.4: Leeds Action Dataset details

Dataset property	Value
No: of actions	3
Total no: of videos	51
Videos per class	17
Total no: of frames	10,246
Min no: of frames per video	80
Max no: of frames per video	410
Avg no: of frames per video	201
Avg no: of frames per video in class reach	165
Avg no: of frames per video in class push	195
Avg no: of frames per video in reach-push	242

stand how task-discriminative and domain-agnostic are the action vectors, when features from different layers of NN:UCF101 are used. The more the action vectors are task-discriminative and domain-agnostic, the better the clustering of the action vectors from the same class will be. To analyse the quality of the clusters we propose to use a standard clustering algorithm and evaluation metric. The hypothesis behind this is that, the quality of the clusters will be directly correlated

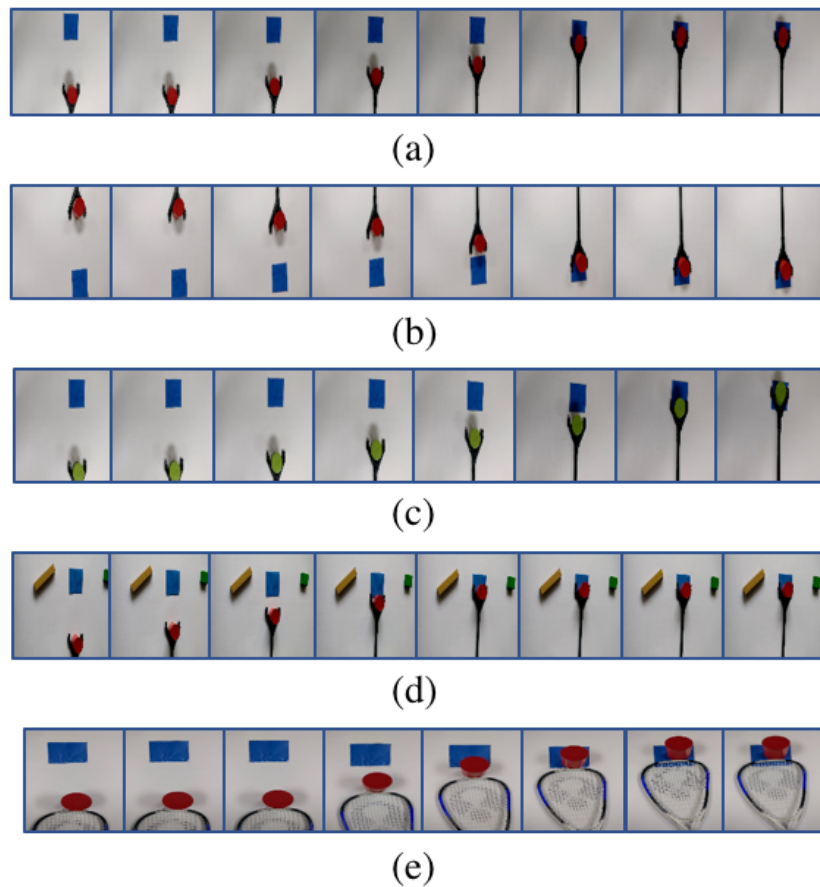


Figure 3.8: Diversity in the data collected for the task of pushing. (a) Normal pushing and pushing with changes in: (b) viewpoint of observation (c) object properties (d) scene background and (e) morphology of the manipulator. Similar diversity can be observed in other classes as well.

to the performance of the clustering algorithm, if all the other factors remain the same.

First, we extract the features from the pool5, fc6, fc7 and fc8 layers of the NN:UCF101 network, for all the 17 videos in LMD. The Baseline-R is obtained using features from the pool5 layer of the same NN:UCF101 network but initialised with random weights. The features extracted from each layer are then clustered with a supervised clustering algorithm. In our experiment we use K-means [192] with the value of $K=3$, corresponding to the number of task classes.

After applying the clustering algorithm, the predicted labels are evaluated against the ground truth labels. We use the standard metric of Adjusted Random Index (ARI) [193, 194] for evaluation. The ARI score gives the extent to which the predicted clustering corresponds to the ground truth clusters by counting pairs that are assigned to the same and different clusters. The ARI score will indicate the quality of clustering and thereby give a measure of task discrimination and domain invariance of the action vectors from different layers of NN:UCF101. In our experiment we use the implementation of K-means clustering algorithm available from [195] with the default settings as given in Table 3.5.

Table 3.5: K-means algorithm parameters used

Parameter	value
No: of clusters	3
Method of initialisation	k-means++ [196]
Algorithm	Elkan method [197]
No: of runs	10
Max: number of iterations per run	300
Relative tolerance	$1e^{-4}$

Adjusted random index (ARI)

The Adjusted Random Index or ARI metric is used for evaluating the performance of the clustering algorithms. Even though other metrics such as V-Measure [198] can be used, we have selected ARI due to the simplicity in implementation and also it is normalised against chance. ARI values are bounded by $[-1, 1]$, where -1 is the lowest score, 0 indicates random clustering and 1 shows that the predicted clustering corresponds to the ground truth perfectly. Normalisation against chance ensures a value close to 0 for random labeling, independently of the number of clusters and samples and exactly 1 when the clustering are identical.

ARI can be calculated as follows: Let x be the number of pairs of items in the same clusters in ground truth clustering and predicted clustering and y be the number of pairs of items in different clusters in ground truth clustering and

predicted clustering. Then the Random Index (raw RI) is given by Equation 3.2.

$$RI = \frac{(x + y)}{C_2^{total}} \quad (3.2)$$

where, C_2^{total} is the possible number of pairs in the dataset without ordering. However, raw RI is not normalised for chance, which means that a random labelling may not always return a score of 0. This drawback is overcome by defining ARI as given in Equation 3.3.

$$ARI = \frac{RI - E(RI)}{Max(RI) - E(RI)} \quad (3.3)$$

where, $E(RI)$ is the expected RI for random labelling and $Max(RI)$ is the maximum RI value, which is 1.

Results

Table 3.6: ARI scores (higher the better). Results show that the features from layers pool5 and fc6 of the NN:UCF101 network are best suited to be used as action vectors.

Layer	Vector size	ARI score
Baseline-R	8192	0.07
pool5	8192	0.26
fc6	4096	0.34
fc7	4096	0.19
fc8	101	0.14

The results of the experiment are tabulated in Table. 3.6. The ARI value for Baseline-R is close to zero as expected and gives us the baseline to compare with. The ARI score increases when features from pool5 to fc6 layers are used as the action vector, but drops for the final fc7 and fc8 layers. The results indicate that the features from pool5 or fc6 layer of the NN:UCF101 network are the most suitable

to be used as the action vector. These results are in agreement with the previous works [199, 200] that study transferability of features from different layers of a pre-trained CNN to new downstream problems. Specifically, Azizpour et.al [199] have shown that the first fully connected layer after the convolutional layers of a pre-trained (for classification) network produced the most generic features for a range of 15 downstream problems.

Furthermore, we also performed clustering analysis when features from different layers are concatenated and used as the action vector. The results are tabulated in Table 3.7. Concatenating features did not produce any improvement in the performance. Hence we use features from each layer (pool5 and fc6) of the NN:UCF101 network separately as the action vector in O_2A . However, it would be interesting to study how to exploit the consistency in performance while concatenating layer outputs, to reduce the dependency on individual layers. Additionally, we also conducted the clustering analysis when only a set of features (unsupervised selection) are used. Experiment details and the results are given in Appendix 6.3.

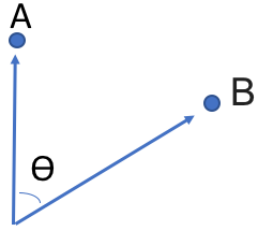
Table 3.7: ARI scores when features from different layers are concatenated and used as the action vector. Results do not show any significant improvement in the performance.

Layer	Vector size	ARI score
pool5+fc6	12288	0.26
pool5+fc7	12288	0.26
pool5+fc8	8293	0.26
fc6+fc7	8192	0.29
fc6+fc8	4197	0.27
fc7+fc8	4197	0.20

3.4.3 Class similarity scores

In this experiment we calculate the interclass and intraclass similarity scores for different classes of LMD in the action vector space. The similarity score between

a pair of action vectors is shown as the cosine of the angle between them as given in Equation 3.4.


$$\text{Similarity} = \text{Cos } \theta = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \times \|\mathbf{B}\|} \quad (3.4)$$

where, \mathbf{A} and \mathbf{B} are two vectors, $\mathbf{A} \cdot \mathbf{B}$ is the dot product between them and $\|\mathbf{A}\|$, $\|\mathbf{B}\|$ are the vector magnitudes. The similarity scores are bounded by $[-1, 1]$ with -1 indicating diametrically opposite vectors and 1 indicating coinciding vectors.

The results are tabulated in Table 3.8. For each chosen feature layer, the diagonal values represent the average of similarity scores between pairs of action vectors from the same class and the non-diagonal values are the average of similarity scores between pairs of action vectors from different classes. The diagonal values are greater than the rest of the values indicating task discrimination and domain invariance for the action vectors extracted. Through experiments reported in Chapter 4, we show that this is adequate for one-shot observational learning. The only exception is for layer fc6 where a greater inter-class similarity score is observed between reach and push classes than the intraclass similarity score for reach class. Provided that both tasks are extremely similar, these results are promising.

Table 3.8: Class similarity scores. The intraclass similarity (diagonal values) are greater than the rest of the values, indicating adequate task-discrimination and domain-invariance.

Random weights	Reach	Push	Reach-Push
Reach	0.9873	0.9870	0.9870
Push	0.9870	0.9874	0.9868
Reach-Push	0.9870	0.9868	0.9889

NN:UCF101(pool5)	Reach	Push	Reach-Push
Reach	0.7391	0.7371	0.6897
Push	0.7371	0.7547	0.6852
Reach-Push	0.6897	0.6852	0.7578

NN:UCF101(fc6)	Reach	Push	Reach-Push
Reach	0.4994	0.5001	0.4052
Push	0.5001	0.5352	0.4022
Reach-Push	0.4052	0.4022	0.4978

3.4.4 Visualisation

We also visualize the action vectors from LMD, projected into 2D using PCA [201], which are shown in Figures 3.9, 3.10a and 3.10b . The clustering of action vectors from the same classes, when compared to the Baseline-R is evident. This further indicates the domain-agnostic and task-discriminative nature of our action vectors. It must be noted that this visualisation collapses the vectors, of much greater dimensions (8192 for Baseline-R/pool5 and 4096 for fc6), into a 2D space, which might be causing some ‘artificial’ overlaps. Additionally, we also experimented with 3D visualisation of LMD, results of which are shown in Appendix 6.4.

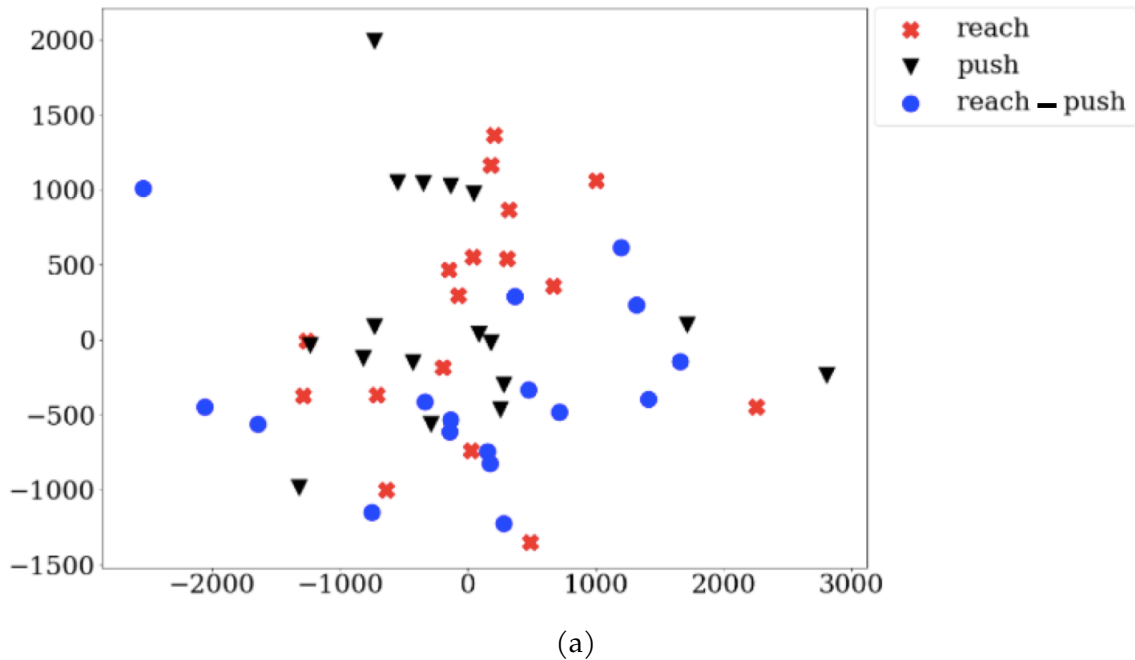


Figure 3.9: Visualising LMD using action vectors for Baseline-R (features from pool5 layer of NN:UCF101 with randomly initialised weights are used)

3.4. Analysing action vectors

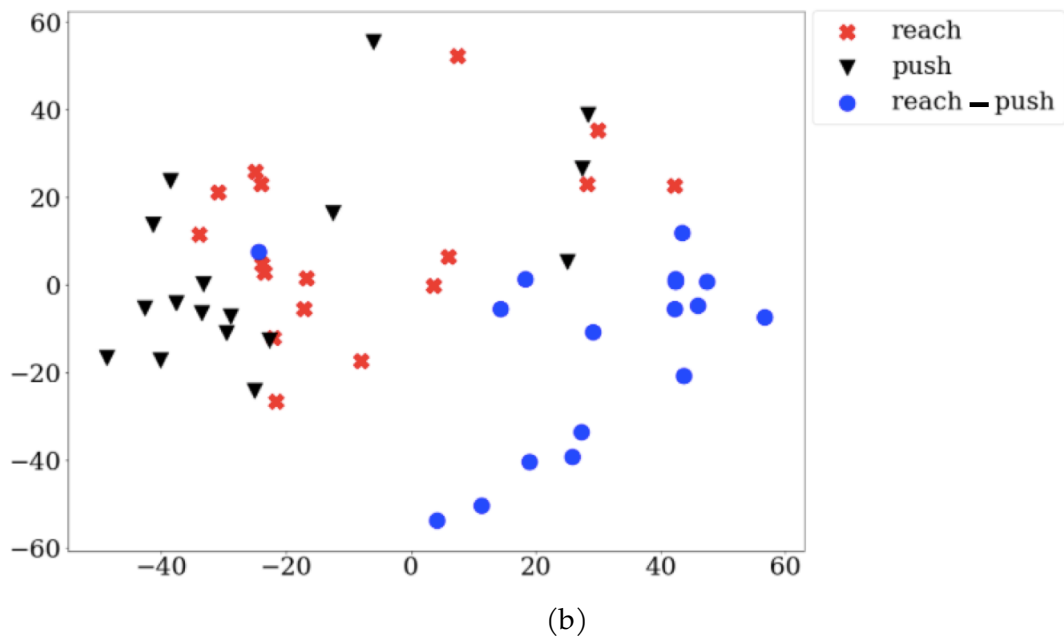
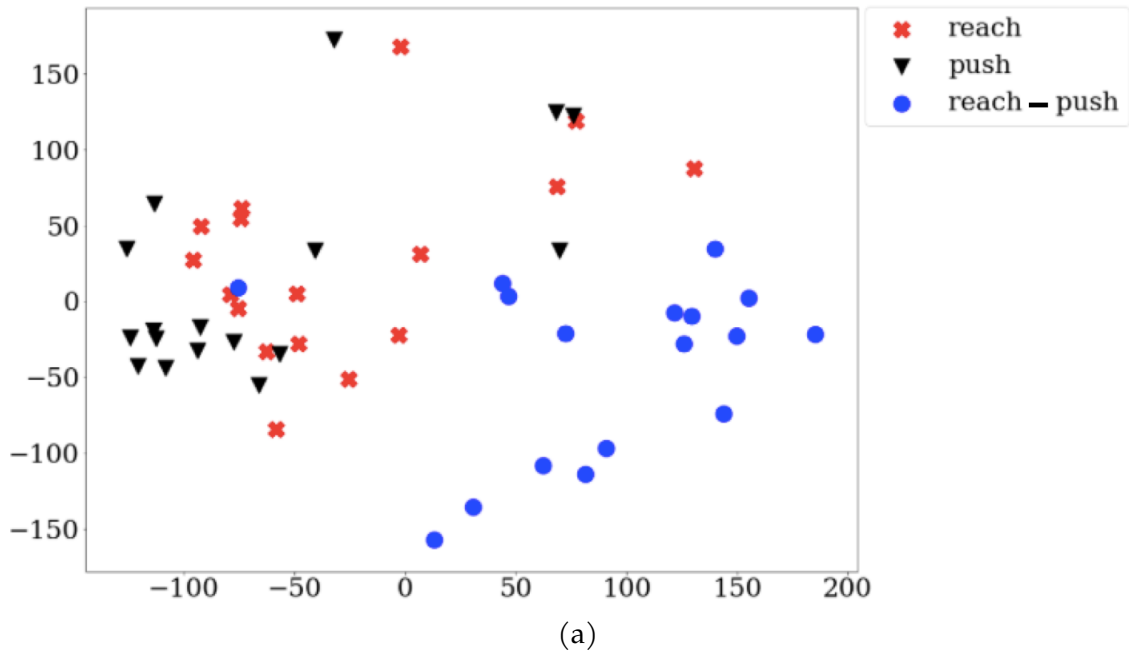


Figure 3.10: Visualising LMD using action vectors from (a) pool5 and (b) fc6 layers of NN:UCF101

3.4. Analysing action vectors

And to further understand the clustering of LMD, we visualised the dataset by merging the reach and push classes into a single class. Considering that both the classes have very similar looking motion dynamics, this merging is justified. The results are shown in 3.11, 3.12a and 3.12b. A clear class based clustering is now evident in the visualisations.

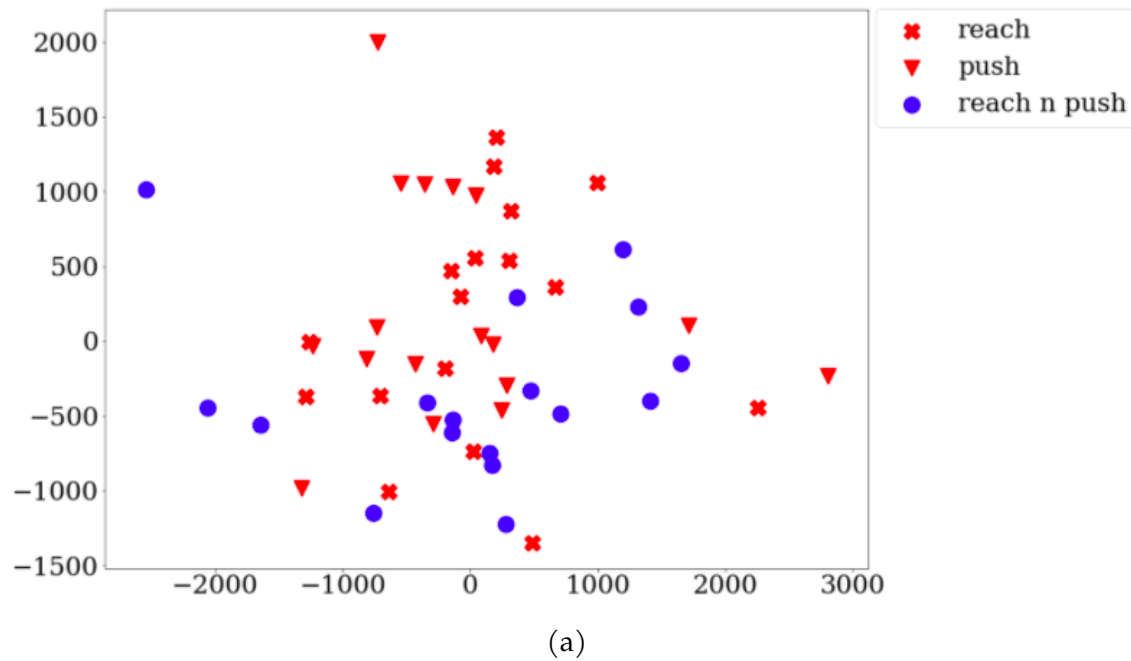


Figure 3.11: Visualising LMD (after merging reach and push classes) using action vectors for Baseline-R (features from pool5 layer of NN:UCF101 with randomly initialised weights are used).

3.4. Analysing action vectors

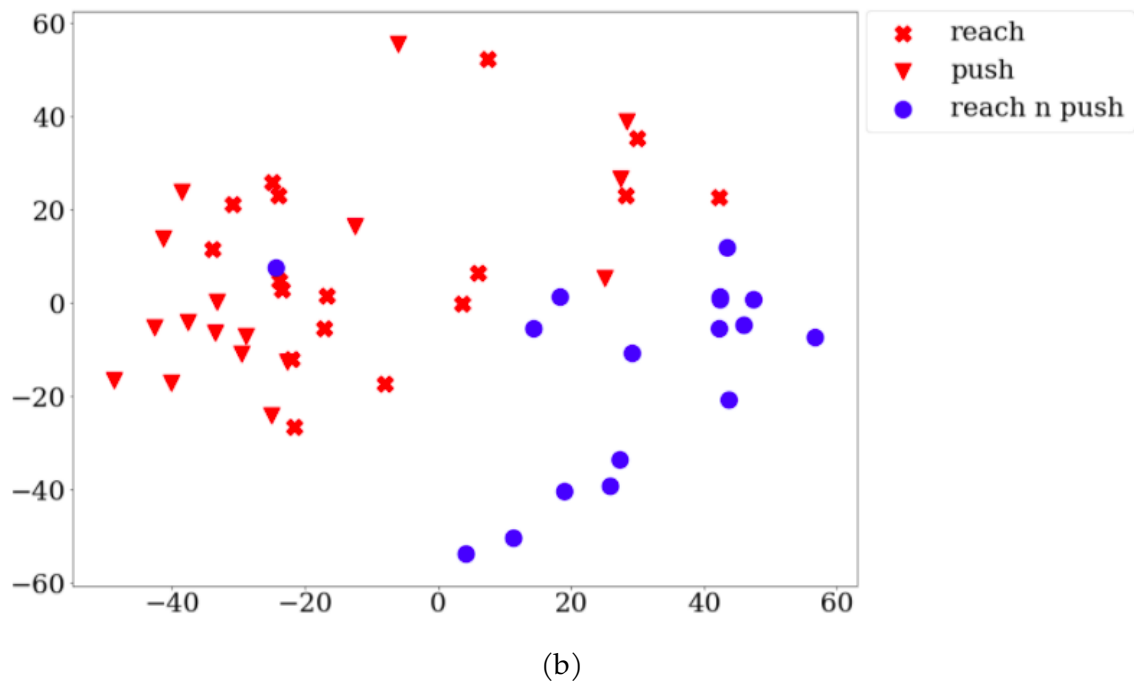
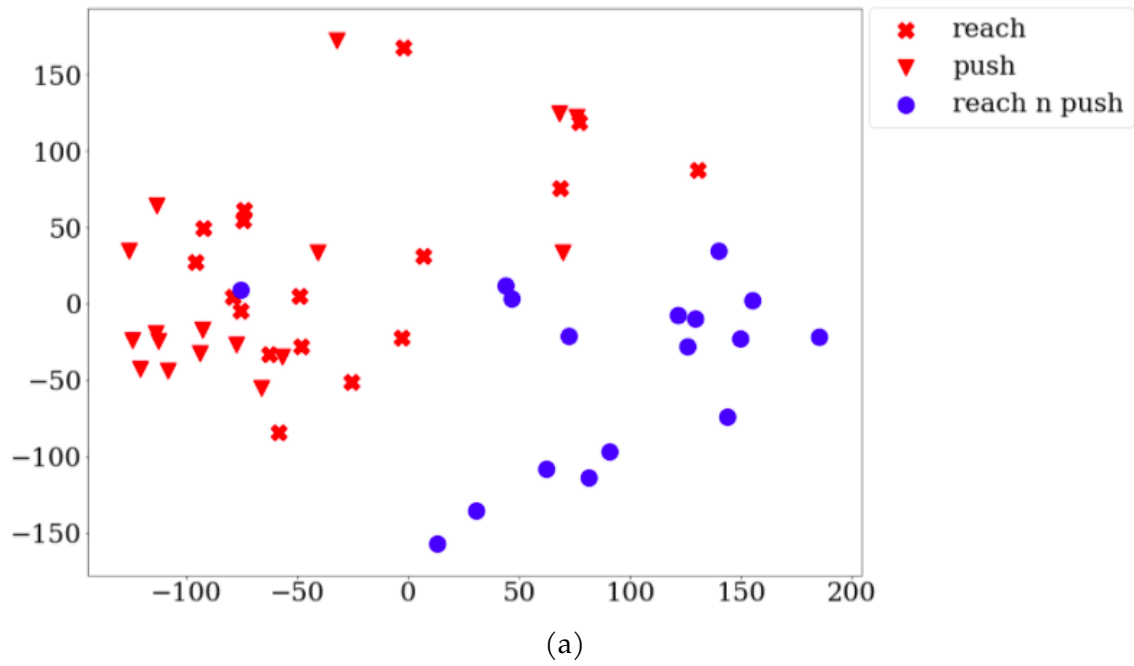


Figure 3.12: Visualising LMD (after merging reach and push classes) using action vectors from (a) pool5 and (b) fc6 layers of NN:UCF101

3.4.5 Discussion

Here we present a critical analysis of the experiments conducted and the results obtained for action vector analysis.

The effectiveness of K-means clustering algorithm tends to reduce as the dimensionality of the features increases, which is commonly referred as the 'curse of dimensionality'. In the higher dimensions, the ratio of distances to the nearest and furthest points approaches to one for the euclidean distance metric used in the K-means clustering [202] algorithm. Hence we further performed PCA dimensionality reduction on features from different layers of NN:UCF101, before being used as the action vectors. The results (reported in the Appendix 6.5), are in agreement with that of when using K-means clustering without dimensionality reduction, showing that the features from pool5 and fc6 layers are best suited to be used as the action vectors.

The intra-class similarity scores are comparatively lower for the tasks of reaching and pushing. However it has to be noted that the tasks of reaching and pushing require similar manipulator movements/trajectories. The intra-class similarity scores are higher for the the third class of reach-push which has different manipulator movements/trajectories from other classes. Similarly, while visualizing the action vectors we rely on PCA to compress action vectors from a very high dimension into 2D. This could possibly create artificial overlaps and cause difficulty in understanding how well the clusters are formed. However, taken all together these experiments provide enough evidence to be confident that the action vectors generated from the pre-trained extractor have task-discriminative and domain-agnostic properties when applied to unseen manipulation tasks.

3.4.6 Pre-training accuracy and transfer performance

Here we study the transfer performance of the action vector extractor (NN:UCF101) when trained further for higher testing accuracies on the source (UCF101) dataset. We evaluate the transfer performance using clustering analysis on LMD. The results are tabulated in Table 3.9 and plotted in Figure 3.13. The ARI scores for features from pool5 layer increase to .34 from .26, linearly with increase in testing accuracies. However, ARI scores for features from fc6 layer show no significant

3.4. Analysing action vectors

improvements and varies non-linearly with testing accuracies. The results show that transfer performances have plateaued for the given pair of source (UCF101) dataset and the downstream task of generating action vectors from LMD.

Table 3.9: Transfer performance evaluation when action vector extractor (NN:UCF101) is pre-trained for higher action recognition testing accuracies.

Training details					
Training time	48 hrs (Base model)	60 hrs	72 hrs	84 hrs	96 hrs
Testing accuracy (UCF101 test set)	60.72%	70.23%	77.05%	79.52%	80.62%
ARI scores (transfer performance)					
pool5	.26	.33	.33	.34	.34
fc6	.34	.23	.33	.36	.35

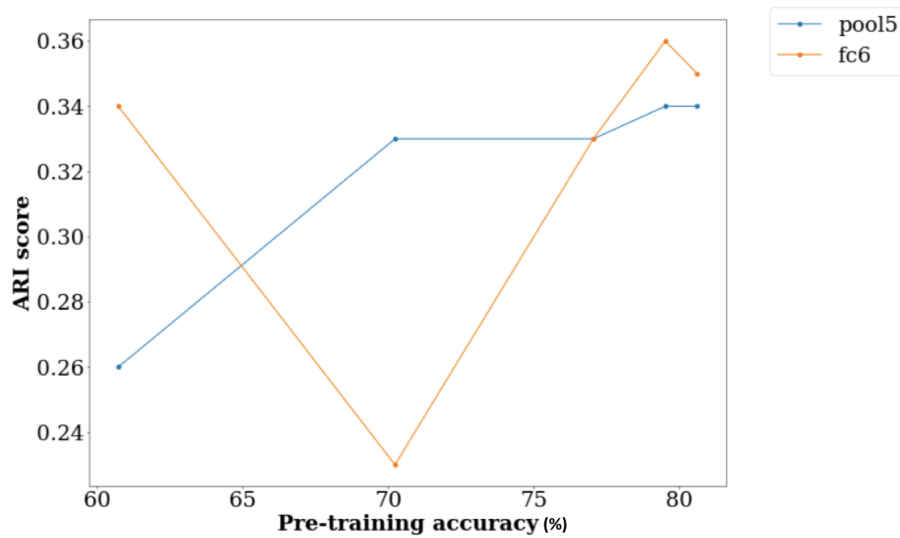


Figure 3.13: ARI scores when NN:UCF101 network is pre-trained for higher action recognition accuracies on UCF101 dataset.

3.5 Conclusion

In this chapter we discussed in detail, the concept of action vectors and our proposed method of action vector extraction from videos. The action vector extractor is pre-trained on a large generic action dataset for action recognition. We hypothesised that these extractors can generalize to unseen manipulation tasks that we wish the robot to learn with observational learning. We also reported the results of experiments conducted to study these assumptions. The results showed that the action vectors generated by pre-trained extractors have a certain degree of task discrimination and domain invariance when evaluated with a challenging LMD dataset we collected, containing manipulation tasks unseen during pre-training.

However, a critical question remains: Are the action vectors sufficient to enable one-shot observational learning of manipulation tasks in robotic systems? In the next chapter, we aim to answer this question with a range of experiments, both in simulation and with a real robot.

**

One-shot observational learning

4.1 Introduction

In the previous chapter we introduced and explained the concept of action vectors. We proposed a novel method for extracting action vectors from unseen manipulation tasks, using an action vector extractor pre-trained for action recognition on a generic action dataset. Through the experiments, we showed that these action vectors can be task-discriminative and domain-agnostic. Now the question that remains to be answered is: Can these action vectors be used for one-shot observational learning of robotic manipulation tasks? In this chapter we present O_2A in detail, a method for one-shot observational learning with action vectors. We also report on extensive robotic experiments and their results conducted to evaluate the performance of O_2A .

4.2 O_2A overview

A diagrammatic overview of O_2A is given in Figure 4.1. The method takes as input a video clip (observed in third-person view point) of someone demonstrating a task for a single time. The objective is to replicate the task with a robotic manipulator. The robot views the scene from an egocentric camera mounted in a fixed position. The demonstration video clip is then transformed into an action vector. In a similar fashion, an action vector is obtained from each of the trial executions by the robotic manipulator. The rewards are then calculated as the negative of

the Euclidean distance between the action vectors computed from the video clips of the demonstration and trial executions. Then we use a reinforcement learning (RL) algorithm to acquire an optimal control policy for performing the demonstrated task using this reward.

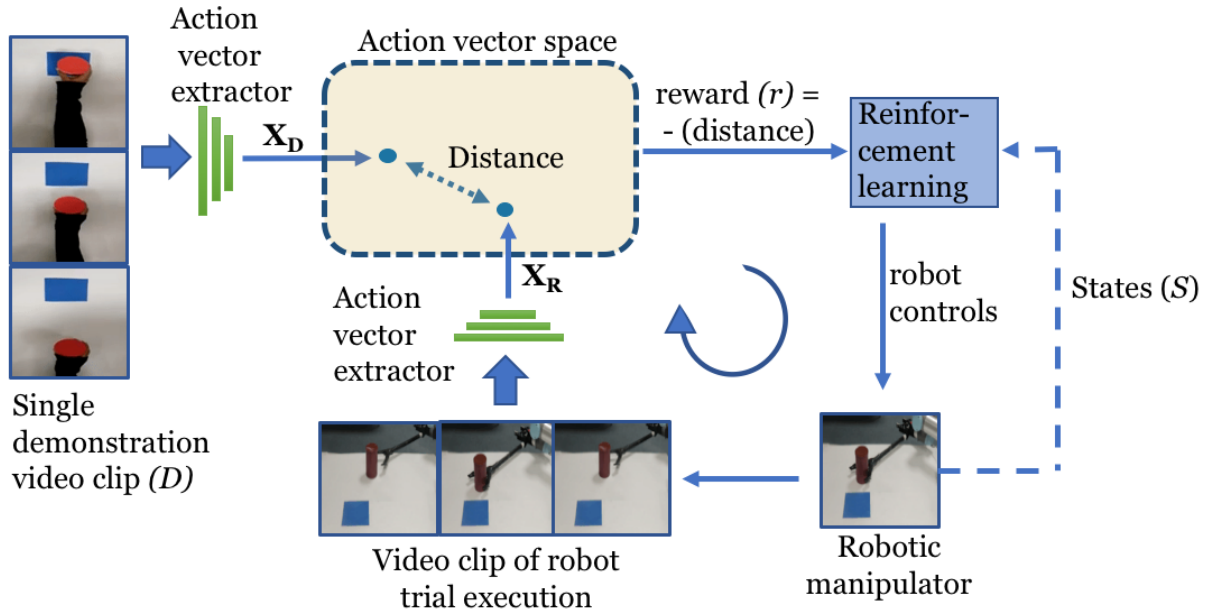


Figure 4.1: Overview of O₂A method. A 3D-CNN action vector extractor is used to extract action vectors \mathbf{X}_D and \mathbf{X}_R from the video clips of the demonstration and robot trial execution respectively. A reward function is used to compare \mathbf{X}_D and \mathbf{X}_R in the action vector space, generating a reward signal (r) based on their closeness. The RL algorithm then iteratively learns an optimal control policy by maximizing this reward signal, thus enabling observational learning.

With reference to Fig 4.1, let D be the single demonstration video clip of a task to be learned. We extract the n -dimensional action vectors \mathbf{X}_D and \mathbf{X}_R from the demonstration video D and the video clip of a robot trial execution respectively. The reward (r) for the RL algorithm is then calculated as the negative of the euclidean distance between action vectors \mathbf{X}_D and \mathbf{X}_R as given below:

$$r = -\|\mathbf{X}_D - \mathbf{X}_R\|_2 \quad (4.1)$$

Thus the reward directly measures the closeness of the demonstrated task and of the robot trial executions at an abstract level (in the action vector space). The RL algorithm will then maximize this reward function to learn an optimal control policy. This optimal control policy will control the robotic manipulator to generate an action like the one shown in the demonstration video. Thereby, O₂A enables one-shot observational learning of the demonstrated task.

In the upcoming sections, we present the robotic experiments and results evaluating the performance of the O₂A system. We conducted experiments both in simulation and with a real robot. The tasks used are reaching and pushing in simulation and pushing, hammering, sweeping and striking for the real robot experiment. The task definitions and completion measures are given in Table 4.1. Note that the task completion measures are only used for evaluating the performance of O₂A and are not used during task learning.

Table 4.1: Task definitions and completion measures

Task	Description	Task completion measure
Reaching (Simulation)	Reach a target zone	1-(final distance / initial distance between the center of the manipulator and the center of the target zone)
Pushing (Simulation & real robot)	Push an object into the target zone	1-(final distance/initial distance between the centers of the target zone and the pushed object)
Hammering (Real robot)	Hammer the target object	1-(minimum distance / initial distance between the hammer and the object during the execution)
Sweeping (Real robot)	Sweep crumpled cardboard pieces to the dustbin	The number of cardboard pieces in the dustbin after execution / total number of the cardboard pieces
Striking (Real robot)	Strike down a block of cubes	1-(minimum distance / initial distance between the blocks and manipulator during execution)

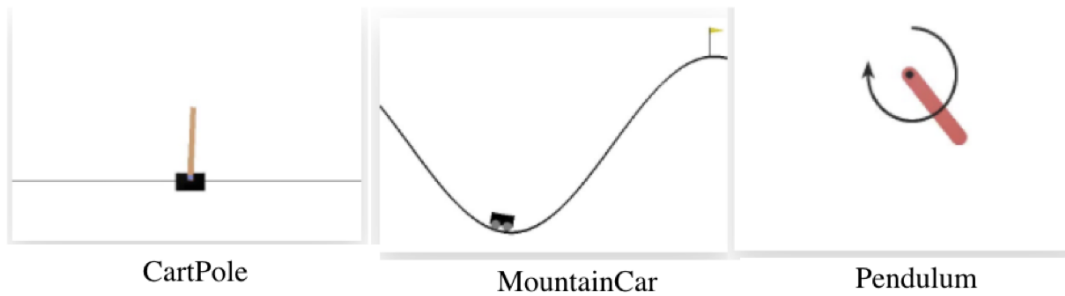


Figure 4.2: Sample environments available in OpenAI Gym

4.3 Simulation experiment

The objective here is to evaluate the performance of O_2A in a simulated environment under different domain shifts and compare its performance with baselines. The simulation experiment is run on a desktop workstation with an Intel core-i7 processor, 16GB RAM, 500GB hard disk and Linux Ubuntu16.04 operating system. The simulation environment is setup using the OpenAI Gym [203] framework. OpenAI Gym is a python framework for implementing and comparing RL algorithms. We present a comparison of OpenAI Gym with rllab, another popular RL framework, based on our selection criteria in Table 4.2. OpenAI Gym provides a variety of simulation environments for experimenting as illustrated in Figure 4.2 and also facilities integration of third party environments. For our experiment, we designed customised robotic manipulation setups using the MuJoCo (Multi Joint dynamics with Contact) physics engine [204]. We designed two environments, one for reaching and the other for pushing as shown in Figure 4.3. The environment consists of a 3DoF manipulator, target region and a 3D cylindrical object (for pushing). The XML designs are given in Appendix 6.6.

4.3.1 DDPG (Deep Deterministic Policy Gradient) RL algorithm

In the simulation experiment, we use the DDPG RL learning algorithm [205] for learning a control policy which will generate controls for the robotic manipulator. DDPG is a model-free and off-policy actor-critic algorithm. DDPG is derived by combining DQN (Deep Q network) method [206] with the DPG (Deterministic

Table 4.2: Comparing OpenAI gym and rllab RL frameworks. It shows our criteria for selecting the RL framework for the simulation experiment. ‘*’ symbol shows a better performance. OpenAI Gym is a clear choice for us satisfying all the requirements.

	OpenAI Gym	rllab
Designing custom environments	*	*
Integrating user defined reward functions	*	
Modular and flexible architecture	*	
Standard RL algorithm implementations	*	*
Documentation and tutorial availability	*	
Community support and active development	*	

policy gradients) [207] method. It learns both a Q function (using off policy data and the Bellman equation [208]) and a policy function concurrently. The advantages of using DDPG are:

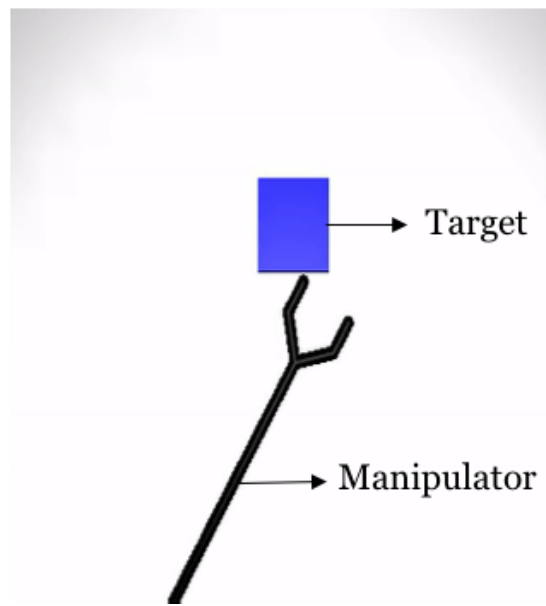
- It can handle high dimensional and continuous control spaces. This makes it suitable for directly controlling the robotic manipulator without discretizing robotic controls into pre-defined movements.
- Sample efficient when compared to other continuous control RL algorithms.

The components of the DDPG algorithm are explained briefly below. The complete algorithm is given in Appendix 6.7.

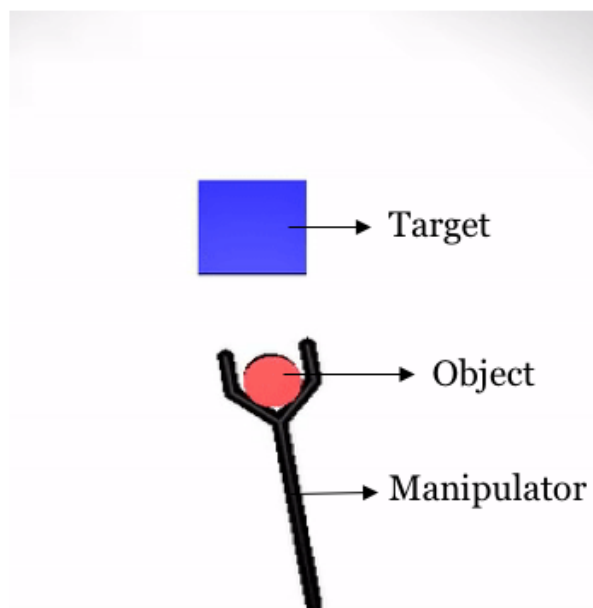
Critic network

The critic network refers to the Q value function $Q(S_t, A_t)$ used to calculate the Q values for each pair of state (S_t) and action (A_t) at time t . The critic network is updated by minimizing the following loss function:

$$L = \frac{1}{N} \sum_i (y_i - Q(S_i, A_i))^2 \quad (4.2)$$



(a)



(b)

Figure 4.3: Custom designed environments for the tasks of (a) reaching and (b) pushing in simulation experiments

4.3. Simulation experiment

where, N is the number of samples in the mini batch taken from the replay buffer used for training, i denotes i^{th} sample and y_i is the predicted Q value obtained with the following equation:

$$y_i = r_i + \gamma Q'(S_{i+1}, \pi'(S_{i+1})) \quad (4.3)$$

where, γ is the discount factor, r_i is the reward, Q' and π' are the target critic and actor networks respectively. The concept of target networks and replay buffer are explained in section 4.3.1. The architecture of the critic network used in our experiments is given in Figure 4.4. A two stream network is used to incorporate the actions and states. It consists of two fully connected layers and a output layer. The action values are included in the second layer of the network. Softplus [209] functions are used as the activation functions for the layers.

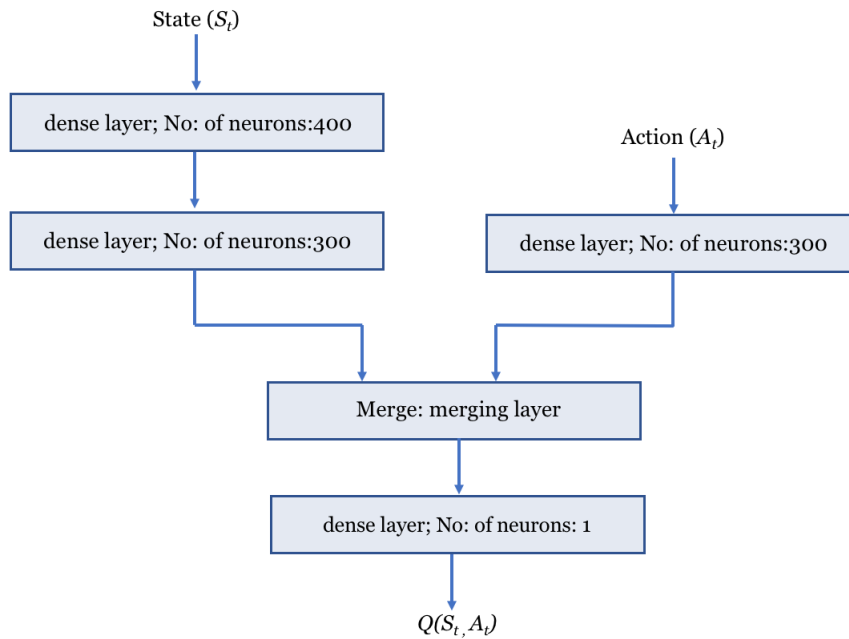


Figure 4.4: Critic network architecture

Actor network

The actor network refers to the policy function $A_t = \pi(S_t)$ that outputs the action at each time step t . DDPG implements a deterministic policy function which outputs the actions directly rather than a probability distribution over the possible set of actions. This makes it suitable for applying in continuous control problems. The action network is updated by the policy gradient with respect to actor parameters ($\nabla_{\theta\pi} J$) as given in Equation 4.4.

$$\nabla_{\theta\pi} J \approx \frac{1}{N} \sum_i \nabla_a Q(S_i, A_i) \nabla_{\theta\pi} \pi(S_i) \quad (4.4)$$

where, ∇ denotes the gradient function, N is the number of samples in the mini batch taken from the replay buffer used for training, i denotes i^{th} sample, Q is the value function represented by the critic network and π denotes the policy function represented by the actor network. The network architecture of the critic network in our experiments is shown in Figure 4.5. The network consists of two fully connected hidden layers followed by an output layer. We use Softplus activation functions for the hidden layers and tanh activation for the final output layer.

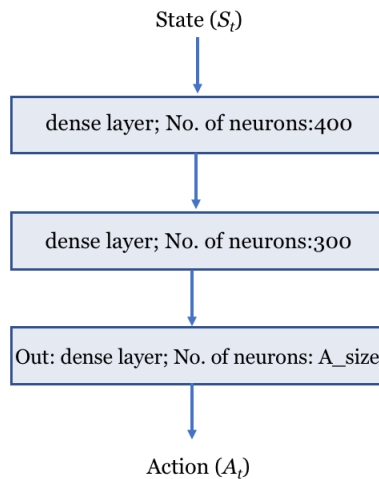


Figure 4.5: Actor network architecture

In our experiment we use continuous control action space. The actions gener-

ated by the actor network are the robotic controls with a size of three corresponding to each of the manipulator joints. The robotic controls could be torques, joint positions, angles or velocities of the manipulator. In the experiment we have used joint angles as the robotic controls.

Target Networks and Replay buffer

The DDPG algorithm uses target networks and replay buffers for stabilizing the training process. The target actor and critic networks are copies of the actor and critic networks respectively. Target networks are made to follow the parameters of the actual actor and critic network slowly with soft updates as given in equations below:

$$\theta^{Q'} = \tau\theta^Q + (1 - \tau)\theta^{Q'} \quad (4.5)$$

$$\theta^{\pi'} = \tau\theta^\pi + (1 - \tau)\theta^{\pi'} \quad (4.6)$$

where $\tau \ll 1$ and $\theta^Q, \theta^\pi, \theta^{Q'}, \theta^{\pi'}$ are parameters of the actor, critic, target actor and target critic networks. In our experiments the value of τ is set to .001.

The replay buffer stores the experiences obtained by exploration of the agent (in our case the robotic manipulator) during training. The experiences are stored as tuples (S_t, A_t, r_t, S_{t+1}) , where the values represent current state, action, reward obtained and next state at time t . Mini batches of these values are then randomly sampled from the replay buffer and used for training. This random sampling from the replay buffer ensures that the training data is independent of each other and makes optimization efficient. The size of the replay buffer used in our experiments is 10,000.

Exploration strategy

Exploration strategy enables RL algorithms to explore unknown areas in the state space during training. In continuous action state spaces the exploration is done

by adding noise directly to the actions or the policy parameters. In DDPG, noise is added to the actions using the ‘Ornstein-Uhlenbeck (OU) [210]’ process to enable exploration. So the actions taken by the agent (the robotic manipulator) during training are given by:

$$A_t = \pi(S_t) + \mathcal{N} \quad (4.7)$$

where, A_t and S_t are the action and state respectively at time t and \mathcal{N} is the OU noise added for exploration. It has to be noted that the noise is added only during training. During evaluation/testing the actions are obtained directly from the policy function (actor network) as in equation 4.8.

$$A_t = \pi(S_t) \quad (4.8)$$

States

States in the RL algorithm are the instantaneous observations of the environment. In our case, it is visual observations by the robotic system. In the experiment, we use VGGNet pre-trained on ImageNet [19] for converting raw RGB images into visual state features. The 4608 long feature vector obtained from the last convolutional layer of the VGG-16 network is used as the state representation.

4.3.2 Experimental setup

To explore the resilience of our method to shifts between the demonstrator and learner domains, we conducted the experiment with six different domain shifts, as defined in Table 4.3. The tasks of reaching and pushing are used in the simulation experiment. The task definitions and completion measures are given in Table 4.1. In each setup (characterising a domain shift), we collect a single demonstration in the real world and run the DDPG algorithm 10 times. Each run has 20 episodes per run and the number of steps per episode are 60 and 160 for reaching and pushing respectively. The hyperparameters used are given in Table 4.4.

For each run, DDPG returns a control policy that corresponds to the maximum reward obtained. After training, we pick the top-2 [211] control policies with the highest rewards, and the task completion measures are calculated. The top control policies were selected to avoid policies from poorly performing runs affecting the overall performance. The output of the control policy are the robotic controls with a size of three corresponding to each of the joints. We perform the experiment with action vectors extracted from both the pool5 and fc6 layers of the NN:UCF101 network. Figure 4.6 shows snapshots of the demonstration and execution of the corresponding learned policy for selected setups. Videos of the results including demonstrations are available in our [webpage](#).

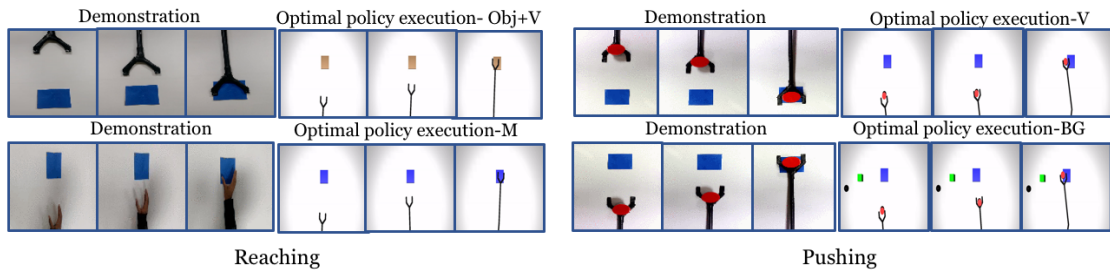


Figure 4.6: Snapshots of the demonstration and the execution of corresponding learned policies in the simulation experiment for selected domain shifts. (Results shown for action vectors extracted from pool5 layer of NN:UCF101 network).

4.3.3 Oracle and baselines

We compare our method with an oracle and two baseline approaches. The oracle is trained by using the corresponding task completion measure specified in Table 4.1 as the reward, in place of a reward derived from action vectors. It represents an upper bound on performance. The two baselines represent a video clip by averaging a ‘static’ representation for each frame, in contrast to the spatio-temporal representation used in O_2A . Rewards are then generated using these representations. In Baseline-1, features from the output of the last convolutional layer of the ImageNet [19] pre-trained VGG-16 network are used and in Baseline-2, HOG [212] features are used. The average of the task completion measures

Table 4.3: Domain shifts used in our experiment

	Domain shift
I	Observation viewpoint, object properties, morphology of the manipulator and scene background remain the same in the demonstration and learning domain
V	Observation viewpoint is different between the demonstration and the learning domain; other factors remain unchanged
Obj	Objects with different colour (for pushing, reaching and hammering tasks) or shape (hammering task) used in the learning domain
Obj+V	Both the viewpoint of observation and object properties vary between the demonstration and the learning domains
BG	Background clutter is introduced to the scene in learning domain, which was not present during the demonstration
M	Manipulators with different morphologies used in the demonstration and the learning domain. Demonstrations with a human hand (reaching and pushing tasks) and with a manipulator with a different morphology (hammering task) used.

Table 4.4: DDPG parameters settings

DDPG parameters	Value
Actor learning rate	0.0001
Critic learning rate	0.001
State size	4608
Action size	3
Optimiser	ADAM
Gamma (γ)	0.99
Tau (τ)	0.001
Mini batch size (N)	64
Replay buffer size	10,000

for the top two control policies for oracle, O₂A and the baseline approaches are plotted along with the standard deviations in Figures 4.7a and 4.7b. The learned policies from O₂A were successful in performing the demonstrated task under different domain shifts with good task completion measures. It also significantly outperforms both baseline approaches and has a comparable performance to the oracle.

4.3.4 Correlation of rewards

We further analysed the quality of the rewards generated in O₂A, the baseline approaches and the oracle. To compare, we calculate the Pearson correlation coefficient [213] between the episodic perceptual rewards (O₂A, baselines) and the oracle rewards for the top two runs. Pearson correlation coefficient (ρ) between two variables X and Y are given by:

$$\rho = \frac{\text{cov}(X, Y)}{\sqrt{\text{var}(X)}\sqrt{\text{var}(Y)}} \quad (4.9)$$

where $\text{cov}(X, Y)$ is the covariance of X and Y , $\text{var}(X)$ is the variance of X and $\text{var}(Y)$ is the variance of Y . A high positive correlation (typically > 0.5 [201]) indicates that the perceptual rewards are as good as the oracle rewards. All the results are tabulated in Table 4.5. From the results, the correlation coefficients are greater than 0.5 in all the cases for O₂A, indicating that our rewards are as accurate as the oracle rewards. Also, the correlation is higher and positive compared to the baselines for a range of domain shifts showing the superior performance of our method.

4.3. Simulation experiment

Table 4.5: Pearson correlation coefficients between the rewards from the oracle, and from O₂A and two baselines. The coefficients are highest in most of the cases and positive for O₂A rewards compared to baseline approaches.

Task 1: Reaching						
	I	V	Obj	Obj+V	BG	M
O ₂ A (NN:UCF101 (pool5))	.8567±.0079	.7807±.0531	.8209±.0157	.6448±.2146	.7736±.0007	.9605±.0048
O ₂ A (NN:UCF101 (fc6))	.8318±.0600	.7911±.0588	.8199±.0718	.8620±.0713	.8108±.1126	.8761±.0032
Baseline-1	.5872±.1744	.4069±.2361	.6112±.2612	.6099±.0901	.5289±.0189	.0487±.0448
Baseline-2	.7387±.0681	-.8106±.0086	.7115±.1272	-.8189±.0501	-.5738±.0337	.1256±.0629
Task 2: Pushing						
O ₂ A (NN:UCF101 (pool5))	.9345±.0034	.9413±.0362	.6943±.1419	.8650±.0847	.8552±.0677	.6594±.1834
O ₂ A (NN:UCF101 (fc6))	.8037±.1125	.8826±.0239	.6898±.1927	.8179±.0702	.9147±.0099	.8489±.0987
Baseline-1	.9372±.0270	.8908±.0615	.5817±.3124	.7488±.0631	.8978±.0704	.5797±.1141
Baseline-2	.0173±.4550	-.1346±.3410	.5900±.1625	-.4352±.1292	-.5386±.1243	.3700±.5195

4.3. Simulation experiment

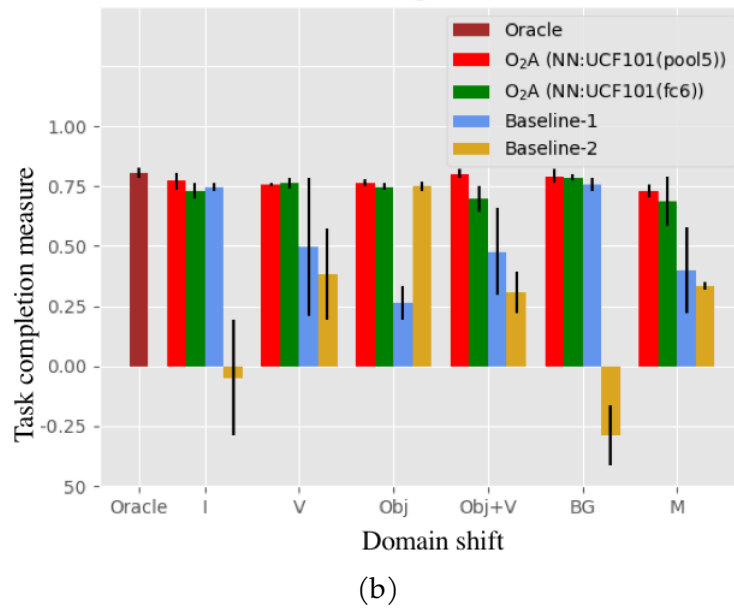
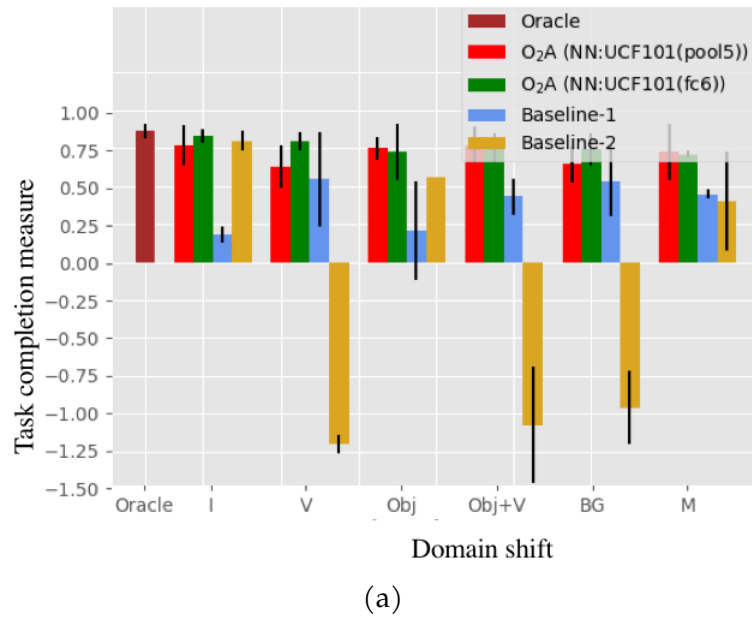


Figure 4.7: Task completion measures for the task of (a) reaching and (b) pushing in the simulation experiment. O₂A outperforms both the baselines and has performance comparable to the oracle under all domain shifts. The oracle score is shown only once since it is unaffected by the domain shifts (refer to Table 4.3 for domain shift definitions).

4.3.5 Trajectory maps

Here we plot the trajectories followed by the robotic manipulator (for reaching task) and pushed object (for pushing task) in each episode during reinforcement learning of the task. This visualisation will help to understand if high rewards are obtained for desired trajectories while learning the demonstrated task. The top-5 trajectories with the highest reward values obtained during task learning are coloured with red and the rest of the trajectories are in blue.

Additionally, we also show the results when the O_2A action vector extractor is pre-trained with a manipulation task dataset, the Multiple Interactions Made Easy (MIME) dataset [117]. The aim is to study how well O_2A performs when pre-trained on a manipulation tasks dataset compared to a generic dataset. Also in Table 4.6, we tabulate details of other publicly available robotic manipulation tasks video datasets. MIME is the largest available video dataset containing both human and robotic demonstration videos.

MIME : Multiple Interactions Made Easy dataset

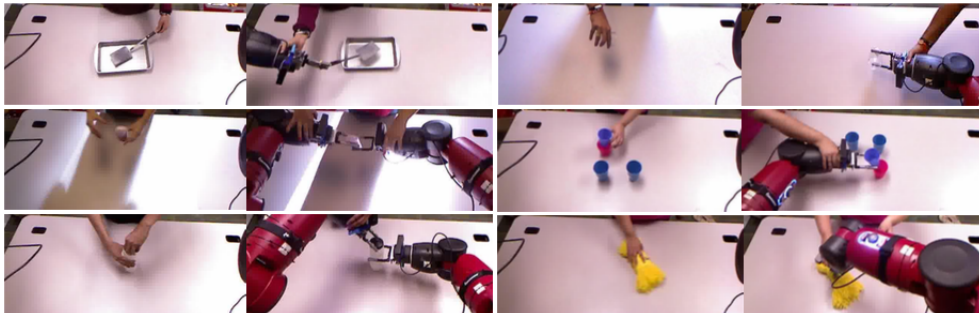


Figure 4.8: Selected few examples (execution by human and robotic manipulator) from MIME action dataset. The tasks are (clockwise from top-left) stirring, pouring, stacking, wiping, opening a bottle and passing. Figure from [18]

The MIME dataset consists of 8260 executions of 20 commonly seen object manipulation tasks. Each task execution is performed by a human as well as a robotic manipulator and is collected from four fixed viewpoints. Common household objects with variations in object appearances, pose and scale are used. The robotic executions are performed by a Baxter robot. Figure 4.8 shows selected examples

Table 4.6: Robotic manipulation tasks video datasets. MIME is the largest available robotic manipulation tasks video dataset containing both human and robotic demonstrations. However, building ImageNet[19] scale robotic manipulation datasets is crucial for future research.

	Release year	No: of classes	Video samples per class	Description
BAIR Dataset [126]	2016	1	59000	Contains pushing task demonstrations by a robot along with their corresponding joint trajectories.
DAML [162]	2018	3	~1200-2400	Contains both human and robot task demonstrations.
MIME [117]	2018	20	~800-2000	Contain human and robot task demonstrations along with corresponding joint trajectories.
RoboTurk-Sim [214]	2018	2	~1108	Only task demonstrations performed by a robot in simulation are present. Corresponding joint trajectories are also included.
RoboTurk-Real [215]	2019	3	~1429	Contain only task demonstrations performed by a robot in real world and the corresponding joint trajectories.

from the dataset. The list of 20 task categories in the MIME dataset is given in Appendix 6.8. The details of the action vector extractor pre-training on MIME dataset are given in Figure 4.9 and Table 4.7. This network is referred to as ‘NN:MIME’.

4.3. Simulation experiment

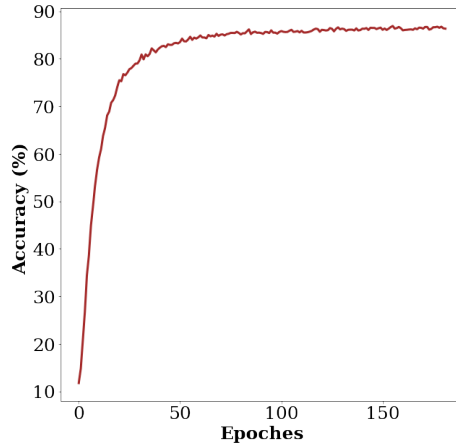


Figure 4.9: Testing accuracy (on MIME test set) per epoch during pre-training the action vector extractor on MIME dataset

Table 4.7: Details of pre-training the action vector extractor on the MIME dataset

	NN:MIME
Number of classes (NC)	20
Batch size	16
Input size	(16, 16, 112, 112, 3)
Output size	(16, 20)
GPUs used	2 x Nvidia P100
Training time	48 hrs
Optimizer	ADAGRAD [191]
Learning rate	0.001
Number of training examples	30,376
Number of testing examples	3,376
Total number of trainable parameters	78,077,716
Number of epochs	181
Best testing accuracy	86.90%

Identical settings (I)

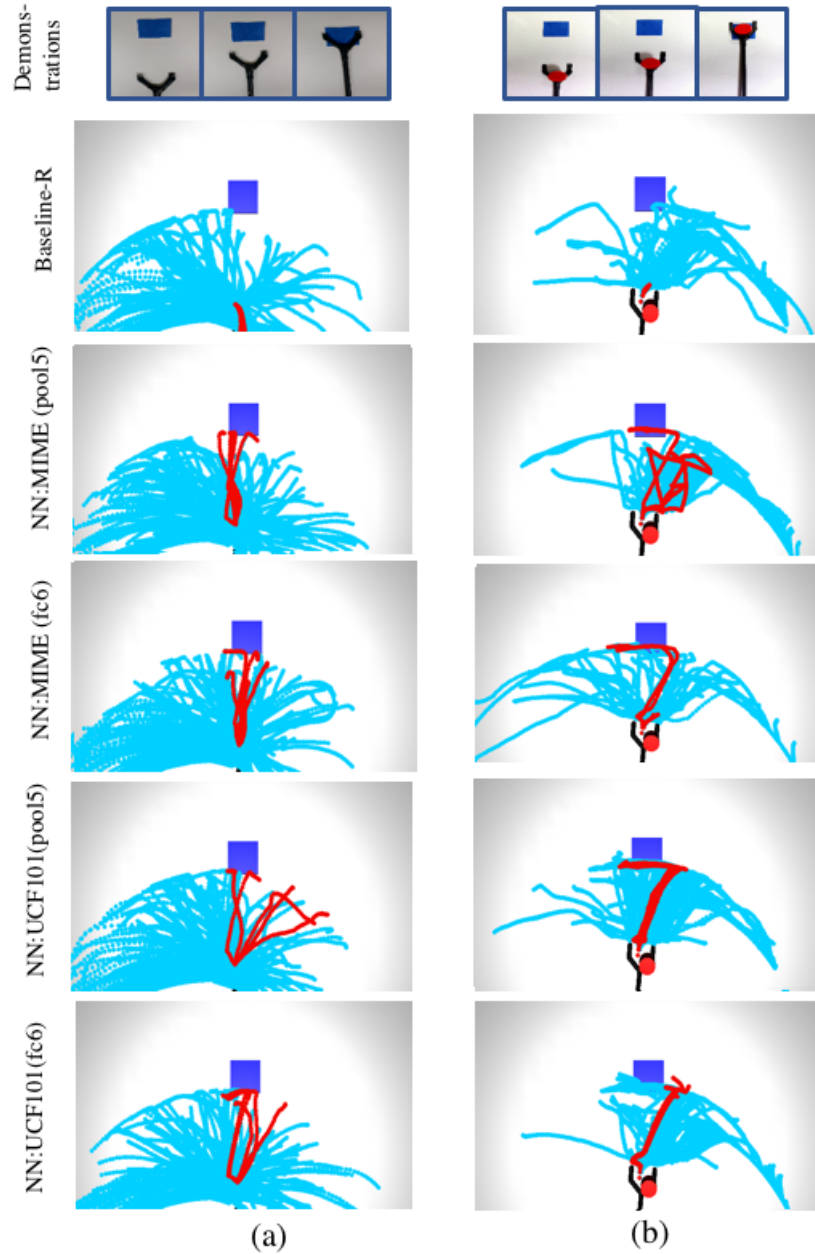


Figure 4.10: Trajectory maps obtained during task learning under identical domain settings for (a) reaching (left) (b) pushing (right). High rewards are obtained for desired trajectories in all the cases.

Change in viewpoint of observation (V)

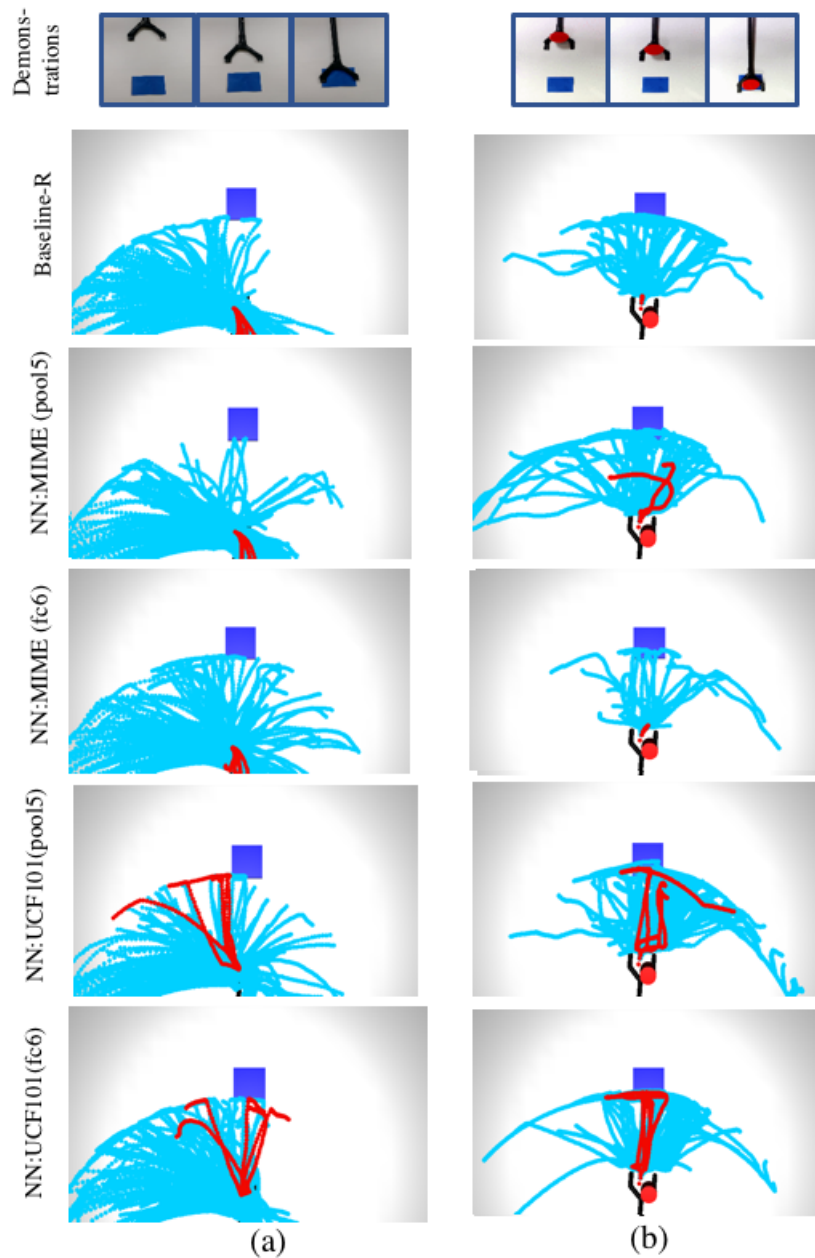


Figure 4.11: Trajectory maps obtained during task learning with changes in viewpoint of observation for (a) reaching (left) (b) pushing (right). High rewards are obtained for desired trajectories when the NN:UCF101 network is used.

Change in object properties (Obj)

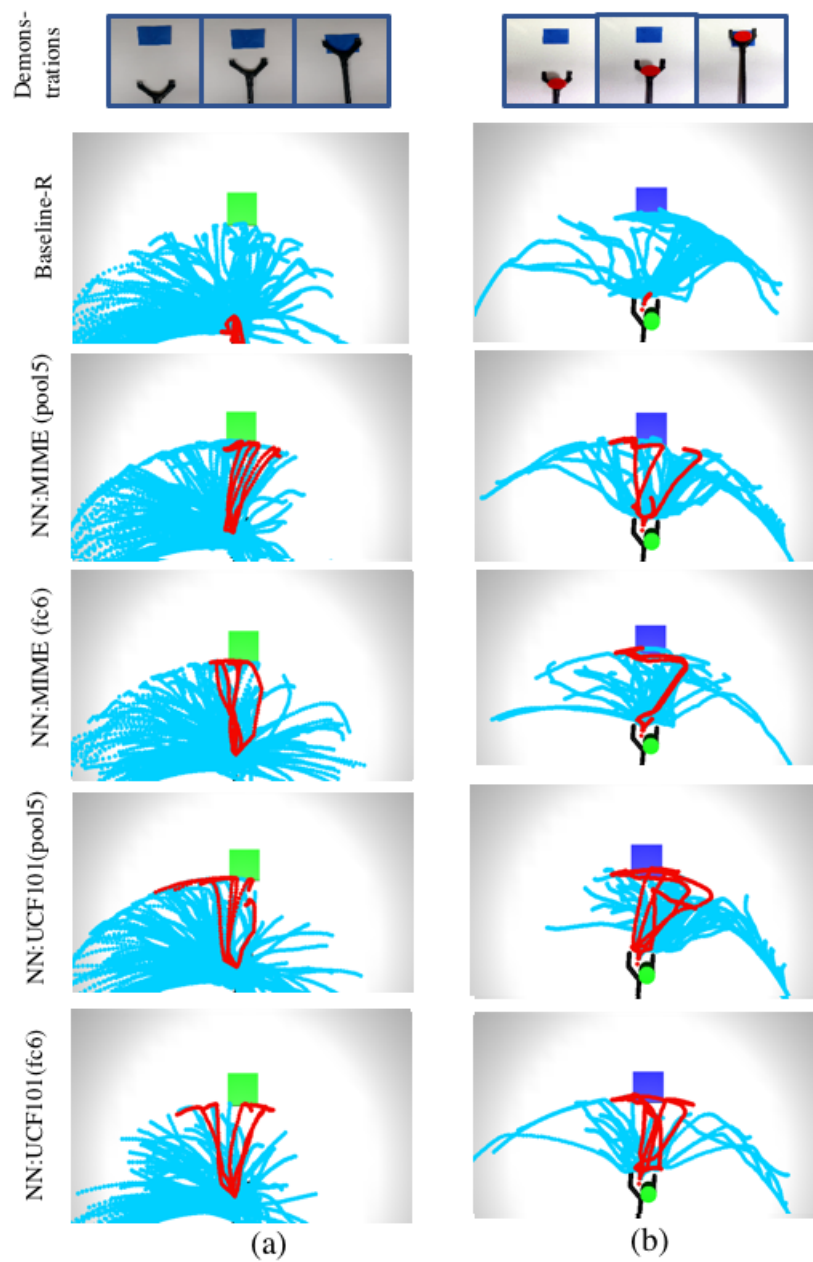


Figure 4.12: Trajectory maps obtained during task learning under identical domain settings for (a) reaching (left) (b) pushing (right). High rewards are obtained for desired trajectories in all the cases.

Change in object properties and viewpoint of observation (Obj+V)

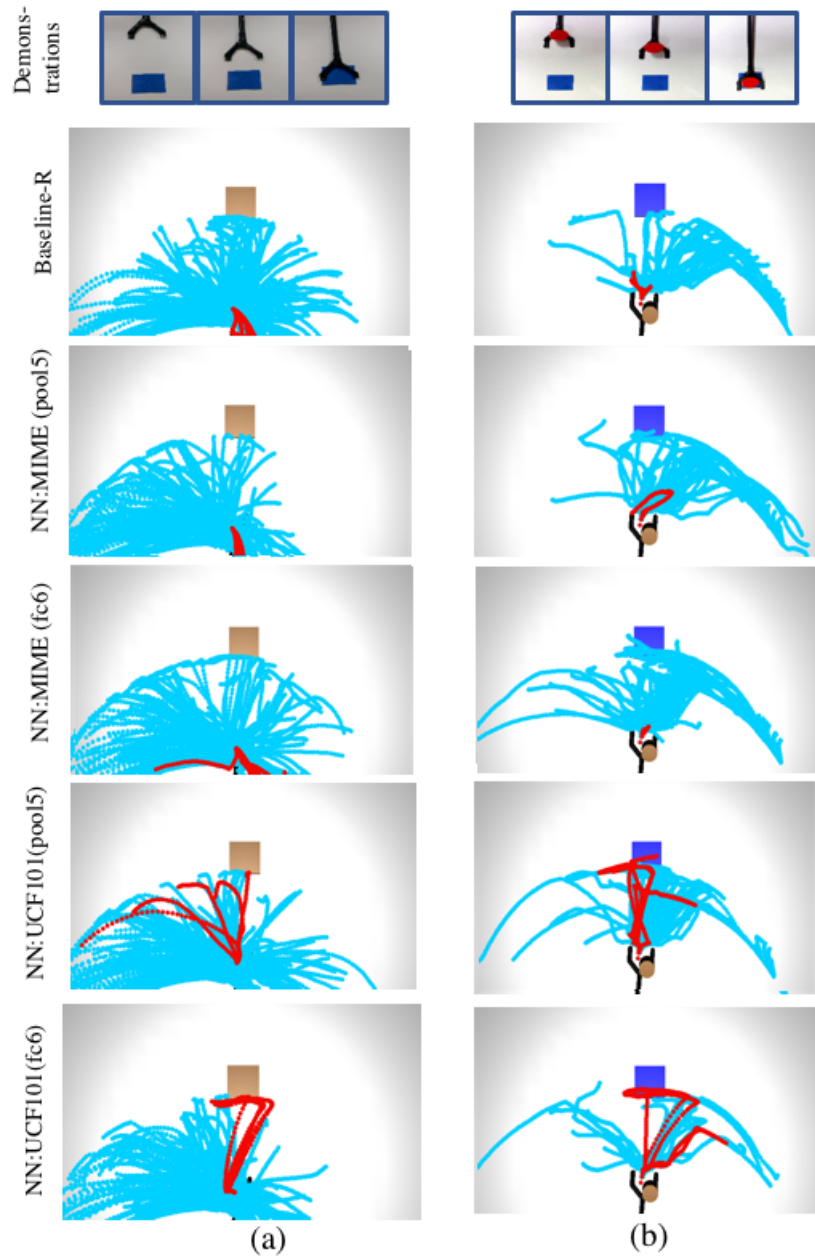


Figure 4.13: Trajectory maps obtained during task learning with changes in viewpoint of observation and object properties for (a) reaching (left) (b) pushing (right). High rewards are obtained for desired trajectories when the NN:UCF101 network is used.

Change in scene background (BG)

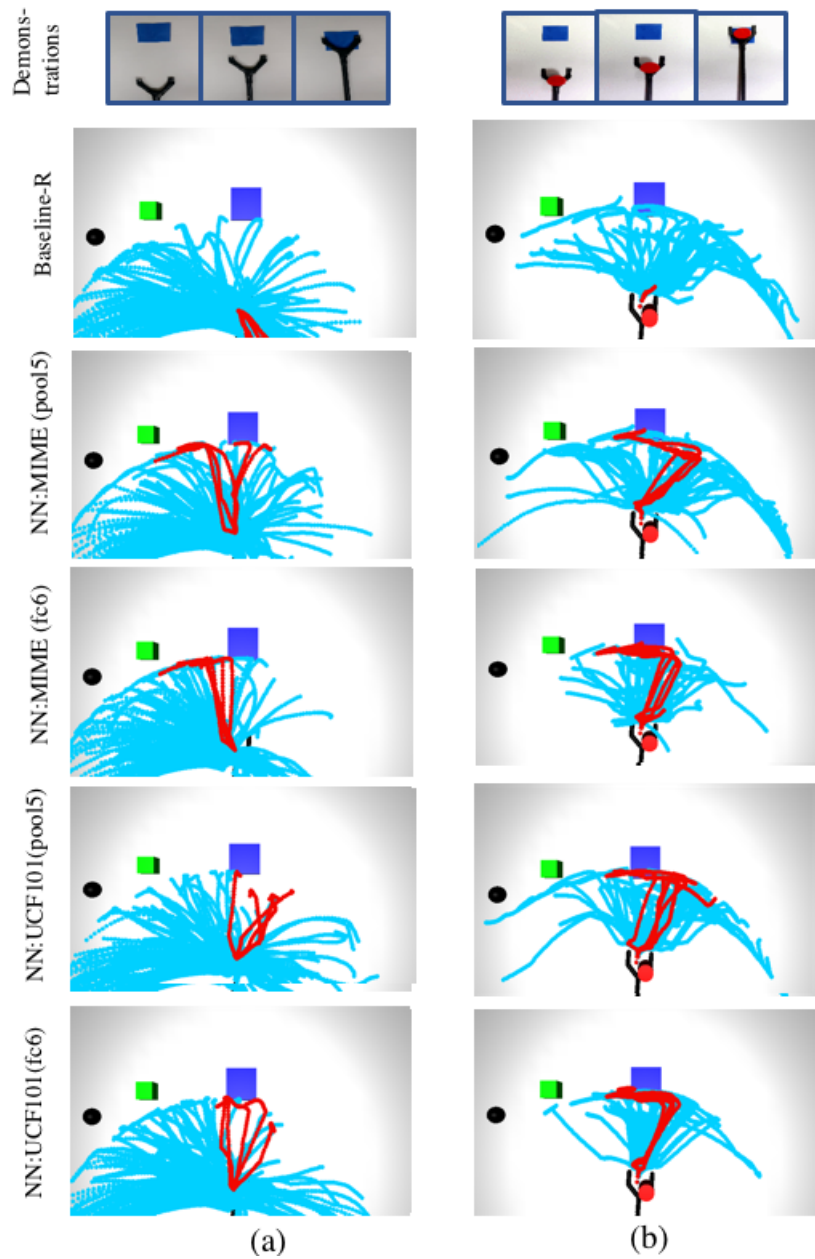


Figure 4.14: Trajectory maps obtained during task learning when background clutter is introduced in to the background for (a) reaching (left) (b) pushing (right). High rewards are obtained for desired trajectories in all the cases.

Change in manipulator (M)

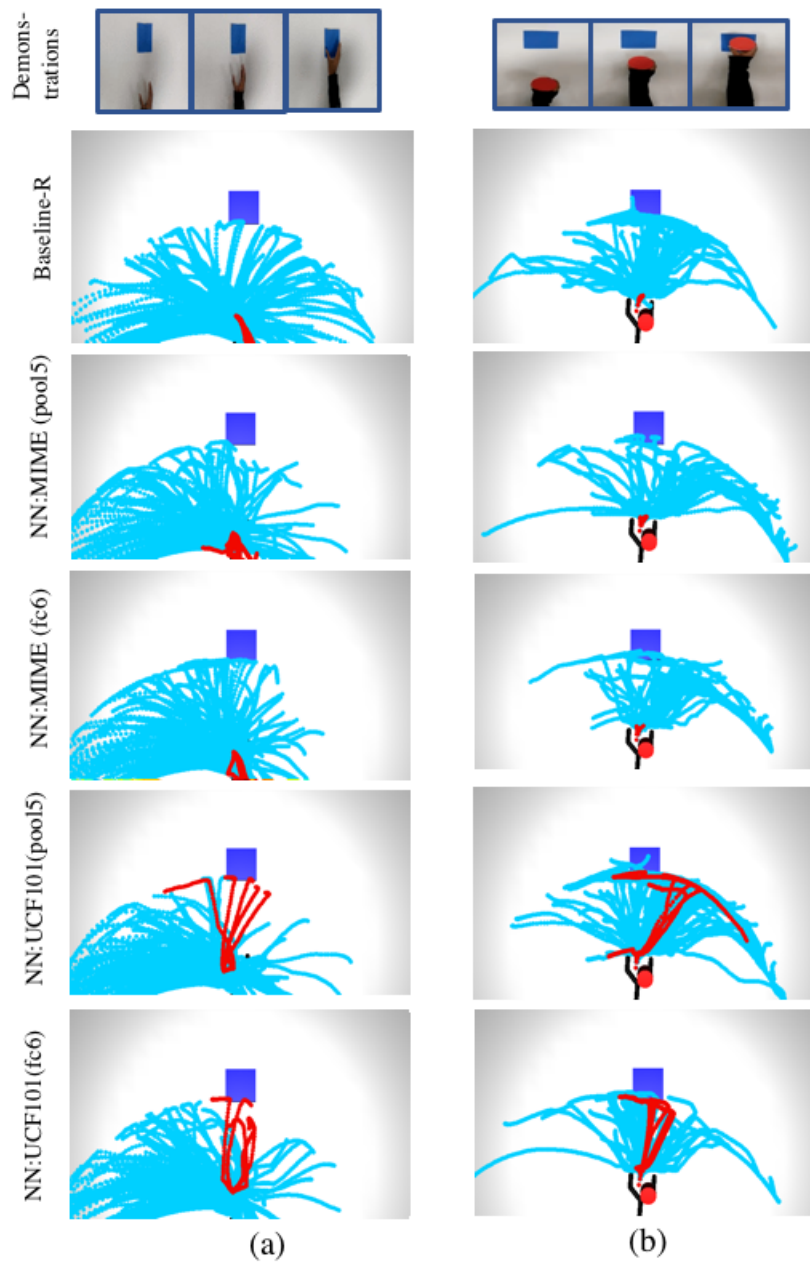


Figure 4.15: Trajectory maps obtained during task learning when demonstration is provided by a human hand for (a) reaching (left) (b) pushing (right). High rewards are obtained for desired trajectories only when NN:UCF101 network is used.

Discussion

The trajectory maps help to qualitatively analyse the performance of O_2A at a very fundamental level. We have collected the trajectory maps under different domain shifts characterised by changes in viewpoint of observation, object properties, scene background and manipulator morphology. We present our inferences below.

Baseline-R(random weights): The reward values for desired trajectories are low in all the cases as expected. This performance of the action vector extractor loaded with random weights serves as the baseline to evaluate other results.

NN:MIME (pool5 and fc6): When features from layers pool5 and fc6 are used as the action vector, the performance is satisfactory in three out of the six domain shifts. Desired trajectories obtain high rewards for: identical settings (I), changes in object properties (Obj) and scene background (BG). However, the rewards failed to identify the required trajectories when there were changes in viewpoint of observation (V, Obj+V) and manipulator morphology (M). This could be explained based on the MIME dataset used for pre-training the action vector extractor. Even though the dataset consist of a large number of manipulation task examples, the variations in terms of viewpoint of observation and manipulators are limited. The videos are collected from the same four fixed viewpoints of observation for all the tasks and also identical manipulators are used in all the examples. These limited variations prevent the action vector extractor from learning to generalise during pre-training.

NN:UCF101 (pool5 and fc6): When features from pool5 and fc6 layers of NN:UCF are used, a better performance is seen for all the domain shifts: I, V, Obj, Obj+V, M and BG. High rewards were obtained for desired trajectories in all these cases. This improved performance can be attributed to the diversity in the UCF101 dataset in terms of viewpoint of observation, object properties, manipulator morphologies and scene backgrounds.

These results show that pre-training O_2A with a generic action dataset performs better than a dataset of manipulation tasks specifically. Diversity in terms of actions and domain settings are important factors for O_2A to work. An interesting direction for future research would be to increase diversity in MIME dataset

using data augmentation techniques [216, 217] and then use for pre-training in O₂A.

Finally, we also performed a visual analysis of trajectory maps to identify outliers. Here the outliers are desired trajectories having low reward values and vice versa. However, we have not found any relevant outliers.

4.3.6 Limitations

To further understand the limitations of O₂A to domain shifts we plotted trajectory maps during reinforcement learning of tasks, when the task layout changes between the demonstration and the learning environment. Different layouts were generated by varying the relative positions of the objects and the manipulator between the demonstration and learning environments.

Change in task layout (L-L)

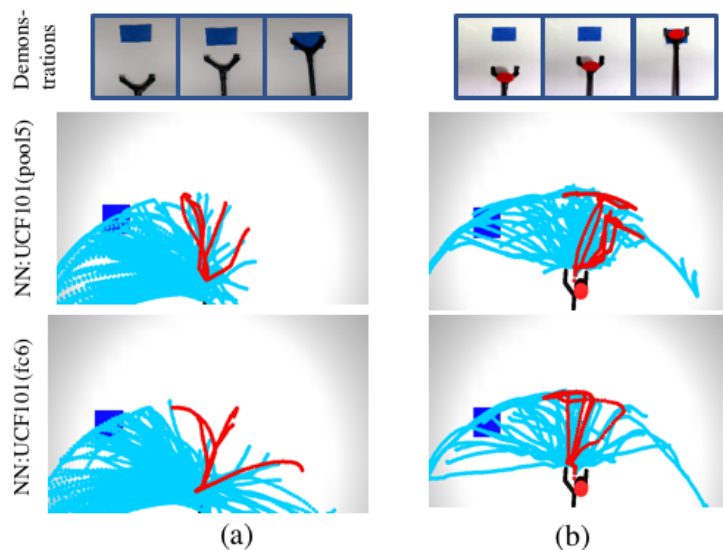


Figure 4.16: Trajectory maps obtained during task learning for (a) reaching and (b) pushing. New layout for the learning environment is generated by moving target region to the left. O₂A was unable to identify desired trajectories for the domain shift.

Change in task layout (L-R)

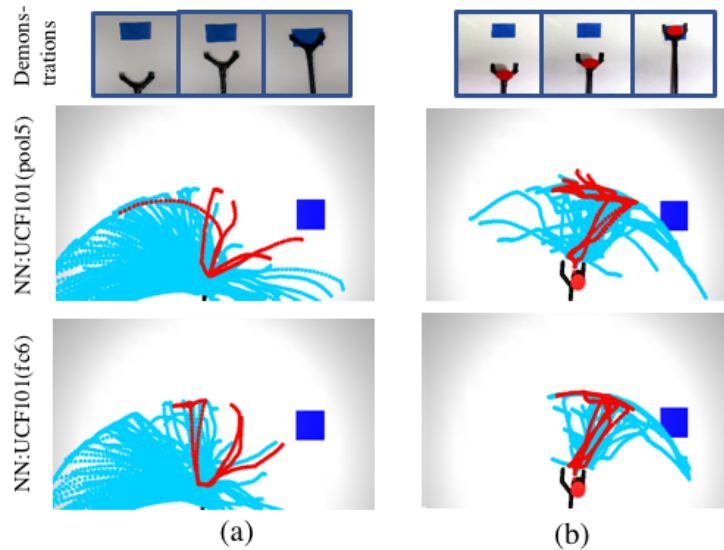


Figure 4.17: Trajectory maps obtained during task learning for (a) reaching and (b) pushing. New layout for the learning environment is generated by moving target region to the right. O_2A was unable to identify desired trajectories for the domain shift.

The results show that O_2A is unable to learn the demonstrated tasks of reaching and pushing when the task layout is different between the demonstration and the learning environments. This could be explained because of the lack of variations in the UCF101 dataset in terms of task layouts. The actions in the dataset occur predominantly at the center of the videos. Hence the action vector extractor could not learn to generalise to domain shifts characterised by task layout changes during pre-training. In future, this could be addressed by increasing task layout variations in the pre-training dataset.

4.3.7 Reach - Push

Finally, we also experimented with the reach-push task to show that O_2A can generate high rewards for desired trajectories for more complex tasks as well. We

4.3. Simulation experiment

manually collected a set of video samples showing varying degrees of task completion and calculated reward values with respect to a task demonstration. Figure 4.18 shows snapshots of the video samples including the task demonstration used. Reward values are calculated for each pair of videos (D-A to D-E) and is plotted in Figure 4.19. The reward values increase as the task moves towards completion. This shows that O_2A reward function can successfully model more complex tasks beyond reaching and pushing.



Figure 4.18: Snapshots of the video samples of the reach-push task collected. It includes a demonstration (D) and video samples showing varying degrees of task completion (A-E).

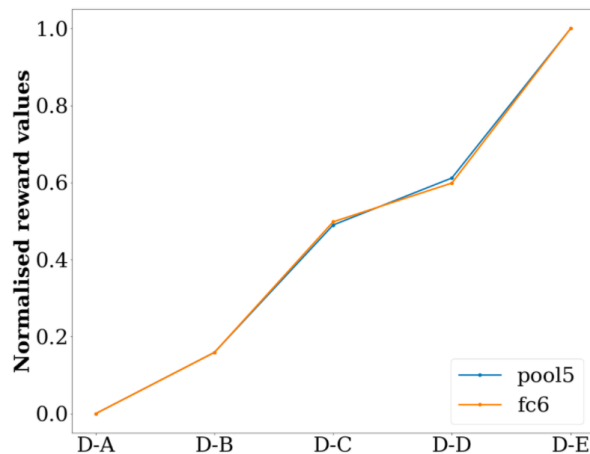


Figure 4.19: Normalised reward values for video pairs D-A to D-E. Higher rewards are obtained when task moves towards completion, showing O_2A can successfully model the more complete reach-push task as well.

4.4 Real robot experiment

The objective here is to evaluate the performance of O₂A in the real world, under different domain shifts. The tasks used are: pushing, hammering, sweeping and striking as shown in Figure 4.20. The task definitions and completion measures were detailed in Table 4.1. We use a 6-DOF UR5 robotic arm with different end effectors suitable for each task. All six domain shifts (see 4.3) were used for the pushing and hammering tasks. Whereas, only three domain shifts (I, V and M) were used for the sweeping and striking tasks, since others did not have a meaning for these tasks. We only used features from the pool5 layer of NN:UCF101 network as the action vector, due to the high cost of running the real robot experiment.

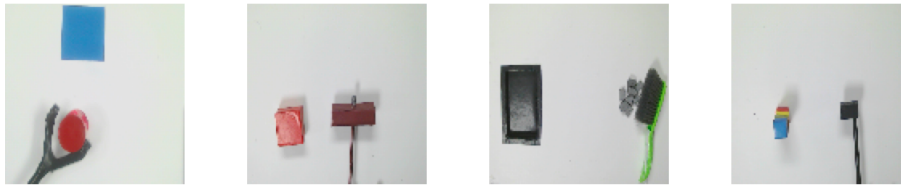


Figure 4.20: Real robot experiment tasks. From left-right: pushing, hammering, sweeping and striking

4.4.1 Stochastic Trajectory Optimisation (STO)

In the real robot experiment we use a manipulation planning algorithm, Stochastic Trajectory Optimization (STO) [218] [219]. Contrary to the RL algorithm, STO generates an optimal sequence of controls ($\mathbf{A} = (a_1, a_2, \dots, a_T)$) instead of an optimal control policy ($A_t = \pi(S_t)$). A different class of algorithms is used because deep RL (or model-free RL in general) is not suitable in real robots due to data-inefficiency. An alternative would have been to use a model-based RL algorithm, where a state transition function ($P(S_{t+1}/S_t, A_t)$) is first modelled. The optimal control policy is then learned using the state transition model, making it data efficient. But this approach has the following limitations:

- Curse of dimensionality: Modelling state transition models for high dimensional state spaces (like raw images) is still an active area of research and

remains a challenging problem [220]. An interesting direction for future work is to first condensate the state spaces into latent vectors [221] and then use them for transition function modelling.

- Unavailability of off-the shelf implementations: Unlike model-free deep RL algorithms, open-source libraries with model-based RL implementations are not widely available. Even though attempts have been made in this direction, like guided policy search library [222], the code bases are still work in progress.
- Need for domain expertise: Modelling state transition functions for real robots require domain expertise and a lot of practice. It is often described as an 'art than science' by practitioners in the field. Considering the aim and scope of our research this was not feasible to achieve.

However, using a different algorithm does not undermine the effectiveness of O₂A. Conversely, this displays its robustness to different manipulation control algorithms. We define the cost function C , to be minimized as:

$$C = r^2 \tag{4.10}$$

where r is the reward obtained from Equation 4.1.

Algorithm implementation

Briefly, we begin with an initial candidate control sequence. We execute this sequence using the manipulator to generate an initial cost. Thereafter, at each iteration we create 8 random control sequences by adding Gaussian noise to the candidate sequence from the previous iteration and execute them using the real robot. At the end of each iteration, we pick the control sequence with the minimum cost. Then set it as the new candidate sequence, thereby iteratively reducing the cost. The initial control sequence is initialised by providing a near solution path, following common practice [54]. This limitation can be an important topic of future

work for developing a more data efficient real world algorithm for robotic control. The Gaussian noise used has a mean of zero (as it is additive noise) and the standard deviation is set as a hyperparameter for each task.

4.4.2 Experimental setup

The real robot experimental setup is shown in the Figure 4.21. The setup consists of two sub-systems: the server and the execution system. The server runs the processing heavy functions while the execution system runs the robotic manipulator control and related functionalities. The sub-systems are integrated using the Robotic Operating system (ROS) [223]. The communications between server and execution system occurs through ROS nodes via ROS messages. All the programs, both at the server and the execution system are written using the python programming language.

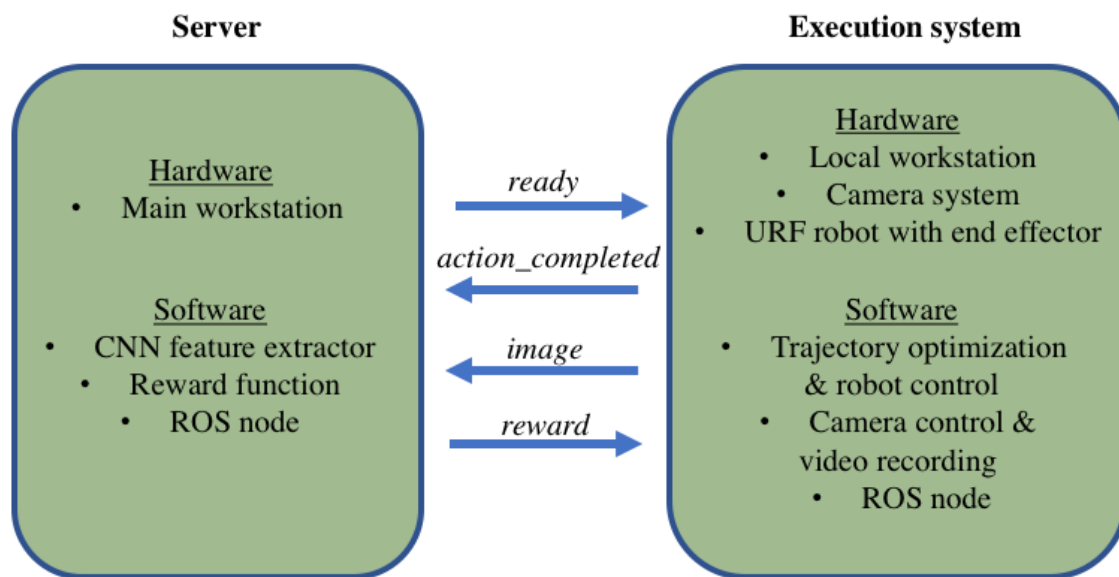


Figure 4.21: Real robot experimental setup overview

Server

The server is a desktop workstation with an Intel core-i7 processor, 32GB RAM and 1TB of memory and has an Linux Ubuntu16.04 operating system with ROS

Kinetic running in a virtual machine (within VirtualBox). It hosts the action vector extractor CNN with Tensorflow, the reward function and the server ROS node which controls the flow of ROS messages in and out of the system.

Execution system

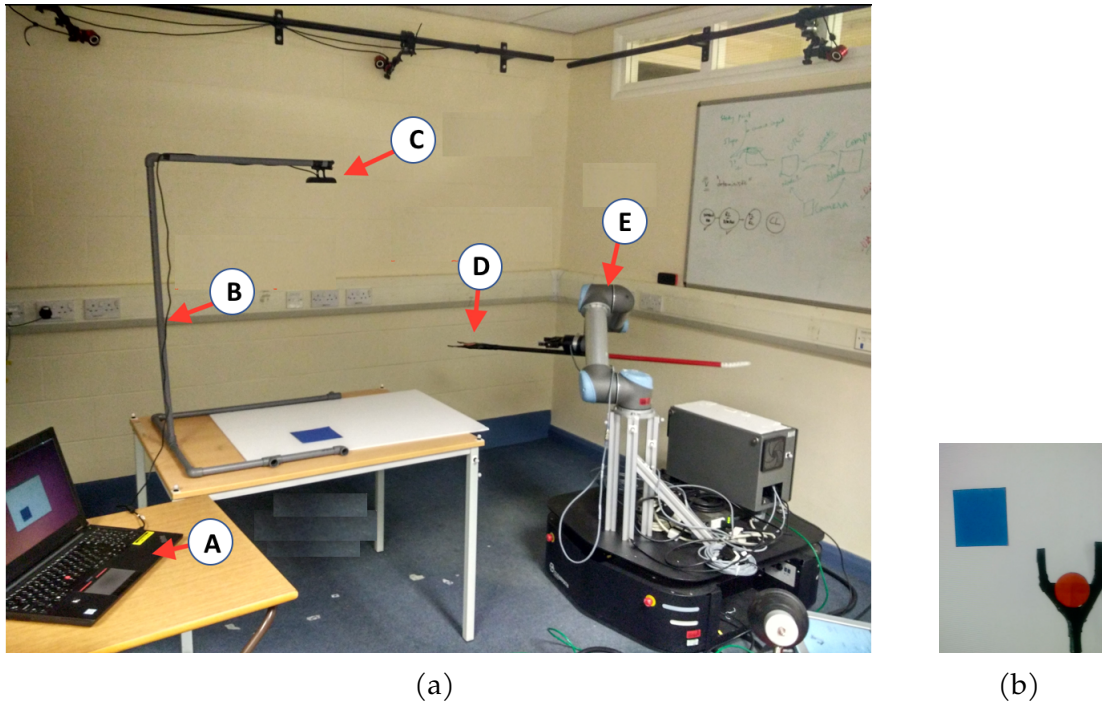


Figure 4.22: (a) Execution system experimental setup built for running real robot experiments. It consists of a local laptop workstation (A), custom made camera holder (B), camera (C) and the end effector (D) attached to the UR5 robot (E). (b) View of the robot.

The execution system as shown in Figure 4.22 consist of a local laptop workstation, camera and a UR5 robotic arm. The laptop workstation has a Intel core-i5 processor, 8GB RAM and 500GB memory and has an Linux Ubuntu14.04 operating system with ROS Indigo. The execution system hosts the STO algorithm that generates control sequences, the program that enable the robotic manipulator to execute the generated control sequence and the execution system ROS node for regulating message flow. It also hosts the camera control program that records

the robot actions during each trial execution. The camera used here is a HD 720p webcam.

Communications

The communications between server and the execution system is through ROS messages. Messages are sent under 5 different topics:

- *Ready* message: The ready message is sent from the server to the execution system. It indicates that the CNN action vector extractor is loaded with pre-trained weights and is ready to process data.
- *Action_completed* message: The message indicates the end of the execution of a control sequence by the robotic manipulator. It is sent from the execution system to the server.
- *Image* message: After every control sequence is executed, the recorded video of this robot action execution is sent to the server from the execution system. The videos are sent frame by frame under the image message ROS topic.
- *Reward* message: The reward message carries the reward from the server to the execution system. The STO algorithm uses this reward to generate a control sequence after every iteration.

Overall working

Firstly, the action vector extractor is loaded with pre-trained weights in the server. After that *ready* message is sent to the execution system. The execution system then runs the STO algorithm generating a control sequence to be executed by the robotic manipulator. While the control sequence is being executed, the camera control program records the robot action and sends them to the server. After the control sequence is executed, the *action_completed* message is sent to the server. Upon receiving this message the action vector extractor extracts the action vector from the video of robot action that was executed. The reward function then generate a reward using this action vector and that of the demonstration which is sent to the execution system. The stochastic trajectory optimization algorithm

uses this reward to calculate the next control sequence to be executed. This process continues until an optimal control sequence is obtained.

4.4.3 Results

Each experiment is run twice. The average task completion measures of the optimal control sequence obtained with the standard deviations are plotted in Figure 4.24. The snapshots of executions of optimal control sequences for the selected domain shifts along with the demonstrations are given in Figure 4.23. Our method achieves good task completion measures for different domain shifts. This shows the effectiveness of O_2A in learning tasks on a real robot. Videos of the results including demonstrations are available in our [webpage](#).

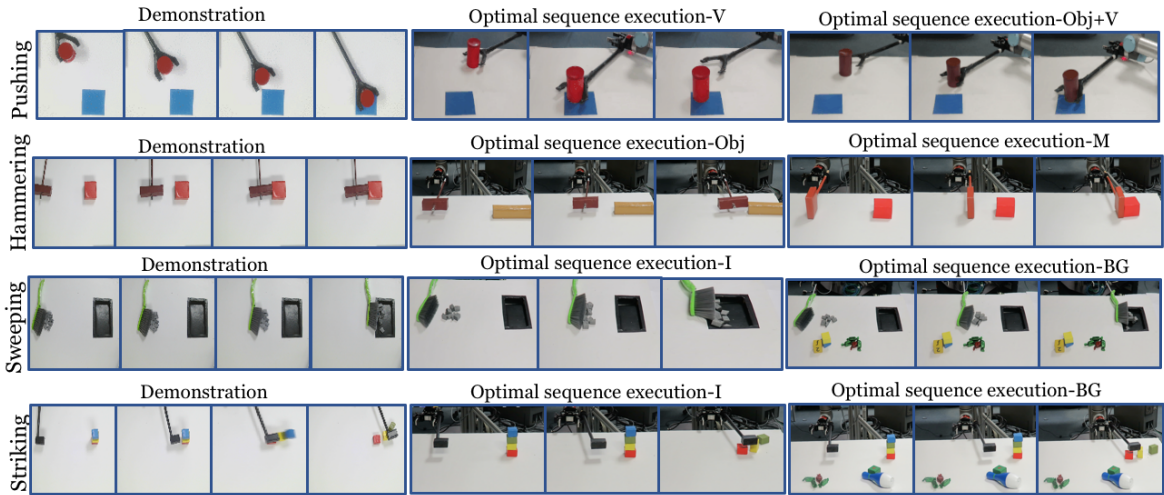


Figure 4.23: Snapshot of the demonstration and execution of the corresponding optimal control sequences obtained for selected domain shifts from the real robot experiment (Results shown for action vectors extracted from the pool5 layer of the NN:UCF101 network).

4.5 Conclusion

In this chapter we presented O_2A , the method for one-shot observational learning using action vectors. The action vectors are extracted from the single demonstration provided and also from the trial robot executions. A reward is generated by

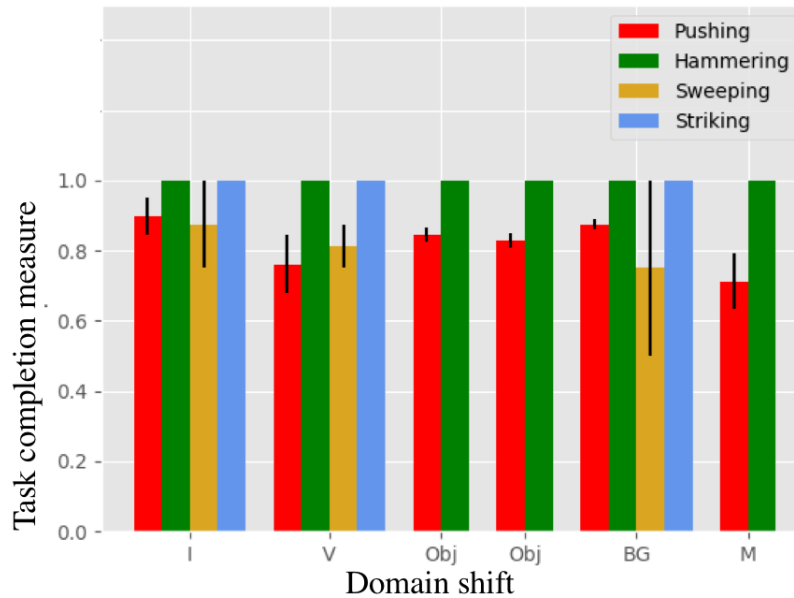


Figure 4.24: Task completion measures for the task of pushing, hammering, sweeping and striking in the real robot experiment. The result shows that O_2A performs well under different domain shifts on a real robot.

comparing these action vectors using Euclidean distance between them. This reward is then used to guide the learning algorithms to obtain an optimal control policy or a control sequence for carrying out the demonstrated task.

We conducted extensive experiments to evaluate the performance of our method under different domain shifts. Using trajectory maps, we showed that high rewards are obtained for desired robotic trajectories when features from the pool5 and fc6 layers of the NN:UCF101 network are used as the action vector. However, the O_2A failed to learn the demonstrated task when task layouts varied between the demonstration and learning environments. We also compared our method with two baselines and an oracle using task completion measures and Pearson correlation coefficients. Our method outperformed the baselines in both the measures. Finally, we conducted real robot experiments to demonstrate the effectiveness of O_2A in real world conditions.

**

Conclusion and Future Work

This chapter concludes the thesis with an overview of the presented research, its limitations and detailing the possible future directions of our research.

5.1 Conclusion of the thesis

In this thesis, we investigated the problem of observational learning, with an aim to develop an one-shot observational learning method which only requires a single demonstration to learn new robotic manipulation tasks. In Chapter 1, we defined the ‘problem of observational learning’ in detail. We explained the three stages in observational learning: observation, feature representation extraction and task execution. We also outlined how observational learning differs from imitation learning due to domain shifts and lack of access to joint trajectories in the demonstrations.

In Chapter 2, the existing observational learning methods were reviewed extensively. Initially we discussed the two types of observations: assisted and direct. Assisted observation uses methods like trackers, visual object detectors, motion capture systems or skeleton tracking. However, with the advent of deep learning, direct observation of the demonstrations was possible. Feature representations are extracted directly from the raw videos of the demonstration. But the methods we reviewed had a major drawback in the number of demonstrations required for learning of a new task. Collecting demonstrations in large numbers is not possible

in practical scenarios. Also these methods are not completely robust to domain shifts. Hence we introduced O_2A , the one-shot observational learning method with robustness to different domain shifts. We studied the domain shifts caused by changes in viewpoint of observation, object properties, scene background and manipulator morphology in this thesis.

In Chapter 3, we presented the core concept of O_2A , the action vectors. Action vectors are abstract task-discriminative and domain-agnostic perceptual representation of a task in a video. We presented a novel way of extracting action vectors from unseen manipulation tasks using a network pre-trained for action recognition. We pre-trained a 3D CNN using a generic action dataset, the UCF101 dataset. For evaluating the generalisation of the pre-trained action vector extractor to unseen manipulation tasks, we collected a new dataset, the LMD. LMD has 3 task classes: reach, push and reach-push collected under identical domain settings. We conducted clustering analysis on action vectors from LMD, when features from pool5, fc6, fc7 and fc8 layers are used. The analysis showed that the generalisation is the highest for features from pool5 and fc6 layers. The layers have ARI scores of .26 and .34 respectively, compared to a baseline score of .07. Also, using features concatenated from different layers as action vectors did not show a significant improvement in performance.

Further, we calculated inter-class and intra-class similarity scores when features from pool5 and fc6 layers are used as the action vectors. We noted high intra-class similarity scores in both cases. Finally, visualisation of LMD in the action vectors space showed emergence of a task-class based clustering compared to the baseline. The results showed that the action vector extractor pre-trained on a generic action dataset can generalise to unseen manipulation tasks. And also the extracted action vectors are task-discriminative and domain-agnostic.

In Chapter 4, we explained O_2A in detail. We showed how action vectors can be used for achieving one-shot observational learning. The action vectors from the trial robot executions are compared with the action vector from the demonstration video and a reward is obtained. The DDPG RL algorithm then finds an optimal control policy by optimizing this reward. We conducted experiments both in a simulation and with a real robot to evaluate O_2A . The simulation experiment environments were designed using OpenAI Gym RL framework for the

the tasks of reaching and pushing. We calculated task completion measures for each of the tasks and compared the results with baselines. The baselines were obtained when features from a VGG network pre-trained on a ImageNet dataset and HOGG features were used as the perceptual task representations respectively. The results showed that our method outperforms both the baselines under different domain shifts of changes in viewpoint of observation, task properties, scene background and manipulator morphology. We also compared O_2A and the baseline approaches by calculating Pearson correlation coefficients with hand-engineered oracle rewards. The coefficients were highest in most of the cases and positive for O_2A rewards compared to the baselines.

Further, we introduced the novel concept of the trajectory maps, which are visualisations of trajectories taken by the robotic manipulator during task learning. The visualisations helped to understand which trajectories had the highest rewards. The trajectory maps showed that O_2A had high rewards for desired trajectories under different domain shifts. We also studied the performance when O_2A was pre-trained on a robotic manipulation dataset, the MIME dataset. Our results showed that that O_2A works best when UCF101 (a generic action dataset) is used rather than a more specific MIME (a manipulation tasks dataset). Specifically, O_2A pre-trained with MIME dataset failed to identify desired trajectories when domain shifts were caused by changes in viewpoint of observation and manipulator morphology. We conclude that the variations in a dataset are more important than its specificity, in determining the generalisation of a pre-trained network. The limitations of O_2A were also studied. Our method was unable to provide high rewards for desired trajectories when the domain shift was caused by changes in task layout. We account this limitation to the lack of variations in UCF101 in terms of task layouts, as the actions always happen in the center of the video frames.

Finally, we evaluated O_2A on a real UR-5 robot. We used a robotic manipulation planning algorithm, the STO. Using a different algorithm did not undermine the effectiveness of O_2A . Conversely, it displayed its robustness to different manipulation control algorithms. We conducted the experiment with four tasks: pushing, hammering, sweeping and striking, under different domain shifts. Our method achieved good task completion measures for all the tasks under domain

shifts.

5.2 Limitations

Limitations of our research are detailed below:

- In Chapter 3, the locally collected LMD dataset for action vector analysis is limited in terms of number of tasks and video samples per task. Collecting a manipulation tasks dataset with more task classes and sample videos per class would provide an extensive action vector analysis. A human intervention free robot data collection strategy similar to Levine et al. [224] can be modified and used. We could let a robot perform random actions and set up a visual/mechanical detection system to identify if the robot have performed the desired task. For example, in a task such as hammering, the task completion detection could be via a touch sensor at the end of the manipulator.
- In Chapter 3, our current formulation of O_2A , relies on sparse rewards. The rewards are obtained at the end of each episode rather than after each step. This could make learning of more complicated tasks difficult. The following formulation of O_2A would be able to obtain rewards after each step and thereby making learning more efficient:

With reference to Section 4.2, for every time step t in RL, a set of frames $\{I_t, I_{t-1}..I_{t-N_f+1}\}$ can be fed to the action extractor to obtain \mathbf{X}_R (N_f is the number of input frames to the extractor network). A reward can then obtained from \mathbf{X}_R using equation 4.1. Thus every tuple (S_t, A_t, r_t, S_{t+1}) will have a non-zero reward value that can be used for reinforcement learning of the task.

- In Chapter 4, O_2A is evaluated only for a limited number of instances per domain shift due to the limitations in computational resources. With availability of more resources in future, O_2A could be evaluated for a variety of instances per domain shift. Some examples are:

- Evaluation for every degree of domain shift between 0-360°, will provide a comprehensive insight into how reward patterns vary with changes in viewpoint.
- Using manipulators with varying degrees of freedom, ideally ranging from 1 to 10.
- In Chapter 4, O₂A requires a large number of trial robot executions to learn the demonstrated task. This limits the deployability of our method in the real world. Possible solutions are to reuse control policies or to generate an optimal control sequence directly from the demonstration as detailed in Section 5.3.3.
- In Chapter 4, the robotic experiments are conducted with a set of 2D tasks: reaching and pushing (in simulation) and pushing, hammering, sweeping and striking (with the real robot). However, it would be interesting in future to use more complex 3D tasks such as pouring or flipping objects to further study the generalisation and limitations of O₂A.

5.3 Future works

Our research has further led to more open questions. Here we detail the possible extensions and future directions to the presented work and our initial research into some of these areas.

5.3.1 Multi-modal (one-shot) observational learning

Observational learning by incorporating other sensing modalities is an important direction for future work. Humans combine multiple sensing modalities while observing a demonstration. For example, sound plays an important role in identifying the keys pressed while observing a piano demonstration. Similarly other modalities such as tactile [225, 226] and olfactory [227] sensing could also be used to observe demonstrations.

We have conducted some preliminary research into using the sound modality for observing demonstrations. We extracted an action vector like abstract representation of sound waves, the '*Sound Vector*'. For our experiment we have used the MFCC features [228] extracted from sound waves as the sound vector. We report our results on a locally collected evaluation dataset, the 'Leeds Sound Dataset' (LSD). LSD consists of sounds samples from two different classes: 'hammering on target' (Class-1) and 'hammering on table' (Class-2), that are generated while performing the manipulation task of hammering a target object. Table 5.1 shows the inter-class and intra-class distances for LSD in the sound vector space. The intra-class distance is lower compared to inter-class distance, showing promise of using sound vectors for observational learning.

Table 5.1: Inter-class and intra-class distances for LSD in the action vector space. Class-1 is 'hammering on target' and Class-2 is 'hammering on table'.

	Distance
Intra-class distance: Class-1	742.33
Inter-class distance: Class-1 and Class-2	1525.33

We also attempted generating sound vectors using a pre-trained 2D CNN. An overview of this approach is shown in Figure 5.1. First the sound waves were converted into audio spectrograms. And then we extracted sound vectors from these spectrograms using a 2D CNN network pre-trained for sound classification. We used the UrbanSound8k [229] dataset as the generic sound dataset for pre-training. This approach also presents a promising future direction. Furthermore, it would be also interesting to see if the concept of domain shifts applies to sound vectors.

5.3.2 Pre-training objectives

In our research we have only used action recognition as the objective for pre-training the action vector extractor. A direct extension of our research would be to explore using other objectives for pre-training. Supervised objectives like contrastive loss [230] and unsupervised objectives like future frame prediction [231]

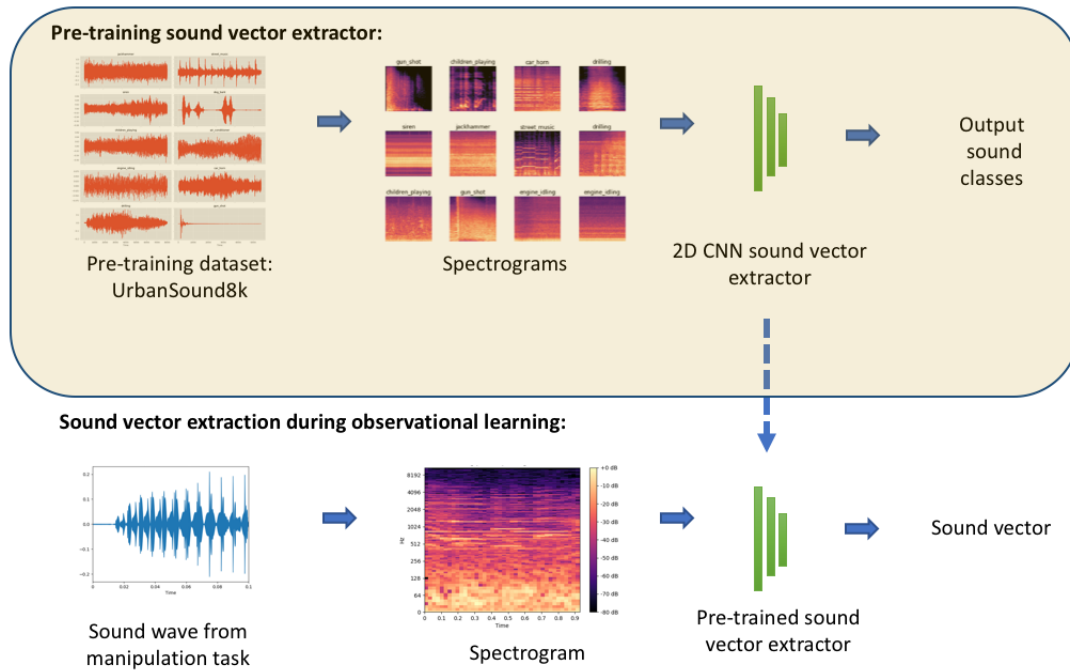


Figure 5.1: Overview of pre-training sound vector extractors using UrbanSound8k dataset and its usage during observational learning.

could be used. An added advantage for using unsupervised objectives is that it does not require class labels during pre-training. Unsupervised objectives could also be used for fine-tuning the extractors after a supervised pre-training.

5.3.3 Reducing number of trial robot executions

Another extension of our work is to reduce the number of trial robot executions required to learn the demonstrated task. Two possible approaches for this are:

- **Control policy re-use [232]:** Instead of the current practice of reinforcement learning of optimal control policies from scratch, we can reuse control policies across different tasks and task instances. Future research could investigate to what extent and order can the policies be reused.
- **Map action vectors directly to robot controls:** Currently we use RL/STO algorithms to learn the demonstrated task by optimizing the reward/cost

function. These algorithms could be replaced by a mapping function that can directly map the action vector of the demonstration to the robotic controls. Ideally a fully connected neural network could be used as the mapping function. A training dataset for this mapping function could be created by collecting action vectors of a set of random robot executions and their corresponding robotic controls. After training, the mapping function can instantaneously generate robotic controls for any new task demonstration video.

5.3.4 O₂A beyond manipulation tasks

An exciting future direction of this work will be to explore using O₂A beyond manipulation tasks. The pre-trained action vector extractor should be able to extract meaningful perceptual representations for other problems which share similar visual dynamics. Briefly we experimented with a modified OpenAI Gym LunarLander RL environment shown in Figure 5.2. Our initial experiment shows a positive correlation of $.6167 \pm .1265$ between perceptual rewards in O₂A and an Oracle reward under identical conditions, which is promising.

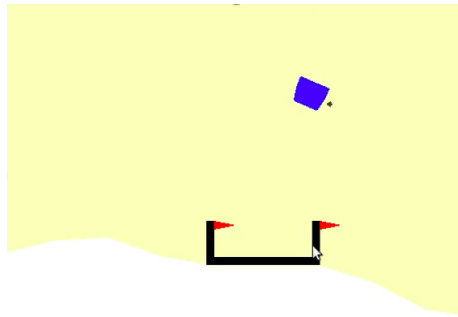


Figure 5.2: Modified LunarLander RL environment

5.3.5 Observational learning of long horizon manipulation tasks

Another direction of our research could extend O₂A to long horizon manipulation tasks such as cooking. One approach to tackle this could be to decompose them

5.3. Future works

into several shorter tasks learnt using the current method, within a curriculum learning framework [233].

**

6.1 Recent Action datasets

Recently released advanced datasets with fine-grained action classes are given in Table 6.1 which could be used in future research.

Table 6.1: Recently released larger action datasets

	Release year	No: of action classes	Video samples per class	Resource
20BN V2[234]	2018	174	~620	Crowd sourcing
HACS[235]	2019	200	~7500	Web videos
HVU[236]	2019	739	~1994	Other datasets
MovieNet[237]	2020	80	~562	Movies
Kinetics-700-2020[238]	2020	700	>700	YouTube

6.2 UCF101 Dataset

The details of the UCF101 action recognition dataset used for pre-training the action vector extractor are given in Table 6.2.

Table 6.2: UCF101 action recognition dataset details

Dataset property	Value
Total no: of videos	13320
Mean video length	7.21 sec
Total duration	1600 min
Min: video length	1.06 sec
Max: video length	71.04 sec
Frame rate	25 fps
Resolution	320 x 240
Audio	Yes (51 videos)

6.2.1 Action catagories

The 101 action categories for UCF101 dataset are: Apply Eye Makeup, Apply Lipstick, Archery, Baby Crawling, Balance Beam, Band Marching, Baseball Pitch, Basketball Shooting, Basketball Dunk, Bench Press, Biking, Billiards Shot, Blow Dry Hair, Blowing Candles, Body Weight Squats, Bowling, Boxing Punching Bag, Boxing Speed Bag, Breaststroke, Brushing Teeth, Clean and Jerk, Cliff Diving, Cricket Bowling, Cricket Shot, Cutting In Kitchen, Diving, Drumming, Fencing, Field Hockey Penalty, Floor Gymnastics, Frisbee Catch, Front Crawl, Golf Swing, Haircut, Hammer Throw, Hammering, Handstand Pushups, Handstand Walking, Head Massage, High Jump, Horse Race, Horse Riding, Hula Hoop, Ice Dancing, Javelin Throw, Juggling Balls, Jump Rope, Jumping Jack, Kayaking, Knitting, Long Jump, Lunges, Military Parade, Mixing Batter, Mopping Floor, Nun chucks, Parallel Bars, Pizza Tossing, Playing Guitar, Playing Piano, Playing Tabla, Playing Violin, Playing Cello, Playing Daf, Playing Dhol, Playing Flute, Playing Sitar, Pole Vault, Pommel Horse, Pull Ups, Punch, Push Ups, Rafting, Rock Climbing Indoor, Rope Climbing, Rowing, Salsa Spins, Shaving Beard, Shotput, Skate Boarding, Skiing, Skijet, Sky Diving, Soccer Juggling, Soccer Penalty, Still Rings, Sumo Wrestling, Surfing, Swing, Table Tennis Shot, Tai Chi, Tennis Swing, Throw Discus, Trampoline Jumping, Typing, Uneven Bars, Volleyball Spiking, Walking with a dog, Wall Pushups, Writing On Board, Yo Yo.

6.3 Feature selection

We also perform supervised feature selection on the features from pool5 and fc6 layers. We then use the subsets of selected features as the action vector. For supervised selection of features we collect a supplementary dataset, the LMD_Sup dataset. It contains 10 video samples each of two manipulation task classes: move and displace, examples of which are shown in Figure 6.1. Note that, LMD_Sup is only used for supervised feature selection. The subset of features are selected by calculating ANOVA F-value [239] for each feature using LMD_Sup. The results are shown in Figure 6.2. ARI scores on LMD are calculated when top 20%, 40%, 60%, 80% and 100% of the features are used as the action vector respectively. The results show an improvement in ARI scores when the top 20% of the features from pool5 and fc6 layers are used. These results are promising and indicate the existence of a subset of features, that are more significant than others to represent manipulation tasks used for observational learning. It will be an interesting direction for future research to identify and extract these subsets of relevant features.

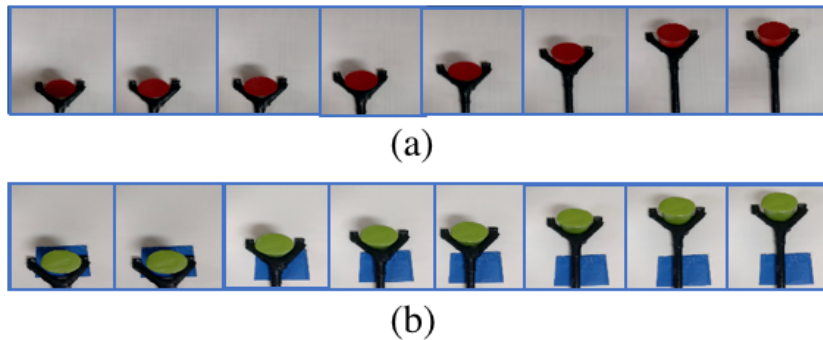


Figure 6.1: Examples from LMD_Sup dataset for the task classes (a) Move and (b) Displace.

6.4 3D visualisation

Here we visualize the action vectors from LMD in 3D, which are shown in Figures 6.3, 6.4 and 6.5. The clustering of action vectors from the same classes, when compared to the Baseline-R is evident.

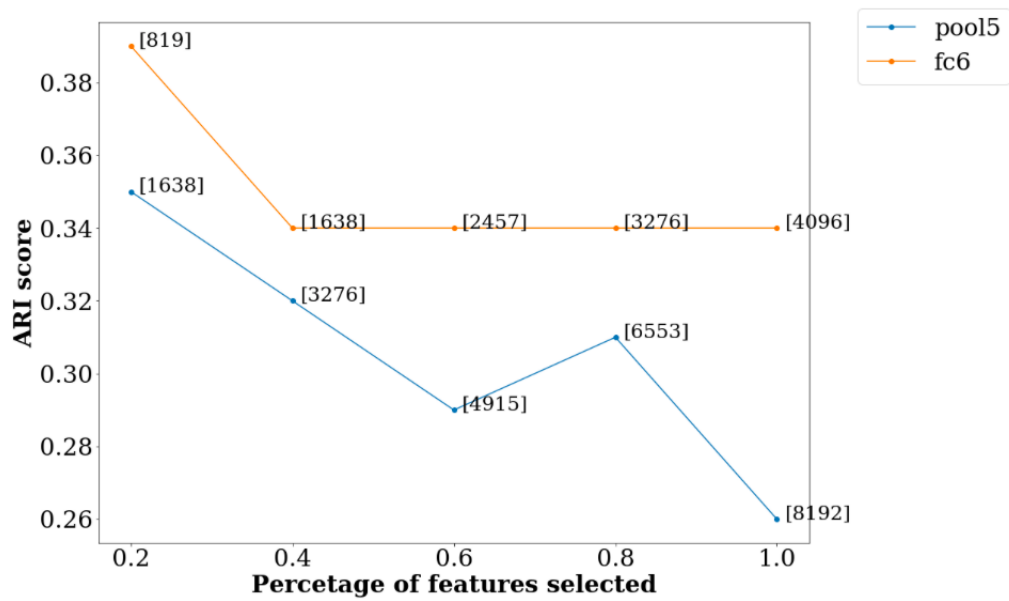
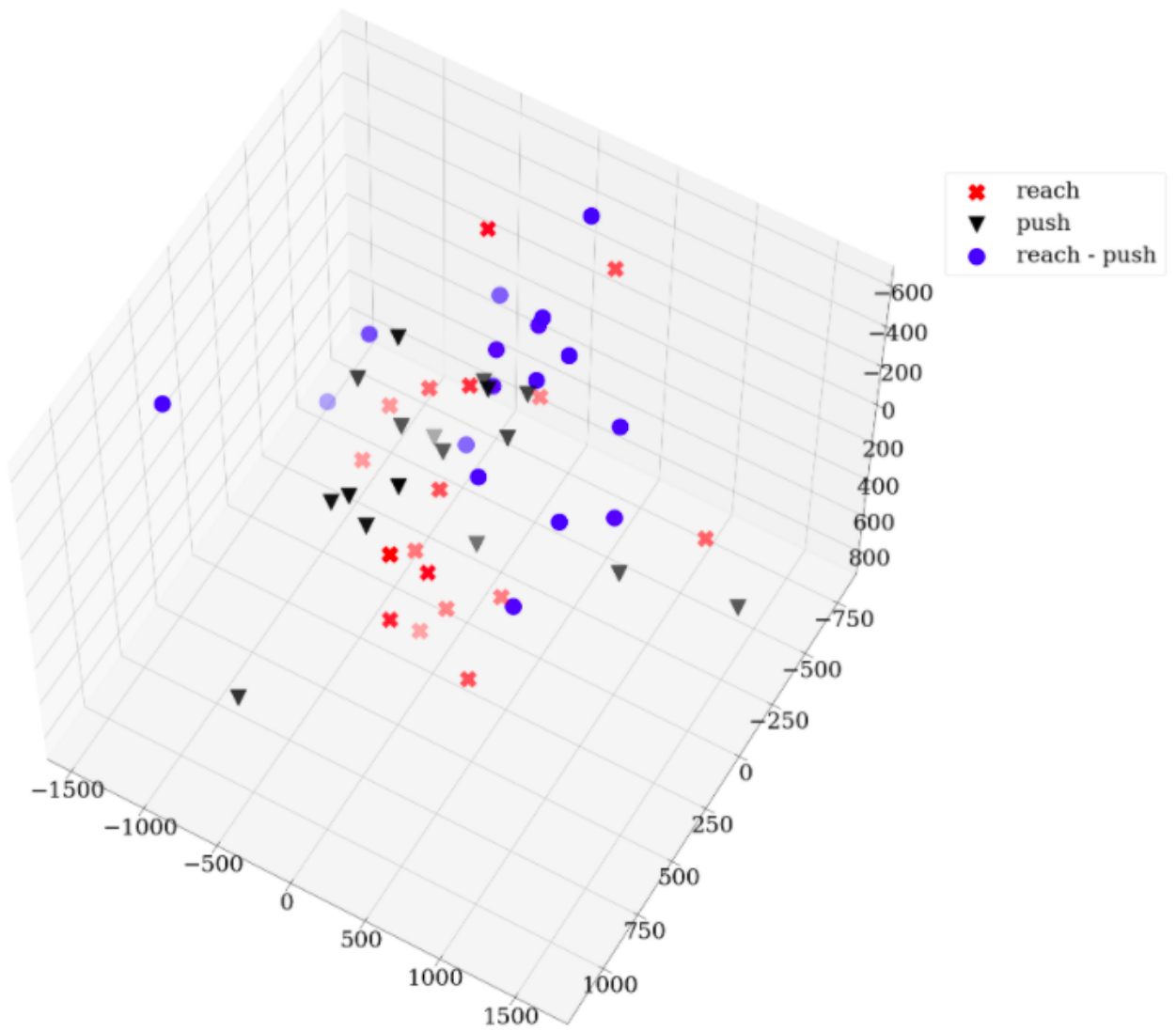
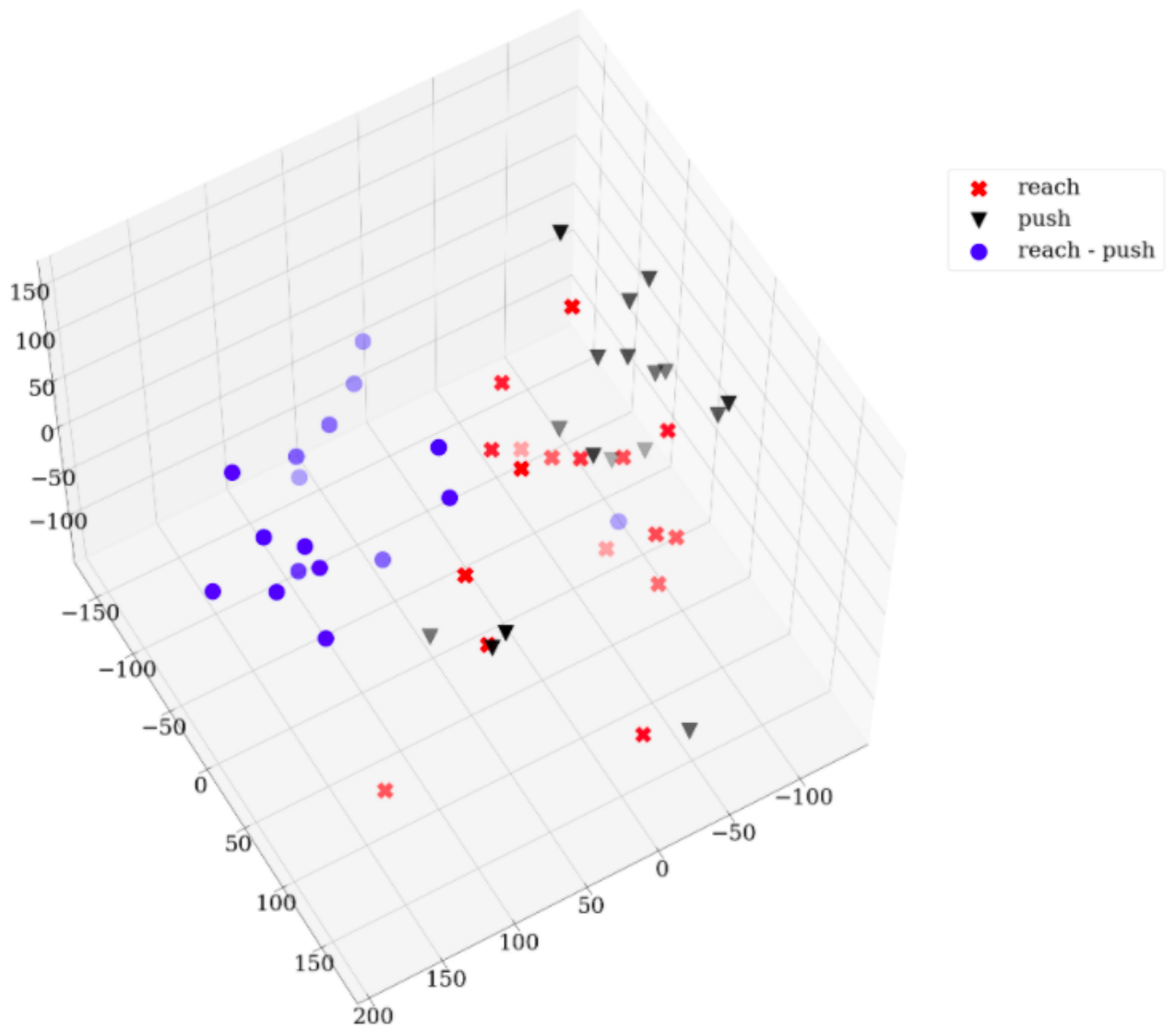


Figure 6.2: ARI scores when different subsets of features from pool5 and fc6 layers are used as the action vector. High ARI scores when only the top 20% of features (based on ANOVA F-values) are used, indicate the existence of a subset of features more significant than others in representing manipulation tasks. Tags at each point give the corresponding number of features selected.



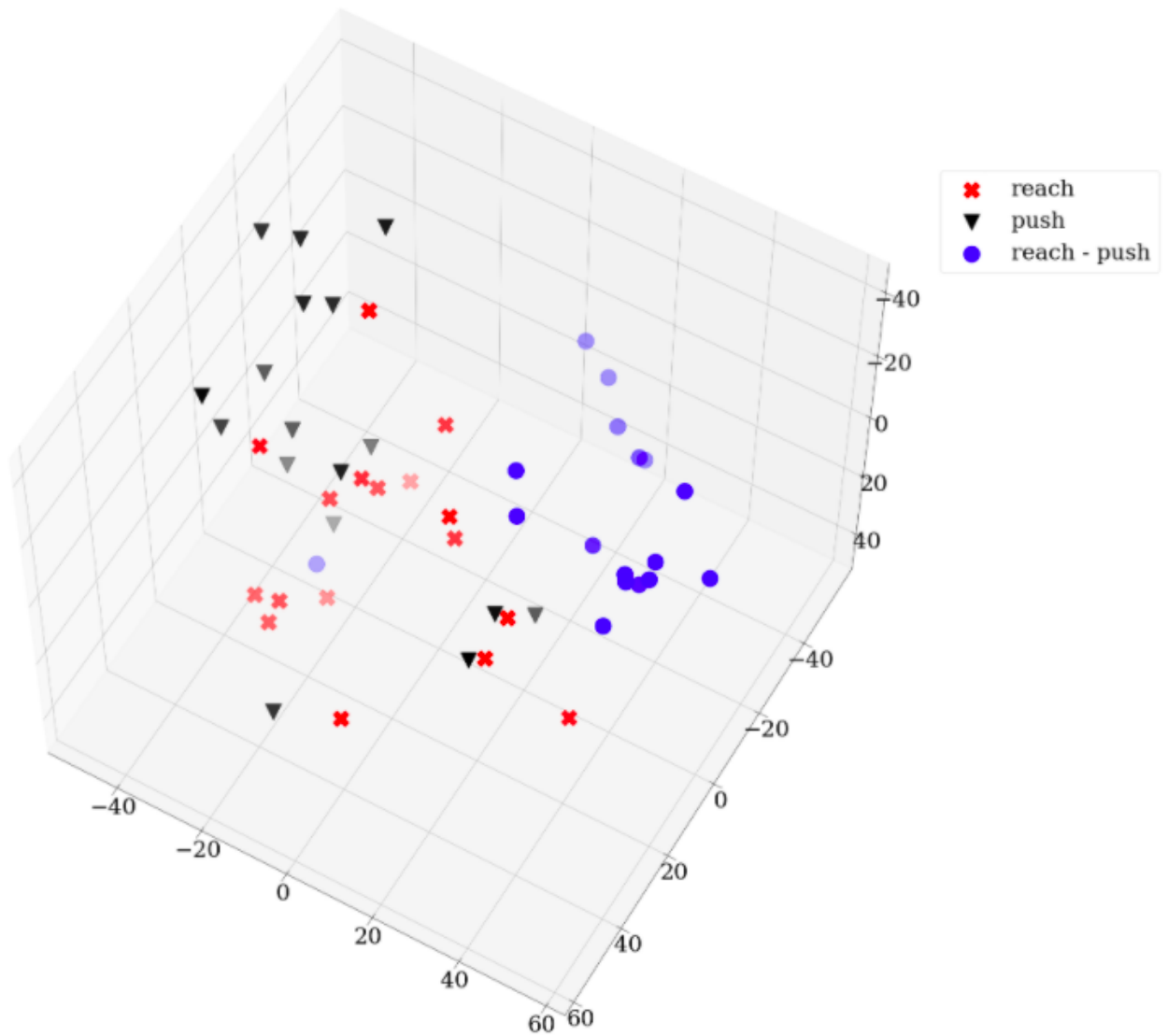
(a)

Figure 6.3: Visualising LMD using action vectors for Baseline-R (features from pool5 layer of NN:UCF101 with randomly initialised weights)



(a)

Figure 6.4: Visualising LMD using action vectors from pool5 layer of NN:UCF101



(a)

Figure 6.5: Visualising LMD using action vectors from fc6 layer of NN:UCF101

6.5 Clustering analysis after PCA

We performed PCA dimensionality reduction on features from different layers of NN:UCF101, before being used as the action vectors. We then conduct clustering analysis on LMD using this dimensionality reduced action vectors. The number of components are automatically selected in each case such that the amount of variance that needs to be explained is greater than 70%. The results are tabulated in Table 6.3. The results show that the features from layers pool5 and fc6 are still best suited to be used as the action vectors.

Table 6.3: ARI scores after applying PCA dimensionality reduction on features from different layers of NN:UCF101

Layer	ARI score
Baseline-R	0.01
pool5	0.35
fc6	0.33
fc7	0.29
fc8	0.18

6.6 Simulation experiment environment designs

6.6.1 Reach

```
<mujoco model="arm3d">
  <compiler inertiafromgeom="true" angle="radian" coordinate="local" />
  <option timestep="0.01" gravity="0 0 0" iterations="20" integrator="Euler" />

  <default>
    <joint armature="0.04" damping="1" limited="true"/>
    <geom friction=".8 .1 .1" density="300" margin="0.002" condim="1" contype="1" conaffinity="1"/>
  </default>

  <worldbody>
    <light diffuse=".9 .9 .9" pos="0 0 5" dir="0 0 -1"/>
    <!--<geom rgba="0.16470588 0.60392157 0.5372549 1.0" type="plane" pos="0 0.5 -0.15" size="2 4 0.1" contype="1" conaffinity="1"/>-->
    <geom rgba="1.0 1.0 1.0 1.0" type="plane" pos="0 0.5 -0.15" size="4 4 0.1" contype="1" conaffinity="1"/>

    <body name="palm" pos="1 2.1 0">
      <geom rgba="0.14117647 0.17647059 0.0 0.0 1.0" type="capsule" fromto="0 0 -0.1 0 0 0.1" size="0.12"/>
      <body name="proximal_1" pos="0 0 -0.075" axisangle="0 0 1 0.785">
        <joint name="proximal_j_1" type="hinge" pos="0 0 0" axis="0 0 1" range="1.5 3.0" damping="1.0" />
        <geom rgba="0.0 0.0 1.0" type="capsule" fromto="0 0 0 1.0 0 0" size="0.03" contype="1" conaffinity="1"/>
      </body>
    </body>
  </worldbody>
</mujoco>
```

6.6. Simulation experiment environment designs

```
<body name="distal_1" pos="1.0 0 0" axisangle="0 0 1 -0.785">
  <joint name="distal_j_1" type="hinge" pos="0 0 0" axis="0 0 1" range="1.5 3.5" damping="1.0"/>
  <geom rgba="0. 0. 0. 1." type="capsule" fromto="0 0 0 1.1 0 0" size="0.03" contype="1" conaffinity="1"/>
  <body name="distal_2" pos="1.1 0 0" axisangle="0 0 1 -1.57">
    <joint name="distal_j_2" type="hinge" pos="0 0 0" axis="0 0 1" range="0.5 2.0" damping="1.0"/>
    <geom rgba="0. 0. 0. 1." type="capsule" fromto="0 0 0 1.2 0 0" size="0.03" contype="1" conaffinity="1"/>
    <body name="distal_4" pos="1.2 0 0">
      <site name="tip arml" pos="0.1 -0.1 0" size="0.01" />
      <site name="tip armr" pos="0.1 0.1 0" size="0.01" />
      <!--<joint name="distal_j_3" type="hinge" pos="0 0 0" axis="1 0 0" range="-3.3213 3.3" damping="0.5"/>-->
      <geom rgba="0. 0. 0. 1." type="capsule" fromto="0 0 0 0.15 0.15 0" size="0.03" contype="1" conaffinity="1" />
      <geom rgba="0. 0. 0. 1." type="capsule" fromto="0.15 0.15 0 0.3 0.15 0" size="0.03" contype="1" conaffinity="1" />
      <geom rgba="0. 0. 0. 1." type="capsule" fromto="0 0 0 0.15 -0.15 0" size="0.03" contype="1" conaffinity="1" />
      <geom rgba="0. 0. 0. 1." type="capsule" fromto="0.15 -0.15 0 0.3 -0.15 0" size="0.03" contype="1" conaffinity="1" />
    </body>
  </body>
</body>
</body>
</body>
</body>

<body name="object" pos="0 3 0"> <!-- Object propoerties --->
  <!--<geom rgba="1. 1. 1. 1" type="box" size="0.05 0.05 0.05" density='0.00001' contype="1" conaffinity="1"/>-->
  <geom rgba="1 0 0 1" type="cylinder" size="0.1 0.0001 -0.0001" density="0.00001" contype="1" conaffinity="1"/>
  <joint name="obj_slidey" type="slide" pos="0.025 0.025 0.025" axis="0 1 0" range="-10.3213 10.3" damping="0.5"/>
  <joint name="obj_slidex" type="slide" pos="0.025 0.025 0.025" axis="1 0 0" range="-10.3213 10.3" damping="0.5"/>
</body>

<!-- Background Object propoerties -->

<body name="bobject0" pos="1.2 -0.9 0">
  <geom rgba="0 1 0 1" type="box" size="0.1 0.1 .1" density='0.00001' contype="1" conaffinity="1"/>
  <joint name="obj_slidey_bg0" type="slide" pos="0.025 0.025 0.025" axis="0 1 0" range="-10.3213 10.3" damping="0.5"/>
  <joint name="obj_slidex_bg0" type="slide" pos="0.025 0.025 0.025" axis="1 0 0" range="-10.3213 10.3" damping="0.5"/>
</body>

<body name="bobject1" pos="1.9 -0.6 0">
  <geom rgba="0 0 0 1" type="ellipsoid" size="0.1 0.1 .1" density='0.00001' contype="1" conaffinity="1"/>
  <joint name="obj_slidey_bg1" type="slide" pos="0.025 0.025 0.025" axis="0 1 0" range="-10.3213 10.3" damping="0.5"/>
  <joint name="obj_slidex_bg1" type="slide" pos="0.025 0.025 0.025" axis="1 0 0" range="-10.3213 10.3" damping="0.5"/>
</body>

<!-- Background Object propoerties --->

<body name="goal" pos="0 -1.0 -0.145">
  <!--<body name="goal" pos="0.0 0.0 -0.1"> rgba (brown)="0.4 .26 0.13 1"-->
  <!--<geom rgba="0.4 .26 0.13 1" type="box" size="0.1 0.1 0.1" density="0.00001" contype="0" conaffinity="0"/>-->
  <geom rgba="0 0 1 1" type="box" size="0.2 0.2 0.01" density="0.00001" contype="0" conaffinity="0"/>
  <joint name="goal_slidey" type="slide" pos="0 0 0" axis="0 1 0" range="-10.3213 10.3" damping="0.5"/>
  <joint name="goal_slidex" type="slide" pos="0 0 0" axis="1 0 0" range="-10.3213 10.3" damping="0.5"/>
</body>

</worldbody>

<actuator>
  <motor joint="proximal_j_1" ctrlrange="-3 3" ctrllimited="true"/>
  <motor joint="distal_j_1" ctrlrange="-3 3" ctrllimited="true"/>
  <motor joint="distal_j_2" ctrlrange="-3 3" ctrllimited="true"/>
  <!--<motor joint="distal_j_3" ctrlrange="-3 3" ctrllimited="true"/>-->
</actuator>
</mujoco>
```

6.6.2 Push

```
<mujoco model="arm3d">
  <compiler inertiafromgeom="true" angle="radian" coordinate="local" />
```

6.6. Simulation experiment environment designs

```
<option timestep="0.01" gravity="0 0 0" iterations="20" integrator="Euler" />

<default>
  <joint armature="0.04" damping="1" limited="true"/>
  <geom friction=".8 .1 .1" density="300" margin="0.002" condim="1" contype="1" conaffinity="1"/>
</default>

<worldbody>
  <light diffuse=".9 .9 .9" pos="0 0 5" dir="0 0 -1"/>
  <!--<geom rgba="0.16470588 0.60392157 0.5372549 1.0" type="plane" pos="0 0 5 -0.15" size="2 4 0.1" contype="1" conaffinity="1"/>-->
  <geom rgba="1.0 1.0 1.0 1.0" type="plane" pos="0 0 5 -0.15" size="4 4 0.1" contype="1" conaffinity="1"/>

  <body name="palm" pos="1 2.1 0">
    <geom rgba="0.14117647 0.17647059 0.0 0.0 1." type="capsule" fromto="0 0 -0.1 0 0 0.1" size="0.12"/>
    <body name="proximal_1" pos="0 0 -0.075" axisangle="0 0 1 0.785">
      <joint name="proximal_j_1" type="hinge" pos="0 0 0" axis="0 0 1" range="1.5 3.0" damping="1.0"/>
      <geom rgba="0.0 0.0 1." type="capsule" fromto="0 0 0 1.0 0 0" size="0.03" contype="1" conaffinity="1"/>
      <body name="distal_1" pos="1.0 0 0" axisangle="0 0 1 -0.785">
        <joint name="distal_j_1" type="hinge" pos="0 0 0" axis="0 0 1" range="1.5 3.5" damping="1.0"/>
        <geom rgba="0.0 0.0 1." type="capsule" fromto="0 0 0 1.1 0 0" size="0.03" contype="1" conaffinity="1"/>
        <body name="distal_2" pos="1.1 0 0" axisangle="0 0 1 -1.57">
          <joint name="distal_j_2" type="hinge" pos="0 0 0" axis="0 0 1" range="0.5 2.0" damping="1.0"/>
          <geom rgba="0.0 0.0 1." type="capsule" fromto="0 0 0 1.2 0 0" size="0.03" contype="1" conaffinity="1"/>
          <body name="distal_4" pos="1.2 0 0">
            <site name="tip arml" pos="0.1 -0.1 0" size="0.01" />
            <site name="tip armr" pos="0.1 0.1 0" size="0.01" />
            <!--<joint name="distal_j_3" type="hinge" pos="0 0 0" axis="1 0 0" range="-3.3213 3.3" damping="0.5"/>-->
            <geom rgba="0.0 0.0 1." type="capsule" fromto="0 0 0 0.15 0 0.15 0" size="0.03" contype="1" conaffinity="1" />
            <geom rgba="0.0 0.0 1." type="capsule" fromto="0.15 0.15 0 0.3 0.15 0" size="0.03" contype="1" conaffinity="1" />
            <geom rgba="0.0 0.0 1." type="capsule" fromto="0 0 0 0.15 -0.15 0" size="0.03" contype="1" conaffinity="1" />
            <geom rgba="0.0 0.0 1." type="capsule" fromto="0.15 -0.15 0 0.3 -0.15 0" size="0.03" contype="1" conaffinity="1" />
          </body>
        </body>
      </body>
    </body>
  </body>

  <body name="object" pos="0.0 0.3 0"> <!-- Object propoerties --->
    <!--<geom rgba="1. 1. 1. 1" type="box" size="0.05 0.05 0.05" density="0.00001" contype="1" conaffinity="1"/>-->
    <geom rgba="1 0 0 1" type="cylinder" size="0.1 0.1 1.4" density="0.00001" contype="1" conaffinity="1"/>
    <joint name="obj_slidey" type="slide" pos="0.025 0.025 0.025" axis="0 1 0" range="-10.3213 10.3" damping="0.5"/>
    <joint name="obj_slidex" type="slide" pos="0.025 0.025 0.025" axis="1 0 0" range="-10.3213 10.3" damping="0.5"/>
  </body>

  <!-- Background Object propoerties -->

  <body name="bobject0" pos="1.2 -0.9 0">
    <geom rgba="0 1 0 1" type="box" size="0.1 0.1 1" density="0.00001" contype="1" conaffinity="1"/>
    <joint name="obj_slidey_bg0" type="slide" pos="0.025 0.025 0.025" axis="0 1 0" range="-10.3213 10.3" damping="0.5"/>
    <joint name="obj_slidex_bg0" type="slide" pos="0.025 0.025 0.025" axis="1 0 0" range="-10.3213 10.3" damping="0.5"/>
  </body>

  <body name="bobject1" pos="1.9 -0.6 0">
    <geom rgba="0 0 0 1" type="ellipsoid" size="0.1 0.1 1" density="0.00001" contype="1" conaffinity="1"/>
    <joint name="obj_slidey_bg1" type="slide" pos="0.025 0.025 0.025" axis="0 1 0" range="-10.3213 10.3" damping="0.5"/>
    <joint name="obj_slidex_bg1" type="slide" pos="0.025 0.025 0.025" axis="1 0 0" range="-10.3213 10.3" damping="0.5"/>
  </body>

  <!-- Background Object propoerties --->

  <body name="goal" pos="0 -1.0 -0.145">
    <!--<body name="goal" pos="0 0 0 -0.1">-->
    <!--<geom rgba="1. 0. 0. 1" type="box" size="0.1 0.1 0.1" density="0.00001" contype="0" conaffinity="0"/>-->
    <geom rgba="0.0 0. 1. 1" type="box" size="0.2 0.2 0.01" density="0.00001" contype="0" conaffinity="0"/>
    <joint name="goal_slidey" type="slide" pos="0 0 0" axis="0 1 0" range="-10.3213 10.3" damping="0.5"/>
    <joint name="goal_slidex" type="slide" pos="0 0 0" axis="1 0 0" range="-10.3213 10.3" damping="0.5"/>
  </body>
```

```

</body>
</worldbody>
<actuator>
  <motor joint="proximal_j_1" ctrlrange="-3 3" ctrlimited="true"/>
  <motor joint="distal_j_1" ctrlrange="-3 3" ctrlimited="true"/>
  <motor joint="distal_j_2" ctrlrange="-3 3" ctrlimited="true"/>
  <!--<motor joint="distal_j_3" ctrlrange="-3 3" ctrlimited="true"/>-->
</actuator>
</mujoco>

```

6.7 DDPG RL algorithm

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ .

Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$

Initialize replay buffer R

for episode = 1, M **do**

 Initialize a random process \mathcal{N} for action exploration

 Receive initial observation state s_1

for $t = 1, T$ **do**

 Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise

 Execute action a_t and observe reward r_t and observe new state s_{t+1}

 Store transition (s_t, a_t, r_t, s_{t+1}) in R

 Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R

 Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$

 Update critic by minimizing the loss: $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$

 Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

 Update the target networks:

$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$

end for
end for

Figure 6.6: DDPG RL algorithm

6.8 MIME dataset task categories

The 20 task categories in MIME dataset are: 1. Pour 2. Stir 3. Pass 4. Stack 5. Place objects in box 6. Open Bottles 7. Push 8. Rotate 9. Wipe 10. Press Buttons 11. Close Book 12. Picking (single hand) 13. Picking (both hands) 14. Poke 15. Pull (two hands) 16. Push (two hands) 17. Toy Car Trajectories 18. Roll 19. Drop Objects 20. Pull (Single hand).

"If We Knew What it Was We Were Doing, it Would Not be Called Research, Would it?"

– Albert Einstein

Bibliography

- [1] T. Zhang, Z. McCarthy, O. Jowl, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5628–5635, 2018. ([document](#)), 1, 1.1, 1
- [2] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007. ([document](#)), 1, 1.1, 1
- [3] V. Krüger, D. Herzog, S. Baby, A. Ude, and D. Kragic, "Learning actions from observations," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 30–43, 2010. ([document](#)), 2.2.1, 2.3, 2.3.6
- [4] A. Gupta, C. Eppner, S. Levine, and P. Abbeel, "Learning dexterous manipulation for a soft robotic hand from human demonstrations," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-Novem, pp. 3786–3793, 2016. ([document](#)), 2.2.1, 2.4
- [5] D. C. Bentivegna, A. Ude, C. G. Atkeson, and G. Cheng, "Humanoid robot learning and game playing using PC-based vision," *IEEE International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2449–2454, 2002. ([document](#)), 2.2.1, 2.5

- [6] A. D. Dragan and S. S. Srinivasa, "Online customization of teleoperation interfaces," *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, pp. 919–924, 2012. ([document](#)), [2.2.1](#), [2.6](#)
- [7] 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)*, pp. 1134–1141, IEEE, 2018. ([document](#)), [2.3.1](#), [2.7](#), [2.6](#)
- [8] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018. ([document](#)), [2.8](#)
- [9] L. Smith, N. Dhawan, M. Zhang, P. Abbeel, and S. Levine, "AVID: Learning Multi-Stage Tasks via Pixel-Level Translation of Human Videos," in *Proceedings of Robotics: Science and Systems*, (Corvallis, Oregon, USA), July 2020. ([document](#)), [2.9](#), [2.3.3](#), [2.3.3](#), [2.4.3](#), [2.2](#), [2.3](#)
- [10] P. Sharma, D. Pathak, and A. Gupta, "Third-person visual imitation learning via decoupled hierarchical controller," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019. ([document](#)), [1](#), [2.3.3](#), [2.3.3](#), [2.10](#), [2.6](#), [2.2](#), [2.3](#)
- [11] J. Jin, L. Petrich, Z. Zhang, M. Dehghan, and M. Jagersand, "Visual geometric skill inference by watching human demonstration," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8985–8991, IEEE, 2020. ([document](#)), [2.11](#), [2.3.5](#)
- [12] Z. Jia, M. Lin, Z. Chen, and S. Jian, "Vision-based robot manipulation learning via human demonstrations," *ArXiv*, vol. abs/2003.00385, 2020. ([document](#)), [2.12](#), [2.3.6](#), [2.4.4](#), [2.2](#), [2.3](#)
- [13] A. Nguyen, D. Kanoulas, L. Muratore, D. G. Caldwell, and N. G. Tsagarakis, "Translating Videos to Commands for Robotic Manipulation with Deep Recurrent Neural Networks," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3782–3788, 2018. ([document](#)), [2.13](#), [2.3.8](#), [2.14](#), [2.4.4](#), [2.6](#)

- [14] M. Arnold, R. R. Negenborn, G. Andersson, and B. De Schutter, "Multi-area predictive control for combined electricity and natural gas systems," in *2009 European Control Conference (ECC)*, pp. 1408–1413, 2009. ([document](#)), [2.15](#)
- [15] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*, pp. 1126–1135, PMLR, 2017. ([document](#)), [2.4.6](#), [2.16](#)
- [16] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012. ([document](#)), [3.3.1](#), [3.1](#), [3.2](#)
- [17] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 4489–4497, 2015. ([document](#)), [3.1](#), [3.3.2](#), [3.3.2](#), [3.3](#)
- [18] P. Sharma, L. Mohan, L. Pinto, and A. Gupta, "Multiple interactions made easy (MIME): large scale demonstrations data for imitation," in *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, vol. 87 of *Proceedings of Machine Learning Research*, pp. 906–915, PMLR, 2018. ([document](#)), [4.8](#)
- [19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015. ([document](#)), [3.3.2](#), [4.3.1](#), [4.3.3](#), [4.6](#)
- [20] J. Forlizzi and C. DiSalvo, "Service robots in the domestic environment: a study of the roomba vacuum in the home," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pp. 258–265, ACM, 2006. [1](#)
- [21] J. Leven, D. Burschka, R. Kumar, G. Zhang, S. Blumenkranz, X. D. Dai, M. Awad, G. D. Hager, M. Marohn, M. Choti, *et al.*, "Davinci canvas: a telerobotic surgical system with integrated, robot-assisted, laparoscopic ultrasound capability," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 811–818, Springer, 2005. [1](#)

- [22] L. Pauly, M. Baiju, P. Viswanathan, P. Jose, D. Paul, and D. Sankar, "Cambot: Customer assistance mobile manipulator robot," in *2015 IEEE Bombay Section Symposium (IBSS)*, pp. 1–4, IEEE, 2015. [1](#)
- [23] A. Okano, H. Matsubara, and H. Inoue, "Design and implementation of a task-oriented robot language," *Advanced Robotics*, vol. 3, no. 3, pp. 177–191, 1988. [1](#)
- [24] T. Lozano-Perez and P. H. Winston, "Lama: A language for automatic mechanical assembly," in *IJCAI*, 1977. [1](#)
- [25] S. Schaal, "Is imitation learning the route to humanoid robots?," *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999. [1](#)
- [26] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Survey: Robot programming by demonstration," *Springer Handbook of Robotics*, pp. 1371–1394, 2008. [1](#)
- [27] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009. [1](#), [2.1](#)
- [28] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013. [1](#)
- [29] L. Tai, J. Zhang, M. Liu, J. Boedecker, and W. Burgard, "A survey of deep network solutions for learning control in robotics: From reinforcement to imitation," *arXiv preprint arXiv:1612.07139*, 2016. [1](#)
- [30] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "An Algorithmic Perspective on Imitation Learning," *Foundations and Trends in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018. [1](#)
- [31] F. Torabi, G. Warnell, and P. Stone, "Recent advances in imitation learning from observation," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2019-Augus, no. August, pp. 6325–6331, 2019. [1](#), [1](#), [2.1](#)

- [32] L. Pauly, “Defining the problem of observation learning,” *arXiv preprint arXiv:1808.08288*, 2018. [1](#), [2.1](#)
- [33] D. Borsa, N. Heess, B. Piot, S. Liu, L. Hasenclever, R. Munos, and O. Pietquin, “Observational learning by reinforcement learning,” *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, vol. 2, pp. 1117–1124, 2019. [1](#), [2.1](#)
- [34] P. Bakker and Y. Kuniyoshi, “Robot see, robot do: An overview of robot imitation,” in *AISB96 Workshop on Learning in Robots and Animals*, pp. 3–11, 1996. [1](#)
- [35] Y. Kuniyoshi, M. Inaba, and H. Inoue, “Learning by Watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance,” *IEEE Transactions on Robotics and Automation*, vol. 10, no. 6, pp. 799–822, 1994. [1](#), [1](#)
- [36] T. Suehiro, “Toward an Assembly Plan from Observation Part I: Task Recognition with Polyhedral Objects,” *IEEE Transactions on Robotics and Automation*, vol. 10, no. 3, pp. 368–385, 1994. [1](#), [1](#)
- [37] T. Hamabe, H. Goto, and J. Miura, “A programming by demonstration system for human-robot collaborative assembly tasks,” in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1195–1201, IEEE, 2015. [1](#), [2.2.1](#)
- [38] J. Jin, L. Petrich, M. Dehghan, and M. Jagersand, “A geometric perspective on visual imitation learning,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5194–5200, 2020. [1](#), [2.3.5](#)
- [39] B. C. Stadie, P. Abbeel, and I. Sutskever, “Third person imitation learning,” in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. [1](#), [2.2.2](#), [2.3.2](#), [2.4.5](#), [2.2](#), [2.3](#)
- [40] Y. Liu, A. Gupta, P. Abbeel, and S. Levine, “Imitation from Observation: Learning to Imitate Behaviors from Raw Video via Context Transla-

- tion," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1118–1125, 2018. [1](#), [1.2](#), [2.3.3](#), [2.3.3](#), [2.4.1](#), [2.5](#), [2.6](#), [2.2](#), [2.3](#)
- [41] R. Okumura, M. Okada, and T. Taniguchi, "Domain-adversarial and -conditional state space model for imitation learning," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5179–5186, 2020. [1](#)
- [42] A. Bandura and R. H. Walters, *Social learning theory*, vol. 1. Prentice-hall Englewood Cliffs, NJ, 1977. [1.1](#)
- [43] A. Bandura, *Social foundations of thought and action : a social cognitive theory*. Prentice-Hall, 1986. [1.1](#)
- [44] A. Bandura and R. H. Walters, *Social learning and personality development*. New York, 1963. [1.1](#)
- [45] C. L. Nehaniv, K. Dautenhahn, and K. Dautenhahn, *Imitation in animals and artifacts*. MIT press, 2002. [1.1](#)
- [46] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016. [1.2](#), [2.2.2](#)
- [47] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. [1.2](#), [2.2.2](#)
- [48] L. Pauly, H. Peel, S. Luo, D. Hogg, and R. Fuentes, "Deeper networks for pavement crack detection," in *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, vol. 34, 2017. [1.2](#)
- [49] L. Pauly, R. D. Raj, and B. Paul, "Hand written digit recognition system for south indian languages using artificial neural networks," in *2015 Eighth International Conference on Contemporary Computing (IC3)*, pp. 122–126, IEEE, 2015. [1.2](#)
- [50] L. Pauly and D. Sankar, "Non intrusive eye blink detection from low resolution images using hog-svm classifier," *International Journal of Image, Graphics and Signal Processing*, vol. 8, no. 10, p. 11, 2016. [1.2](#)

- [51] L. Pauly and D. Sankar, "A new method for sorting and grading of mangoes based on computer vision system," in *2015 IEEE International Advance Computing Conference (IACC)*, pp. 1191–1195, IEEE, 2015. [1.2](#)
- [52] L. Pauly and D. Sankar, "A novel method for eye tracking and blink detection in video frames," in *2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS)*, pp. 252–257, IEEE, 2015. [1.2](#)
- [53] Y. Liu, A. Gupta, P. Abbeel, and S. Levine, "Imitation from observation: Learning to imitate behaviors from raw video via context translation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1118–1125, IEEE, 2018. [1.2](#)
- [54] P. Sermanet, K. Xu, and S. Levine, "Unsupervised perceptual rewards for imitation learning," *Proceedings of Robotics: Science and Systems (RSS)*, 2017. [1.2](#), [2.3.4](#), [2.4.1](#), [2.4.1](#), [2.5](#), [2.2](#), [2.3](#), [4.4.1](#)
- [55] 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)*, pp. 1134–1141, IEEE, 2018. [1.2](#), [2.4.1](#), [2.4.1](#), [2.5](#), [2.2](#), [2.3](#)
- [56] Y. Lee, E. S. Hu, Z. Yang, and J. J. Lim, "To follow or not to follow: Selective imitation learning from observations," in *Proceedings of the Conference on Robot Learning* (L. P. Kaelbling, D. Kragic, and K. Sugiura, eds.), vol. 100 of *Proceedings of Machine Learning Research*, pp. 11–23, PMLR, 30 Oct–01 Nov 2020. [1.2](#)
- [57] L. Pauly, W. C. Agboh, D. C. Hogg, and R. Fuentes, "O2A: One-shot Observational learning with Action vectors," *arXiv e-prints*, p. arXiv:1810.07483, Dec. 2020. [1.5](#)
- [58] C. Do, C. Gordillo, and W. Burgard, "Learning to pour using deep deterministic policy gradients," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3074–3079, IEEE, 2018. [2.1](#)

- [59] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning.,” in *ICLR (Poster)*, 2016. [2.1](#)
- [60] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine, “Collective robot reinforcement learning with distributed asynchronous guided policy search,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 79–86, IEEE, 2017. [2.1](#)
- [61] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, “Learning dexterous in-hand manipulation,” *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020. [2.1](#)
- [62] H. van Hoof, T. Hermans, G. Neumann, and J. Peters, “Learning robot in-hand manipulation with tactile features,” in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 121–127, 2015. [2.1](#)
- [63] A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar, “Deep dynamics models for learning dexterous manipulation,” in *Conference on Robot Learning (CoRL)*, 2019. [2.1](#)
- [64] A. Singh, L. Yang, C. Finn, and S. Levine, “End-to-end robotic reinforcement learning without reward engineering.,” in *Robotics: Science and Systems*, 2019. [2.1](#)
- [65] C. Daniel, M. Viering, J. Metz, O. Kroemer, and J. Peters, “Active reward learning.,” in *Robotics: Science and systems*, 2014. [2.1](#)
- [66] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences,” in *Advances in Neural Information Processing Systems*, pp. 4299–4307, 2017. [2.1](#)
- [67] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, “Deep imitation learning for complex manipulation tasks from virtual reality teleoperation,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, IEEE, 2018. [2.1](#)

- [68] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*, pp. 763–768, IEEE, 2009. [2.1](#)
- [69] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, pp. 4565–4573, 2016. [2.1](#)
- [70] N. Ratliff, J. A. Bagnell, and S. S. Srinivasa, "Imitation learning for locomotion and manipulation," in *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pp. 392–397, IEEE, 2007. [2.1](#)
- [71] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3758–3765, IEEE, 2018. [2.1](#)
- [72] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *in Proc. 17th International Conf. on Machine Learning*, Citeseer, 2000. [2.1](#)
- [73] M. Field, D. Stirling, F. Naghdy, and Z. Pan, "Motion capture in robotics review," *2009 IEEE International Conference on Control and Automation, ICCA 2009*, pp. 1697–1702, 2009. [2.2.1](#)
- [74] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Trajectory formation for imitation with nonlinear dynamical systems," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2, pp. 752–757, 2001. [2.2.1](#)
- [75] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2, no. May, pp. 1398–1403, 2002. [2.2.1](#)
- [76] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 109–116, 2004. [2.2.1](#)

- [77] Y. Yang, Y. Li, C. Fermüller, and Y. Aloimonos, "Robot learning manipulation action plans by "watching" unconstrained videos from the World Wide Web," *Proceedings of the National Conference on Artificial Intelligence*, vol. 5, pp. 3686–3692, 2015. [2.2.1](#)
- [78] Y. Demiris and B. Khadhour, "Hierarchical attentive multiple models for execution and recognition of actions," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 361–369, 2006. [2.2.1](#)
- [79] K. Lee, Y. Su, T. K. Kim, and Y. Demiris, "A syntactic approach to robot imitation learning using probabilistic activity grammars," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1323–1334, 2013. [2.2.1](#)
- [80] K. Ramirez-Amaro, M. Beetz, and G. Cheng, "Transferring skills to humanoid robots by extracting semantic representations from observations of human activities," *Artificial Intelligence*, vol. 247, pp. 95–118, 2017. [2.2.1](#)
- [81] M. Sieb, Z. Xian, A. Huang, O. Kroemer, and K. Fragkiadaki, "Graph-structured visual imitation," in *Conference on Robot Learning*, pp. 979–989, PMLR, 2020. [2.2.1](#)
- [82] H. Zhang, P. Lai, S. Paul, S. Kothawade, and S. Nikolaidis, "Learning collaborative action plans from youtube videos," in *International Symposium on Robotics Research (ISRR)*, 2019. [2.2.1](#)
- [83] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *International journal of computer vision*, vol. 128, no. 2, pp. 261–318, 2020. [2.2.1](#)
- [84] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019. [2.2.1](#)
- [85] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *arXiv preprint arXiv:1905.05055*, 2019. [2.2.1](#)

- [86] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009. [2.2.1](#)
- [87] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann, "Imitation learning of dual-arm manipulation tasks in humanoid robots," *International Journal of Humanoid Robotics*, vol. 5, no. 02, pp. 183–202, 2008. [2.2.1](#)
- [88] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4929–4937, 2016. [2.2.1](#)
- [89] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7291–7299, 2017. [2.2.1](#)
- [90] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "Rmpe: Regional multi-person pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2334–2343, 2017. [2.2.1](#)
- [91] Bazarevsky, Valentin and Zhang, Fan, "On-device, real-time hand tracking with mediapipe." <https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html>, 2019. [Online; Accessed: 9-June-2020]. [2.2.1](#)
- [92] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by Watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 6, pp. 799–822, 1994. [2.2.1](#)
- [93] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017. [2.2.2](#)

- [94] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. [2.2.2](#)
- [95] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015. [2.2.2](#)
- [96] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *Workshop at International Conference on Learning Representations*, 2014. [2.2.2](#)
- [97] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014. [2.2.2](#)
- [98] M. Suresha, S. Kuppa, and D. Raghukumar, "A study on deep learning spatiotemporal models and feature extraction techniques for video understanding," *International Journal of Multimedia Information Retrieval*, pp. 1–21, 2020. [2.2.2](#)
- [99] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1735–1742, IEEE, 2006. [2.3.1](#)
- [100] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015. [2.3.1](#)
- [101] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4004–4012, 2016. [2.3.1](#)

- [102] K. Sohn, “Improved deep metric learning with multi-class n-pair loss objective,” in *Advances in neural information processing systems*, pp. 1857–1865, 2016. [2.3.1](#)
- [103] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin, “Deep metric learning with angular loss,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2593–2601, 2017. [2.3.1](#)
- [104] W. Kim, B. Goyal, K. Chawla, J. Lee, and K. Kwon, “Attention-based ensemble for deep metric learning,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 736–751, 2018. [2.3.1](#)
- [105] M. Boudiaf, J. Rony, I. M. Ziko, E. Granger, M. Pedersoli, P. Piantanida, and I. Ben Ayed, “A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses,” in *European Conference on Computer Vision*, pp. 548–564, Springer, 2020. [2.3.1](#)
- [106] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, pp. 2672–2680, 2014. [2.3.2](#)
- [107] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, “Deep domain confusion: Maximizing for domain invariance,” *arXiv preprint arXiv:1412.3474*, 2014. [2.3.2](#)
- [108] D. Charte, F. Charte, S. García, M. J. del Jesus, and F. Herrera, “A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines,” *Information Fusion*, vol. 44, pp. 78–96, 2018. [2.3.3](#), [2.3.3](#)
- [109] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017. [2.3.3](#)
- [110] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical im-*

- age computing and computer-assisted intervention*, pp. 234–241, Springer, 2015. [2.3.3](#)
- [111] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009. [2.3.4](#)
- [112] A. Sax, J. O. Zhang, B. Emi, A. Zamir, S. Savarese, L. Guibas, and J. Malik, “Learning to navigate using mid-level visual priors,” in *Conference on Robot Learning*, pp. 791–812, PMLR, 2020. [2.3.4](#), [2.4.2](#)
- [113] B. Zhou, P. Krähenbühl, and V. Koltun, “Does computer vision matter for action?,” *Science Robotics*, vol. 4, no. 30, p. 6661, 2019. [2.3.4](#)
- [114] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016. [2.3.4](#)
- [115] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, IEEE, 2009. [2.3.4](#)
- [116] X. Wang, R. Girshick, A. Gupta, and K. He, “Non-local neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7794–7803, 2018. [2.3.4](#)
- [117] P. Sharma, L. Mohan, L. Pinto, and A. Gupta, “Multiple interactions made easy (mime): Large scale demonstrations data for imitation,” in *Conference on Robot Learning*, pp. 906–915, PMLR, 2018. [2.3.4](#), [2.4.2](#), [2.5](#), [2.2](#), [2.3](#), [4.3.5](#), [4.6](#)
- [118] R. A. Gammons, “Eskimo: An expert system for kodak injection molding operations,” in *Artificial Intelligence in Engineering Design*, pp. 81–104, Elsevier, 1992. [2.3.5](#)
- [119] Wikipedia contributors, “Geometric primitives: Wikipedia, the free encyclopedia.” https://en.wikipedia.org/w/index.php?title=Geometric_primitive&oldid=785463009, 2017. [Online; Accessed: 9-June-2020]. [2.3.5](#)

- [120] BuildingSMART International Limited, "Industry foundation classes." https://standards.buildingsmart.org/IFC/DEV/IFC4_2/FINAL/HTML/link/constraint-association.htm, 2016. [Online; Accessed: 9-June-2020]. 2.3.5
- [121] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019. 2.3.6
- [122] R. Goroshin, M. F. Mathieu, and Y. LeCun, "Learning to linearize under uncertainty," in *Advances in Neural Information Processing Systems*, pp. 1234–1242, 2015. 2.3.7
- [123] W. R. Softky, "Unsupervised pixel-prediction," in *Advances in neural information processing Systems*, pp. 809–815, 1996. 2.3.7
- [124] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *International conference on machine learning*, pp. 843–852, 2015. 2.3.7
- [125] A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. 2.3.7
- [126] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in *Advances in neural information processing systems*, pp. 64–72, 2016. 2.3.7, 4.6
- [127] A. W. Tow, N. Sünderhauf, S. Shirazi, M. Milford, and J. Leitner, "What would you do? acting by learning to predict," in *International Conference on Intelligent Robots and Systems*, vol. 24, p. 28. 2.3.7, 2.2, 2.3
- [128] X. Wang, W. Chen, J. Wu, Y.-F. Wang, and W. Yang Wang, "Video captioning via hierarchical reinforcement learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4213–4222, 2018. 2.3.8

- [129] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2625–2634, 2015. [2.3.8](#), [3.3.2](#)
- [130] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation," in *Twenty-fifth AAAI conference on artificial intelligence*, 2011. [2.3.8](#)
- [131] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, "Sequence to sequence - video to text," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. [2.3.8](#)
- [132] H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu, "Video paragraph captioning using hierarchical recurrent neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4584–4593, 2016. [2.3.8](#)
- [133] V. Ramanishka, A. Das, J. Zhang, and K. Saenko, "Top-down visual saliency guided by captions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7206–7215, 2017. [2.3.8](#)
- [134] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014. [2.3.8](#)
- [135] H. Kuehne, A. Arslan, and T. Serre, "The language of actions: Recovering the syntax and semantics of goal-directed human activities," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 780–787, 2014. [2.3.8](#)
- [136] S. Yang, W. Zhang, W. Lu, H. Wang, and Y. Li, "Learning Actions from Human Demonstration Video for Robotic Manipulation," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1805–1811, 2019. [2.3.8](#), [2.4.4](#), [2.2](#), [2.3](#)

- [137] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, p. 1, ACM, 2004. [2.4.1](#)
- [138] R. Bellman, "A markovian decision process," *Indiana Univ. Math. J.*, vol. 6, pp. 679–684, 1957. [2.4.1](#)
- [139] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press, 2018. [2.4.1](#)
- [140] Wikipedia contributors, "Huber loss — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Huber_loss&oldid=995902670, 2020. [Online; accessed 12-January-2021]. [2.4.1](#)
- [141] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3, AAAI'08*, p. 1433–1438, AAAI Press, 2008. [2.4.1](#)
- [142] O. Kroemer, H. van Hoof, G. Neumann, and J. Peters, "Learning to predict phases of manipulation tasks as hidden states," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4009–4014, 2014. [2.4.1](#)
- [143] J. Yuan, H. Wang, L. Xiao, W. Zheng, J. Li, F. Lin, and B. Zhang, "A formal study of shot boundary detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 2, pp. 168–186, 2007. [2.4.1](#)
- [144] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013. [2.4.3](#)
- [145] M. D. Killpack, A. Kapusta, and C. C. Kemp, "Model predictive control for fast reaching in clutter," *Autonomous Robots*, vol. 40, no. 3, pp. 537–560, 2016. [2.4.3](#)
- [146] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the hrp-2 humanoid," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3346–3351, IEEE, 2015. [2.4.3](#)

- [147] G. Garimella and M. Kobilarov, "Towards model-predictive control for aerial pick-and-place," in *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 4692–4697, IEEE, 2015. [2.4.3](#)
- [148] D. Lunni, A. Santamaria-Navarro, R. Rossi, P. Rocco, L. Bascetta, and J. Andrade-Cetto, "Nonlinear model predictive control for aerial manipulation," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 87–93, IEEE, 2017. [2.4.3](#)
- [149] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx," in *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 4397–4404, IEEE, 2015. [2.4.3](#)
- [150] D. Limon, J. Calliess, and J. Maciejowski, "Learning-based nonlinear model predictive control," *20th IFAC World Congress*, vol. 50, no. 1, pp. 7769–7776, 2017. [2.4.3](#)
- [151] S. Manschitz, J. Kober, M. Gienger, and J. Peters, "Learning to sequence movement primitives from demonstrations," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4414–4421, IEEE, 2014. [2.4.3](#)
- [152] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine, "Solar: Deep structured representations for model-based reinforcement learning," in *International Conference on Machine Learning*, pp. 7444–7453, 2019. [2.4.3](#)
- [153] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013. [2.4.3](#)
- [154] A. Rocchi, E. M. Hoffman, D. G. Caldwell, and N. G. Tsagarakis, "Opensot: a whole-body control library for the compliant humanoid robot coman," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6248–6253, IEEE, 2015. [2.4.4](#)

- [155] T.-T. Do, A. Nguyen, and I. Reid, "Affordancenet: An end-to-end deep learning approach for object affordance detection," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1–5, IEEE, 2018. [2.4.4](#)
- [156] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2961–2969, 2017. [2.4.4](#)
- [157] D. Paulius and Y. Sun, "A survey of knowledge representation in service robotics," *Robotics and Autonomous Systems*, vol. 118, pp. 13–30, 2019. [2.4.4](#)
- [158] C. Dornhege, M. Gissler, M. Teschner, and B. Nebel, "Integrating symbolic and geometric planning for mobile manipulation," in *2009 IEEE International Workshop on Safety, Security & Rescue Robotics (SSRR 2009)*, pp. 1–6, IEEE, 2009. [2.4.4](#)
- [159] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, 2015. [2.4.5](#)
- [160] Wikipedia contributors, "Meta learning — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Meta_learning&oldid=998509683, 2021. [Online; accessed 23-June-2021]. [2.4.6](#)
- [161] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on Robot Learning*, pp. 1094–1100, PMLR, 2020. [2.4.6](#)
- [162] T. Yu, C. Finn, S. Dasari, A. Xie, T. Zhang, P. Abbeel, and S. Levine, "One-shot imitation from observing humans via domain-adaptive meta-learning," *Proceedings of Robotics: Science and Systems (RSS)*, 2018. [2.4.6](#), [2.2](#), [2.3](#), [4.6](#)
- [163] T. Yu, P. Abbeel, S. Levine, and C. Finn, "One-shot hierarchical imitation learning of compound visuomotor tasks," *arXiv preprint arXiv:1810.11043*, 2018. [2.4.6](#), [2.2](#), [2.3](#)

- [164] N. A. Lynnerup, L. Nolling, R. Hasle, and J. Hallam, "A survey on reproducibility by evaluating deep reinforcement learning algorithms on real-world robots," in *Proceedings of the Conference on Robot Learning*, vol. 100 of *Proceedings of Machine Learning Research*, pp. 466–489, PMLR, 30 Oct–01 Nov 2020. [2.6](#)
- [165] G. Rizzolatti and L. Craighero, "The mirror-neuron system," *Annu. Rev. Neurosci.*, vol. 27, pp. 169–192, 2004. [3.1](#)
- [166] L. Cattaneo and G. Rizzolatti, "The mirror neuron system," *Archives of neurology*, vol. 66, no. 5, pp. 557–560, 2009. [3.1](#)
- [167] A. Lago-Rodríguez, B. Cheeran, G. Koch, T. Hortobagay, and M. Fernandez-del Olmo, "The role of mirror neurons in observational motor learning: an integrative review," *European Journal of Human Movement*, vol. 32, pp. 82–103, 2014. [3.1](#)
- [168] G. Rizzolatti, "The mirror neuron system and its function in humans," *Anatomy and embryology*, vol. 210, no. 5-6, pp. 419–421, 2005. [3.1](#)
- [169] Y. Wang, M. Long, J. Wang, and P. S. Yu, "Spatiotemporal pyramid network for video action recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1529–1538, 2017. [3.1](#)
- [170] S. Herath, M. Harandi, and F. Porikli, "Going deeper into action recognition: A survey," *Image and vision computing*, vol. 60, pp. 4–21, 2017. [3.1](#)
- [171] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of NAACL-HLT*, pp. 2227–2237, 2018. [3.2](#)
- [172] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," technical report, OpenAI, 2018. [3.2](#)
- [173] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," technical report, OpenAI, 2019. [3.2](#)

- [174] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020. [3.2](#)
- [175] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Association for Computational Linguistics, June 2019. [3.2](#)
- [176] Y. Xie and D. Richmond, "Pre-training on grayscale imagenet improves medical image classification," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 0–0, 2018. [3.2](#)
- [177] 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 37th International Conference on Machine Learning*, 2020. [3.2](#)
- [178] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby, "Big transfer (bit): General visual representation learning," in *ECCV*, 2020. [3.2](#)
- [179] P. Goyal, M. Caron, B. Lefaudeaux, M. Xu, P. Wang, V. Pai, M. Singh, V. Liptchinsky, I. Misra, A. Joulin, *et al.*, "Self-supervised pretraining of visual features in the wild," *arXiv preprint arXiv:2103.01988*, 2021. [3.2](#)
- [180] M. Marszałek, I. Laptev, and C. Schmid, "Actions in context," in *IEEE Conference on Computer Vision & Pattern Recognition*, 2009. [3.1](#)
- [181] K. K. Reddy and M. Shah, "Recognizing 50 human action categories of web videos," *Machine vision and applications*, vol. 24, no. 5, pp. 971–981, 2013. [3.1](#)

- [182] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: a large video database for human motion recognition," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. [3.1](#)
- [183] O. Kliper-Gross, T. Hassner, and L. Wolf, "The action similarity labeling challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 615–621, 2011. [3.1](#)
- [184] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012. [3.3.2](#)
- [185] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, pp. 568–576, 2014. [3.3.2](#)
- [186] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014. [3.3.2](#)
- [187] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *European conference on computer vision*, pp. 20–36, Springer, 2016. [3.3.2](#)
- [188] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010. [3.3.2](#)
- [189] K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," in *Theoretical Foundations of Machine Learning (TFML)*, 2017. [3.3.3](#)
- [190] Y. Ding, L. Wang, and B. Gong, "Analyzing deep neural network's transferability via frechet distance," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3932–3941, 2021. [3.3.4](#)

- [191] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011. [3.3](#), [4.7](#)
- [192] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, (Berkeley, Calif.), pp. 281–297, University of California Press, 1967. [3.4.2](#)
- [193] L. Hubert and P. Arabie, “Comparing partitions,” *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985. [3.4.2](#)
- [194] Wikipedia contributors, “Rand index: Wikipedia, the free encyclopedia.” https://en.wikipedia.org/w/index.php?title=Rand_index&oldid=957507615, 2020. [Online; accessed 24-July-2020]. [3.4.2](#)
- [195] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, “Scikit-learn: Machine learning in python,” *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011. [3.4.2](#)
- [196] D. Arthur and S. Vassilvitskii, “k-means++: The advantages of careful seeding,” technical report, Stanford, 2006. [3.5](#)
- [197] C. Elkan, “Using the triangle inequality to accelerate k-means,” in *Proceedings of the 20th international conference on Machine Learning (ICML-03)*, pp. 147–153, 2003. [3.5](#)
- [198] A. Rosenberg and J. Hirschberg, “V-measure: A conditional entropy-based external cluster evaluation measure,” in *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pp. 410–420, 2007. [3.4.2](#)
- [199] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, “Factors of transferability for a generic convnet representation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 9, pp. 1790–1802, 2015. [3.4.2](#)

- [200] B. Athiwaratkun and K. Kang, "Feature representation in convolutional neural networks," *arXiv preprint arXiv:1507.02313*, 2015. [3.4.2](#)
- [201] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999. [3.4.4](#), [4.3.4](#)
- [202] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *International conference on database theory*, pp. 420–434, Springer, 2001. [3.4.5](#)
- [203] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016. [4.3](#)
- [204] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, Oct 2012. [4.3](#)
- [205] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations*, 2015. [4.3.1](#)
- [206] 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*, vol. 518, no. 7540, pp. 529–533, 2015. [4.3.1](#)
- [207] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on Machine Learning*, vol. 32, pp. 387–395, PMLR, 22–24 Jun 2014. [4.3.1](#)
- [208] R. Bellman, "The theory of dynamic programming," *Bull. Amer. Math. Soc.*, vol. 60, pp. 503–515, 11 1954. [4.3.1](#)
- [209] H. Zheng, Z. Yang, W. Liu, J. Liang, and Y. Li, "Improving deep neural networks using softplus units," in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–4, IEEE, 2015. [4.3.1](#)

- [210] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Phys. Rev.*, vol. 36, pp. 823–841, Sep 1930. [4.3.1](#)
- [211] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *AAAI*, 2018. [4.3.2](#)
- [212] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005. [4.3.3](#)
- [213] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*, pp. 1–4, Springer, 2009. [4.3.4](#)
- [214] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emons, A. Gupta, E. Orbay, *et al.*, "Roboturk: A crowdsourcing platform for robotic skill learning through imitation," in *Conference on Robot Learning*, pp. 879–893, 2018. [4.6](#)
- [215] A. Mandlekar, J. Booher, M. Spero, A. Tung, A. Gupta, Y. Zhu, A. Garg, S. Savarese, and L. Fei-Fei, "Scaling robot supervision to hundreds of hours with roboturk: Robotic manipulation dataset through human reasoning and dexterity," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1048–1055, IEEE, 2019. [4.6](#)
- [216] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019. [4.3.5](#)
- [217] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6023–6032, 2019. [4.3.5](#)
- [218] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*, pp. 4569–4574, IEEE, 2011. [4.4.1](#)

- [219] W. C. Agboh and M. R. Dogar, "Pushing fast and slow: Task-adaptive planning for non-prehensile manipulation under uncertainty," in *International Workshop on the Algorithmic Foundations of Robotics*, pp. 160–176, Springer, 2018. [4.4.1](#)
- [220] A. Plaat, W. Kusters, and M. Preuss, "Model-based deep reinforcement learning for high-dimensional problems, a survey," *arXiv preprint arXiv:2008.05598*, 2021. [4.4.1](#)
- [221] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," in *International Conference on Learning Representations*, 2019. [4.4.1](#)
- [222] C. Finn, M. Zhang, J. Fu, X. Tan, Z. McCarthy, E. Scharff, and S. Levine, "Guided policy search code implementation," 2016. Software available from rll.berkeley.edu/gps. [4.4.1](#)
- [223] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009. [4.4.2](#)
- [224] 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*, vol. 37, no. 4-5, pp. 421–436, 2018. [5.2](#)
- [225] A. Yamaguchi and C. G. Atkeson, "Recent progress in tactile sensing and sensors for robotic manipulation: can we turn tactile sensing into vision?," *Advanced Robotics*, vol. 33, no. 14, pp. 661–673, 2019. [5.3.1](#)
- [226] W. Chen, H. Khamis, I. Birznieks, N. F. Lepora, and S. J. Redmond, "Tactile sensors for friction estimation and incipient slip detection—toward dexterous robotic manipulation: A review," *IEEE Sensors Journal*, vol. 18, no. 22, pp. 9049–9064, 2018. [5.3.1](#)

- [227] Y. Shi, F. Gong, M. Wang, J. Liu, Y. Wu, and H. Men, "A deep feature mining method of electronic nose sensor data for identifying beer olfactory information," *Journal of Food Engineering*, vol. 263, pp. 437–445, 2019. [5.3.1](#)
- [228] V. Tiwari, "Mfcc and its applications in speaker recognition," *International journal on emerging technologies*, vol. 1, no. 1, pp. 19–22, 2010. [5.3.1](#)
- [229] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *22nd ACM International Conference on Multimedia (ACM-MM'14)*, (Orlando, FL, USA), pp. 1041–1044, Nov. 2014. [5.3.1](#)
- [230] R. Qian, T. Meng, B. Gong, M.-H. Yang, H. Wang, S. Belongie, and Y. Cui, "Spatiotemporal contrastive video representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6964–6974, 2021. [5.3.2](#)
- [231] Y. Zhou, H. Dong, and A. El Saddik, "Deep learning in next-frame prediction: A benchmark review," *IEEE Access*, vol. 8, pp. 69273–69283, 2020. [5.3.2](#)
- [232] R. Glatt and A. Costa, "Policy reuse in deep reinforcement learning," in *Proceedings of the AAI Conference on Artificial Intelligence*, vol. 31, 2017. [5.3.3](#)
- [233] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *Journal of Machine Learning Research*, vol. 21, no. 181, pp. 1–50, 2020. [5.3.5](#)
- [234] F. Mahdisoltani, G. Berger, W. Gharbieh, D. Fleet, and R. Memisevic, "Fine-grained video classification and captioning," *arXiv preprint arXiv:1804.09235*, vol. 5, no. 6, 2018. [6.1](#)
- [235] H. Zhao, A. Torralba, L. Torresani, and Z. Yan, "Hacs: Human action clips and segments dataset for recognition and temporal localization," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8668–8678, 2019. [6.1](#)

- [236] A. Diba, M. Fayyaz, V. Sharma, M. Paluri, J. Gall, R. Stiefelhagen, and L. Van Gool, "Large scale holistic video understanding," in *European Conference on Computer Vision*, pp. 593–610, Springer, 2020. [6.1](#)
- [237] Q. Huang, Y. Xiong, A. Rao, J. Wang, and D. Lin, "Movienet: A holistic dataset for movie understanding," in *The European Conference on Computer Vision (ECCV)*, 2020. [6.1](#)
- [238] L. Smaira, J. Carreira, E. Noland, E. Clancy, A. Wu, and A. Zisserman, "A short note on the kinetics-700-2020 human action dataset," *arXiv preprint arXiv:2010.10864*, 2020. [6.1](#)
- [239] M. Kuhn and K. Johnson, *Feature engineering and selection: A practical approach for predictive models*. CRC Press, 2019. [6.3](#)