Reaping the Rewards of Autonomous Robotic Harvesting: Imitation Learning of Picking Motions Directly from Video

By

Antonios Porichis

Supervisors: Dr Vishwanathan Mohan, Dr Anirban Chowdhury, Dr Panos Chatzakos

School of Computer Science and Electronic Engineering

University of Essex

A thesis submitted for the degree of

Doctor of Philosophy

October 2024

To Theodosis-Konstantinos, Eleni-Emmanuela, and Katerina,

Acknowledgements

This is the hardest section to write; one cannot always express their gratitude as thoroughly as necessary, particularly when they are not writing in their native tongue. With moderate hesitation and enormous excitement, I proceed.

I would like to extend my heartfelt thanks to my academic supervisors Dr. Vishwanathan Mohan and Dr. Anirban Chowdhury. Their deep expertise and insightful advice have been crucial in deepening my understanding of the nuances of Imitation Learning and the challenges of applying it to robotic manipulation tasks. Vishuu, your balanced combination of deep theoretical knowledge and practical mindset is rare among the world of academia, and it has been invaluable throughout this research.

I must also express my deepest gratitude to Dr. Panos Chatzakos, my industrial supervisor, a man of rare intellect and the most genuine care for helping people to grow I have ever encountered. Pano, I consider myself lucky to have enjoyed your mentorship.

My research was made possible by the sponsorship and support of Lloyd's Register Foundation. The work was enabled through, and undertaken at, the National Structural Integrity Research Centre (NSIRC), a postgraduate engineering facility for industry-led research into structural integrity established and managed by TWI through a network of both national and international Universities. The greatest part of this work has received funding from the EU's Horizon 2020 research and innovation programme, under grant agreement no. 101017054 (SoftGrip project). A big thank you to my colleagues and close collaborators in the SoftGrip project, Konstantinos Vasios, Thanasis Mastrogeorgiou, Dora Panteliou, Leonidas Delimpasis, Nikos Kegkeroglou and Myrto Inglezou.

My academic endeavour would have been impossible without my family's support. To my father Theodosis, and my mother Chrysoula, thank you for sacrifice and for instilling in me the values of hard work and perseverance. Your endless belief in my potential has been a powerful force. Panagiotis and Dimitris-Stefanos, thank you for showing your older brother the importance of creativity, dedication and commitment to a goal. Your jokes and wit have provided much-needed laughter during challenging times, reminding me not to take life too seriously.

To my wife Katerina, any choice of words seems shamefully inadequate to express the depth of my gratitude to you. You are my shining light. Your tremendous patience, your sense of commitment and persistence have been my guiding beacon through the highs and lows of this fascinating journey. To my son, Theodosis-Konstantinos, and my daughter Eleni-Emmanouela, your hugs and smiles have been my greatest joy and source of motivation. Thank you all from the bottom of my heart.

Abstract

The agricultural sector faces severe challenges; amidst strong pressures for cost saving while maintaining sustainable practices in the face of climate change, and steep labour shortages which significantly disrupt harvesting logistics, there is tremendous value to be unlocked by robotic automation. Yet, a large portion of crops is still harvested manually, requiring human workers carry out tedious, menial tasks under conditions that raise considerable health and safety risks. Harvesting requires a combination of motions such as reaching, grasping, twisting, and pulling, all under force constraints to avoid damaging the crop being harvested. This sequence can seem effortless for human expert harvesters thanks to the tremendous capabilities and physical intelligence of the human hand, but it is extremely difficult, and at times utterly impossible, to program into a robotic controller. Imitation Learning allows a robotic agent to learn how to mimic an expert in accomplishing an activity completely circumventing the need for programming explicit procedures. This thesis explores Imitation Learning pipelines for robotic harvesting. We focus on optimizing the performance of robots in the delicate task of mushroom harvesting, leveraging end-to-end learning techniques that are adaptable across various crops. The novel contributions in this work include (i) the design of efficient representations to improve action prediction accuracy while keeping computational costs low, (ii) the development of a one-shot Imitation Learning approach for mushroom harvesting enabling high success rates from a single expert demonstration combined with a small number of auxiliary trajectories that are cheap and straightforward to obtain and (iii) the introduction of a novel, end-to-end trainable Representation Learning module that produces interpretable representation, significantly boosting the transparency of the overall Imitation Learning pipelines. These advancements lay the groundwork for the next generation of robotic automation in agriculture, ultimately facilitating significant enhancements to productivity, sustainability and resilience for this strategic sector.

Contents

Acknowledgements iv
Abstract v
Contents vi
List of Figures ix
List of Tables xv
List of Abbreviations xvi
1 Introduction 1
1.1 Digital technologies in agriculture – challenges and opportunities 1
1.2 Case In Point - Robotic Mushroom Harvesting 2
1.2.1 Environmental Drivers 4
1.2.2 Social Drivers 5
1.3 Contributions of this Thesis 5
1.4 Overview 7
2 Imitation Learning Background and Related Work 9
2.1 Background 9
2.1.1 Deep Neural Networks 9
2.1.2 Variational Autoencoders 12
2.1.3 Vector Quantisation 14
2.1.4 Mixer Models 15
2.2 Related Work 17
2.2.1 Inverse Reinforcement Learning 17
2.2.2 Behavioural Cloning 18
2.3 Few-shot Imitation Learning 20
2.4 Explainable AI 23

2.5 Representation and Object Centric Learning 24
3 Mushroom Harvesting Human Expert Demonstration Collection 27
3.1 Data Streams and Recording Equipment 27
3.1.1 RGB-D Video Stream 27
3.1.2 Tactile sensing stream 28
3.2 Data acquisition hardware 31
3.3 Data Overview 33
3.3.1 RGB-D Data 33
3.3.2 Tactile Data 33
3.3.3 Synchronised Sequences 35
3.4 Data Annotation 36
4 Behavioural Cloning for Mushroom Picking in Simulated environment 39
4.1 Imitation Learning Architecture 39
4.2 Experiments 41
4.3 Results 46
4.4 Discussion 48
5 Behavioural Cloning for mushroom picking with a rigid gripper 51
5.1 Imitation Learning Architecture 51
5.2 Environment 53
5.3 Experiments 56
5.4 Results 59
5.5 Discussion 63
6 One-shot Imitation Learning for Autonomous Mushroom Picking 65
6.1 Imitation Learning Architecture 65
6.2 Environment 68
6.3 Experiments 70
6.4 Results 73

6.5 Discussion 80
7 Interpretable Representations for Imitation Learning 83
7.1 Representation Learning Architecture 84
7.2 Environments 87
7.2.1 Generic Robotic Manipulation Tasks 87
7.2.2 Mushroom Picking Task 88
7.3 Experiments 89
7.3.1 Lift, Can and Square manipulation tasks 89
7.3.2 Mushroom picking task 90
7.4 Results 91
7.4.1 Lift, Can and Square manipulation tasks 91
7.4.2 Mushroom picking task 94
7.5 Discussion 97
8 Conclusions and Future Work 99
8.1 General Discussion 99
8.2 Retrospective Analysis of our Work 100
8.3 Directions for Future Work 102
References 104

List of Figures

Figure 1.1 Four size categories of Agaricus bisporus ready for picking. The cap diameter can span a range from 12cm down to 2cm: (a) 40mm diameter closed cup, (b) 50 mm diameter closed cup, (c) 60mm diameter open cup, (d) 90mm diameter closed cup. 3

Figure 1.2. Picking mushrooms is one of the most physically demanding jobs. 3

Figure 2.1 Conceptual schematic of a CNN showing the three types of layers [22] 11

Figure 2.2. Conceptual schematic of a VAE operation 13

Figure 2.3 The architecture of the VQ-VAE [23] 14

Figure 2.4 The MLP mixer architecture [25] 15

Figure 2.5 The ConvMixer architecture [27] 16

Figure 2.6 Time-contrastive network architecture. A reward function is learned through contrastive learning using pairs of aligned videos. Negative and positive samples are drawn based on temporal distance. [32] 17

Figure 2.7 The Behaviour Transformer architecture. The implementation includes discretising the action by splitting each vector into a discrete component via a simple k-means process while keeping the residual. A standard, small-scale Generative Pre-training Transformer is used as the action decoder [40]. 19

Figure 2.8 The Perceiver-Actor architecture. The approach implements a language-conditioned behavior-cloning agent trained with supervised learning to detect actions. The input is a language goal and a voxel grid reconstructed from RGB-D sensors. The models operate on 3D patches of the voxelised grid [43]. 20

Figure 2.9 The Diffusion Policy architecture. The diffusion model refines noise into actions via a learned gradient field. This formulation is considered to stabilise training, enabling the learning of multi-modal action distributions, and accommodating high-dimensional action sequences [46]. 20

Figure 2.10 The WHIRL's approach three main components. The robot first obtains human priors such as hand movement and object interactions by watching the expert and then post processing the image to remove the expert from each frame via inpainting. These priors are repeated by interacting in the real world, trying to achieve task success and explore around the prior. The task

policy is improved leveraging an agent-agnostic objective function which aligns human and robot videos [58]. 21

Figure 2.11 The coarse-to-fine IL approach. The task is deconstructed into a Visual Servoing and a Manipulation subtask. The former is learned based on a dataset collected by moving the end-effector to various positions above the target, while the latter is simply replayed from the single expert demonstration [59]. 22

Figure 3.1. RGB-D camera positions. 27

Figure 3.2. Expert demonstration collection setup. 28

Figure 3.3 Pressure map profiles tracked by Tekscan Grip System 28

Figure 3.4. Used glove with stains indicating finger regions most frequently contacting the mushroom surface. 29

Figure 3.5. Tekscan Grip System sensing elements mounted on the mushroom picker's gloves. 30

Figure 3.6. Mushroom Picking trial using the sensorised glove. 30

Figure 3.7 Force sensors mounted on the expert picker gloves. 31

Figure 3.8 The ZED camera is connected to the Jetson. A trigger signal initiates the recording procedure. This signal is sent from the Tekscan output port and received at a Jetson GPIO pin. 32

Figure 3.9 Topology of the Jetson devices, the ZED2 cameras and the PC that records the camera streams. 32

Figure 3.10 RGB-D camera 1: Left Image, Right Image and Depth Array 33

Figure 3.11 RGB-D camera 2: Left Image, Right Image and Depth Array 33

Figure 3.12 Aggregated pressures on fingertips (Tekscan Grip System). 34

Figure 3.13 Aggregated pressures on finger mid-sections (Tekscan Grip System). 34

Figure 3.14 Forces on finger tips (Mitsui sensors). 35

Figure 3.15 Overlay - finger region correspondence. 35

Figure 3.16 Mushroom picking sequence with pressure maps overlay 36

Figure 3.17 State sequence for recording presented in Figure 14 and Figure 15. 37

Figure 3.18 Segmentation of the index tip pressure profile. 37

Figure 4.1 Architecture and flow of data (solid lines) and gradients (dashed lines) during training. Orange flow: The pretraining sequence, where the RepL module is trained on random trajectories. Blue Flow: Joint training of both the RepL and the BC modules where the gradients from the LBC flow all the way back to the encoder of the RepL module © 2023 IEEE. 41

Figure 4.2 PyBullet debug window rendering for the gym simulation environment for mushroom harvesting © 2023 IEEE 42

Figure 4.3 A sequence of a failed outrooting attempt; (1) the gripper starts at a random position, (2) moves fingers around the mushroom, (3) grasps the mushroom, (4-5) pulls the mushroom upwards without twisting resulting in slippage, (6) mushroom completely slips away 44

Figure 4.4 A sequence of a successful episode rollout by the trained agent. (1) the gripper starts at a random position, (2-3) moves fingers around the mushroom, (4) grasps the mushroom, (5) twists, and (6) pulls the mushroom upwards. © 2023 IEEE 45

Figure 4.5 The trajectories (x, z position and yaw angle) and the squeezing forces observed on the gripper fingers of the trained agent (cyan) and the expert demonstrator (orange). © 2023 IEEE 47

Figure 4.6 (a) RGB image, (b) depth map with artificial noise. © 2023 IEEE..... 48

Figure 5.1 Architecture and flow of data (blue lines) and gradients (orange and yellow lines) during training. Dashed lines indicate gradient copying to accommodate non differentiable modules. The ampersand (&) symbol indicates concatenation. 51

Figure 5.2 Top- and side-view of the five mushroom models used in the picking experiments. 53

Figure 5.3 Our experimental setup; Left: the robotic manipulator (xArm6 by Ufactory) and an example of the mushroom 3D-printed mock-ups. Right: The 3D printed mount and the camera (Realsense D435f) used for imaging 54

Figure 5.4 Failed attempt without twisting; (1) the gripper starts at the random position, (2) moves and grasps the mushroom mock-up, (3) starts pulling upwards causing mushroom mock-up to slip, (4) mushroom mock-up completely slips away 55

Figure 5.5 Successful attempt with twisting; (1) the gripper starts at the random position, (2) moves and grasps the mushroom mock-up, (3) twists the mushroom mock-up, breaking some of the adhesive bonds, (4) pulls mushroom mock-up upwards successfully 56

Figure 5.6 A sequence of a successful episode rollout by the trained agent. (1) the gripper starts at a random position; (2) moves above the mushroom; (3) reaches down; (4) grasps; (5) twists, breaking the adhesive bonds; and (6) pulls the mushroom upwards. 58

Figure 5.7 The trajectories (x, y, z position and yaw angle) observed on the gripper fingers of the trained agent (skyblue) and the expert demonstrator (orange). 59

Figure 5.8 Grouped bar chart of success rates for each of the 8 models tested for each of the 5 mushroom models. 61

Figure 5.9 Top: embeddings with VQ, Bottom: embeddings without VQ. Color shade indicates the episode step index 62

Figure 6.1 Schematic of Imitation learning driven visual servoing approach. The architecture integrates an Image Encoder for processing RGB images, an Image Decoder for frame reconstruction, a Vector Quantization (VQ) module for embedding quantization based on a codebook that is updated using Exponential Moving Average (EMA), and a Target Position Decoder for predicting the Target Position. The system is trained in a self-supervised manner, using a combined loss function that includes reconstruction loss (L_IR), VQ loss (L_VQ), and the Target Position loss (L_TP), facilitating accurate end-effector positioning based on visual and encoder inputs. Dashed lines denote gradient copying to account for the fact that the quantization operation is not differentiable per se. 66

Figure 6.2 The actuated cartesian robotic system used for mushroom picking trials. The gantry-like robot moves along the x-axis with actuated wheels (M1), and along the y-axis and the z-axis with linear slides (M2 and M3 respectively). 69

Figure 6.3 The in-hand camera position as well as the motors (M1, M2 and M3) of controlling the motion of the gripper. Only M3 which controls the twisting motion was active during our experiments. 70

Figure 6.4 Four screenshots of the eye-in-hand camera during a spiral trajectory, Right: 3D plot of the trajectory of the gripper in the 3D space. 71

Figure 6.5 3D plot of the trajectory of the gripper in the 3D space 72

Figure 6.6 Front view and view of a sequence of a successful episode rollout by the trained agent on real mushrooms. (1) the gripper starts at a random position; (2) moves above the mushroom; (3) reaches down; (4) grasps; (5) twists; and (6) pulls the mushroom upwards. 74

Figure 6.7 Eye-in-hand view of a sequence of a successful episode rollout by the trained agent on real mushrooms. (1) the gripper starts at a random position; (2) moves above the mushroom; (3) reaches down; (4) grasps; (5) twists; and (6) pulls the mushroom upwards. 75

Figure 6.8 Predictions of different IL pipelines on the expert demonstration. Predictions are produced offline, based on the observations obtained during the expert trajectory. 76

Figure 6.9. Expert and trained agent (vq-rec) trajectories on rollouts with the same environment conditions. Trained agent predictions are produced online by rolling out a new episode. 77

Figure 6.10 t-SNE mapping of embeddings produced by the three IL approaches considered. Colour shade indicates the episode step index. 78

Figure 6.11 Annotated screenshots during trained agent (vq-rec) episode rollouts: (a) the bottom left right finger is significantly displaced due to collision with a nearby mushroom, (b) the top finger is moderately displaced due to hysteresis. 79

Figure 7.1 Interpretable Representation Learning Module Architecture. 85

Figure 7.2 Screenshots from the three simulated tasks, Top: Front camera view, Bottom: Eye-in-Hand camera view. 87

Figure 7.3 Screenshot from simulated mushroom picking task, Left: Eye-in-hand camera view, Right: Front camera view. 88

Figure 7.4 Screenshot from real-world mushroom picking task, Left: Eye-in-hand camera view, Right: Front camera view. 89

Figure 7.5 Indicative example of determining a target mushroom. Right: Target pixel is painted on the image. Left: Extra image channel carrying an inverse modulated distance map as described in eq. (7.9). 91

Figure 7.6 Screenshots of the three highest scores for each of the four slots for a single timestep of Lift environment. 92

Figure 7.7 Screenshots of highest scores for each of the four slots for a single timestep of Can environment. 92

Figure 7.8 Screenshots of highest scores for each of the four slots for a single timestep of Square environment. 93

Figure 7.9 Screenshots of a single slot across multiple episode steps of the Square episode for the eye-in-hand camera. 94

Figure 7.10 Screenshots of highest scores for each of the three slots for a single timestep of the real-world mushroom picking environment. Top: Implementation with target pixel drawn on the image with a red cross. Bottom: Implementation with target pixel concatenated to the embedding. For visualisation purposes, we draw the target pixel in blue colour to distinguish it from the previous case. 96

Figure 7.11 Screenshots of a single slot across multiple episode steps of the real world mushroom picking task. Top: Implementation with target pixel drawn on the image with a red cross. Bottom: Implementation with target pixel concatenated to the embedding. For visualisation purposes, we draw the target pixel in blue colour to distinguish it from the previous case. 96

Figure 8.1 Taxonomy of desired features when it comes to Imitation Learning approaches..... 99

Figure 8.2 Mapping of our implementations with respect to the aforementioned taxonomy . 102

List of Tables

Table 4.1 Key Simulation Parameter Values 43

Table 4.2 Results with Grayscale Image as Input © 2023 IEEE 47

Table 5.1 Success Rates for Different Model Pipelines 60

Table 6.1. The proposed model parameters. 68

Table 6.2 Success rates in mushroom grasping an lifting of different approaches. 73

Table 7.1 Success rates of original and modified implementation of the BC benchmarks. 91

Table 7.2 Success rates for different IL pipelines with different target pixel input modes in simulated mushroom picking task. 95

Table 7.3 Success rates for different IL pipelines with different target pixel input modes in real-world mushroom picking task. 95

List of Abbreviations

ML: Machine Learning

AI: Artificial Intelligence

IL: Imitation Learning

RL: Reinforcement Learning

IRL: Inverse Reinforcement Learning

GAN: Generative Adversarial Network

RepL: Representation Learning

VQ: Vector Quantisation

DNN: Deep Neural Network

MLP: Multi-layer Perceptron

CNN: Convolutional Neural Network

ViT: Vision Transformer

VAE: Variational Auto-Encoder

XAI: Explainable Artificial Intelligence

ELBO: Evidence Lower Bound

KL: Kullback-Leibler

BeT: Behaviour Transformer

DP: Diffusion Policy

1 Introduction

1.1 Digital technologies in agriculture – challenges and opportunities

The labour shortage in agriculture has become a significant and widespread challenge, posing threats to the global food supply chain. This shortage is primarily attributed to several factors, including demographic shifts, changing preferences among the workforce, and increased competition for labour from other industries. Younger generations are increasingly opting for non-agricultural careers, leading to a diminishing pool of skilled and unskilled workers in the farming sector. Additionally, stringent immigration policies in some regions have restricted the flow of foreign labour, further exacerbating the shortage. The consequences of this shortage are far-reaching, affecting crop harvests, productivity, and overall agricultural sustainability. Efforts to address this issue often involve implementing technological solutions, such as automation and robotics, to alleviate the reliance on human labour in agriculture.

Europe's agricultural workforce is expected to fall by 28% until 2030, according a 2018 report by the European Commission [1]. The latest edition estimates a cumulative decline of almost 33% between 2007 and 2032 [2]. This comes largely because of structural changes within the EU agri-food industry and better employment opportunities in other sectors. This decline is not confined to Europe: the U.S. Department of Agriculture's (USDA) Economic Research Service reports that the country's agricultural workforce has been dropping for years. Specifically, the number of self-employed and family farmworkers in the U.S. plunged from 7.6 million in 1950 to 2.06 million in 2000—a 73% reduction[3].

It is therefore clear that wide adoption of digital technologies to alleviate the labour shortage and reduce the need for pesticides is imperative for Europe. However, a crucial factor to consider in developing such technologies is the cost of deployment and cost of ownership considering the unique characteristics of the EU agriculture sector. Consolidation of agricultural land in the EU is slow and the market is still extremely fragmented with over 40% of EU agricultural holdings being less than 5 ha in size. There were 9.1 million agricultural holdings in the EU in 2020, using 157 million hectares of land [4]. This means that the average size of an EU farm is ~17ha. That's less than 10 times smaller than the mean US farm size (444 acres, i.e. 180ha in 2020) [5]. Smaller farms have tighter economic constraints in adopting new technologies. Thus, it is crucial that cost effectiveness is among the top priorities in designing, developing and deploying new technology-based solutions.

1.2 Case In Point - Robotic Mushroom Harvesting

The fresh mushroom industry is under growing pressure to reduce production costs due to high levels of competition between supermarket chains and rival competitors in countries where the cost of labour is significantly lower. Moreover, it is highly labour-intensive, with labour costs at anything up to 50% of the product cost. Much of the manual work in food industry requires rapid, repetitive, and monotonous movement and, consequently, low levels of motivation among workers. This leads to poor quality control and a high incidence of industrial accidents. Therefore, the food industry is looking increasingly towards automation and robotics to help lower production costs further.

EU mushroom grower SMEs are struggling to maintain profit margins due to the globalization of agricultural markets, the increasing cost of raw materials and utilities, and often the use of outdated and inefficient cultivation methods; it is no surprise that fresh mushrooms are often delivered under cost price. In Ireland, one of the largest mushroom producers in Europe farm the farms have been reduced from 504 in 2000 to 40 in 2020 [6], although production has been maintained due to improvements in harvesting efficiency and crop management. This is against a backdrop of severe labour shortages and Brexit [7]. In the future, this number is foreseen to drop even more. It is clear that operations of this magnitude have the need of automating as much as possible their plants for lowering the production costs and therefore being competitive. On the other hand, small growers also need more cost-effective methods; otherwise they are sentenced to disappear.

The white button mushroom (Agaricus bisporus) is the most widely cultivated mushroom in the world (15-36% of total market share) leading mushroom crop worldwide with a global production in 2017 of 4.4 million tonnes [8]. More than 60% of this production is destined for the fresh market, while the rest are processed and canned. Whereas for the canned industry mushrooms can be harvested and handled by an automated system (see State of the Art), such automation does not yet exist for picking and handling mushrooms for the fresh market, due to the high-quality standards required by this product. Although downstream elements of the process can be automated, the actual picking of the mushroom is still done manually, due to the dexterity, precision and sensitivity of the human hand that can pick the mushroom and not damage it. All mushroom that are picked are grouped according to their size and whether the cap is closed or open. Common size specifications are: baby buttons, 20 -35 mm; closed cups, 40-60 mm; and flats, 80-120 mm however depending on the customer, there are many different codes with

individual specifications (e.g. some closed cup are 40-50mm or 40-55mm, some baby buttons codes are 20-30mm and some flat codes are 80-100mm or sometimes 100-120mm.); some examples shown in Figure 1.1 and the mushrooms placed in a tray should be of similar size and, above all, with no damage or blemish on their snow-white skin, making hand picking the only option for producers, with a cost accounting for between 20 and 46% of their total production costs [9].

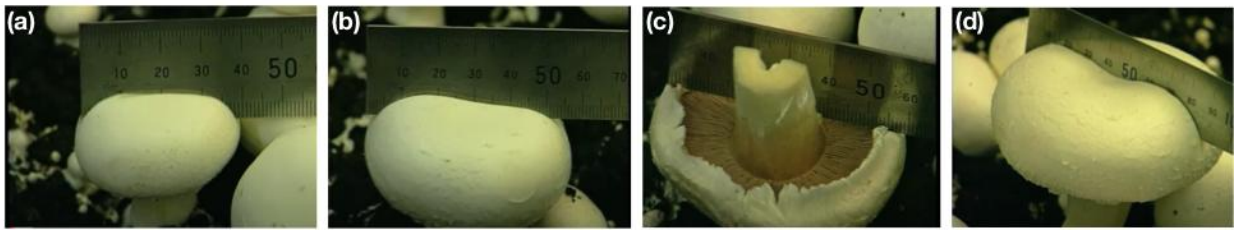

Figure 1.1 Four size categories of Agaricus Bisporus ready for picking. The cap diameter can span a range from 12cm down to 2cm: (a) 40mm diameter closed cup, (b) 50 mm diameter closed cup, (c) 60mm diameter open cup, (d) 90mm diameter closed cup.

In addition, the actual work of picking mushrooms is one of the most physically demanding in the agricultural sector, given the setup of the shelves and the humidity of the growing rooms, causing high incidences of sick leave in detriment to both the workers and their employers (see Figure 1.2). This also makes it difficult for growers to find people willing to work under such conditions.

Figure 1.2. Picking mushrooms is one of the most physically demanding jobs.

Fresh mushroom picking poses significant challenges with respect to the mechatronic functionalities of the robot (hardware and software) required to accomplish the picking task. Despite the tremendous uptake of robotic automation, particularly in indoor vertical farming systems, fresh mushrooms are currently being harvested almost exclusively by human mushroom pickers due to the task's highly complex nature; mushrooms can grow in wildly varying positions and their root system strength is different across different cultivation phases. At the same time, however, mushrooms are extremely sensitive to applied forces and can easily blemish which results in poorer quality and revenue loss for the growers. Thus, human harvesters employ motion patterns combining twisting the mushroom before pulling to achieve outrooting with minimal squeezing force [10]. This composite manipulation pattern in combination with the environment variations make fresh mushroom harvesting a notable example of a task with significant manipulation challenges as well as with substantial industrial impact. Large mushroom growers have made huge improvements in terms of mushroom yields per sq.m. of compost [11] yet they face steep labour shortages that render harvesting a crucial bottleneck in their production pipeline [12], [13]. Thus, enabling automated harvesting of fresh mushrooms is of paramount importance and conventional robotic approaches to this problem require significant engineering in terms of perception, to accurately localize the mushroom, and control, to perform the necessary manipulations.

1.2.1 Environmental Drivers

Mushroom production is one of the most resource efficient and less polluting agricultural practices according to a study by SureHarvest [14]. Water required for mushroom production is a fraction compared to other foods and the production of a 1kg of mushrooms requires only 15 litres of water which is a fraction of water inputs required for many other foods. Growing one kg of mushrooms generates just 0.7 kg of CO2 equivalent, calculated by tracking total emissions from electricity and fuel used for composting equipment and growing operations (e.g. equipment, heating, cooling, etc.). For comparison the carbon footprint of 1kg of beef is 100kg of CO2 equivalent [15], so mushrooms are also a far more sustainable crop, especially considering that they are an excellent source of protein and mushrooms are used to combat protein deficiency and as a supplement to cereal grains [16]. Mushrooms' small growing space conserves soil and the SureHarvest study calculated that growers can produce thousands of tonnes of mushrooms on just a few hectares of land. In addition, the soil used to produce mushrooms is made of composted materials. After mushrooms are harvested, the soil is recycled for multiple

uses, including potting soil. The annual average yield of mushrooms is 34.6 kg per square meter – meaning up to 346 tonnes of mushrooms can be produced on just one hectare.

1.2.2 Social Drivers

Agriculture remains a big employer within the EU; 8.7 million people worked in agriculture in 2020 but only 11.9 % of EU farm managers were under the age of 40 years old in 2020 and farming is a male dominated profession, only 31.6 % of farmers were women in 2020 [4]. This decline is also evident in the US and the number of self-employed and family farmworkers in the U.S.A had been reduced by 73% in the last 30 plus years [17]. According to a 2023 OECD study [18] the agriculture and food sector have the highest rate of skills misalignments (including over and under-qualification) while the demand for employees with higher level entrepreneurial and management skills, and digital know-how is increasing with the largest skill gaps in quality control systems and equipment maintenance.

The role of agriculture is evolving with increasing attention to food security, environmental goods, biodiversity, and social aspects. The way forward is by implementing agricultural policies that encourage investment in skills and human capital and strengthening agricultural innovation and by improving the image of agriculture as a sector capable of offering career opportunities to make it more attractive [18].

1.3 Contributions of this Thesis

There is a significant need for robots to intelligently behave in the face of complex, non-repetitive tasks while minimizing the time and effort required to program a robot for a new activity. Reinforcement Learning (RL) techniques have achieved impressive performance in such complex tasks; however, they require the availability of enormous volumes in terms of data samples as well as per-sample reward annotations which is not always feasible to obtain [19].

Imitation Learning (IL), or Learning by Demonstration, has emerged as an alternative that enables agents to learn how to solve complex tasks by observing the interactions of an expert agent with the environment. Such methods can in principle enable new levels of robotic automation allowing robots to carry out tasks involving high variability with respect to critical characteristics, such as object locations, for which it would be impossible to program a procedural routine [20]. IL, therefore, holds significant promise for industrial applications involving menial work that is still being carried out manually. This is particularly important for sectors where the activities carried out are tedious and of low added value or involve significant health and safety risks.

This thesis introduces several novel contributions towards teaching robots to perform harvesting motions through Imitation Learning. Although our work has primarily focused on fresh mushroom harvesting, we have employed end-to-end learning techniques that can be applied across different crops. Concretely, this thesis makes the following contributions:

1) We investigate Representation Learning pipelines that show significant increases in the action prediction performance of both Behavioural Cloning and one-shot visual servoing agents in the mushroom harvesting setting. These make use of Vector Quantisation techniques under the key assumption that, within a robotic manipulation setting, the latent codes required to describe the observations can be significantly compressed. This performance increases are crucial to achieving acceptable successful picking rates while maintaining the computational complexity of the AI models at moderate levels so that they can be deployed on local computer controllers.
2) We develop a one-shot Imitation Learning approach that takes advantages of the harvesting task composition. Observing that harvesting motion combinations can generally be broken down into a Visual Servoing and a Manipulation phase, with the latter often being straightforward to replicate, we learn a Visual Servoing controller from a single RGB camera that achieves remarkable accuracy in positioning the gripper to grasp and outroot mushrooms. We test this approach on mushroom harvester combining a cartesian mechanism and a bio-inspired soft, pneumatically actuated gripper, moving significantly further than the conventional Imitation Learning benchmarks. The task performances are tested against both highly realistic mushroom mock-ups and real mushrooms and the results are thoroughly analysed.
3) We introduce an interpretable Representation Learning module that enables significant improvements in the transparency of Imitation Learning architectures. This module enables detailed visualization of regions of the image that are regarded as of high importance for the task being solved allowing for easier identification of errors and inherently explaining the model's behaviour. To the best of our knowledge this is the first interpretable Imitation Learning approach for robotic manipulation that can be trained in an end-to-end fashion. We test this Interpretable Representation Learning module into an goal-conditioned Imitation Learning paradigm with straightforward user input; in this regime, a human supervisor indicates the target mushroom to be harvested via a simple click or tap on the camera's visual stream allowing the system to perform the task under

external guidance. This capability is crucial in the mushroom harvesting domain, as prioritising which mushrooms to pick first hugely affects the overall yield.

The work carried out within the scope of this thesis has given rise to the following results:

Two peer-reviewed publications successfully published:

- Porichis, A., Vasios, K., Iglezou, M., Mohan, V., & Chatzakos, P. (2023, June). Visual Imitation Learning for robotic fresh mushroom harvesting. In 2023 31st Mediterranean Conference on Control and Automation (MED) (pp. 535-540). IEEE.
- Porichis, A., Inglezou, M., Kegkeroglou, N., Mohan, V., & Chatzakos, P. (2024). Imitation Learning from a Single Demonstration Leveraging Vector Quantization for Robotic Harvesting. Robotics, 13(7), 98.

Three technical reports as part of the deliverables of the EU-funded project SoftGrip (H2020 project, Grant Agreement No. 101017054), evaluated by experts:

- D5.1 Annotated datasets of human expert demonstrations
- D5.2 Report on multi-task and meta-learning algorithms including neural network architectures
- D5.3 Adaptation and skill transfer learning algorithms

One invited talk titled *"Visual-based Imitation Learning for robotic harvesting with multiple embodiments",* thanks to the kind invitation of Dr. Teena Chakkalayil Hassan as part of her course "Human-Centred Interaction in Robotics" at the Department of Computer Science of the Hochschule Bonn-Rhein-Sieg University of Applied Sciences

One publication aimed at the IEEE Robotics and Automation Letters journal, titled *"Towards Interpretable Representations for Visual-based Imitation Learning"*

1.4 Overview

This Thesis is structured as follows:

- Chapter 2 describes the background and mathematical formulations of the core components we used in our work. It also outline related work with similar aspects to ours in terms of scope.

- Chapter 3 details the methodology of the expert demonstration collection, describing the hardware and electronic components used as well as the annotation of the resulting datasets
- Chapter 4 explains the initial study of using Behavioural Cloning approaches to accomplish mushroom harvesting within a simulation environment.
- Chapter 5 describes the methodologies used to enable harvesting with a real robot using physical mushroom mock-ups, introducing the merits of Vector Quantisation
- Chapter 6 details the methodology used to accomplish harvesting of real mushrooms with a bio-inspired soft, pneumatically actuated gripper focusing on data-efficient approaches
- Chapter 7 explores novel, interpretable representations allowing for transparent Imitation Learning
- Chapter 8 presents the conclusions derived from our work and lay out possible avenues to pursue as future research.

2 Imitation Learning Background and Related Work

2.1 Background

2.1.1 Deep Neural Networks

A deep neural network is a type of machine learning model inspired by the structure and function of the human brain. It consists of multiple layers of interconnected nodes, or "neurons," that process and transform input data to produce an output. Within the context of this thesis, we consider a Deep Neural Network, parametrised by a vector θ, to be a composition of functions:

$$g = f_L \circ f_{L-1} \cdots \circ f_2 \circ f_1 \tag{2.1}$$

where $\circ$ denotes composition, and each f_l denotes the function applied at the l-th layer. Each f_l typically follows the Multi-Layer Perceptron (MLP) structure [21] consisting of two main components, (i) an affine function parametrised by weights $\boldsymbol{W_l}$ and biases $\boldsymbol{b_l}$

$$\boldsymbol{z}_l = \boldsymbol{W}_l \boldsymbol{h}_{l-1} + \boldsymbol{b}_l \tag{2.2}$$

Where $\boldsymbol{h}_{l-1}$ is the output from the previous layer (or the input data for the first layer), and (ii) a non-linear function σ applied elementwise to the linear transformation's output:

$$\boldsymbol{h}_l = \sigma(\boldsymbol{z}_l) \tag{2.3}$$

Common activation functions include:

- the sigmoid function $\sigma(x) = (1 + e^{-x})^{-1}$,
- hyperbolic tangent $\sigma(x) = \tanh(x)$,
- the Rectified Linear Unit (ReLU) $\sigma(x) = \max(0, x)$

These non-linearities introduce the ability to model complex, non-linear relationships in the data.

In the vast majority of applications, DNNs are trained via *gradient descent,* an optimisation algorithm used to minimize the loss function by iteratively updating the network's parameters (weights and biases) in the direction of the negative gradient of the loss function with respect to the parameters:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla \mathcal{L}(\boldsymbol{\theta}) \tag{2.4}$$

were, θ represents the parameters, η is the learning rate, and $\mathcal{L}(\boldsymbol{\theta})$ is the loss function. Variants like stochastic gradient descent (SGD) and adaptive methods like Adam and RMSprop are commonly used to improve convergence speed and performance.

The “layered” structure of DNNs allows the use of *backpropagation*, an efficient formulation of the gradient computation leveraging the chain rule of derivation. Backpropagation involves two passes through the network: a forward pass to compute the output and loss, and a backward pass to propagate the error and compute the gradients.

During the backward pass, the chain rule of calculus is applied to compute the gradient of the loss function with respect to each parameter. For a weight matrix $\boldsymbol{W}_l$ in layer l, the gradient is computed as:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_L} \cdot \frac{\partial \boldsymbol{h}_L}{\partial \boldsymbol{z}_L} \cdot \ldots \cdot \frac{\partial \boldsymbol{h}_l}{\partial \boldsymbol{z}_l} \cdot \frac{\partial \boldsymbol{z}_l}{\partial \boldsymbol{W}_l} \tag{2.5}$$

This recursive application of the chain rule allows the network to efficiently compute the gradients for all parameters, enabling effective learning.

Deep Learning, the overall field encompassing the theory around DNNs, is extremely broad and an exceptional number of NN variants have been proposed, usually differing in architecture-related choices around the combination of the functional building blocks and the resulting computational graph. A well-known family of DNNs, particularly suited for processing data with a grid-like topology, such as images, is that of Convolutional Neural Networks (CNNs). CNNs are designed to automatically and adaptively learn spatial hierarchies of features from input data. CNNs are composed of a series of layers that transform the input data through a sequence of operations. The primary building blocks of CNNs include convolutional layers, pooling layers, and fully connected layers.

Convolutional layers apply convolution operations to the input data using a set of learnable filters usually referred to as kernels. Each filter scans the input spatially e.g., across the width and height of an image, to produce feature maps. These feature maps highlight different aspects of the input, such as edges, textures, or more complex patterns. The mathematical formulation of the convolution operation for a single filter $\boldsymbol{K}$ on an input $\boldsymbol{X}$ is defined as:

$$(\boldsymbol{X} * \boldsymbol{K})(i,j) = \sum_m \sum_n \boldsymbol{I}(i+m, j+n)\boldsymbol{K}(m,n) \tag{2.6}$$

where (i,j) denotes the spatial location, and (m,n) are the dimensions of the filter.

Pooling layers reduce the spatial dimensions of the feature maps, effectively downsampling the input. This process helps to reduce the computational load and the number of parameters, as well as to achieve spatial invariance to small translations of the input. Common pooling

operations include max pooling and average pooling. Max pooling, for instance, selects the maximum value within a pooling window w:

$$\boldsymbol{Y}(i,j) = \max_{(m,n) \in w} \boldsymbol{X}(i+m, j+n) \tag{2.6}$$

After a series of convolutional and pooling layers, the high-level reasoning in the neural network is performed through *fully connected*, also termed *dense*, layers. These layers have neurons that are fully connected to all activations in the previous layer, like those in traditional DNNs. Before feeding the feature maps into fully connected layers, they are flattened into a single vector. This vector is then processed by one or more fully connected layers, ultimately leading to the output layer which performs the final prediction. The overall operation of a typical CNN is illustrated in Figure 2.1.

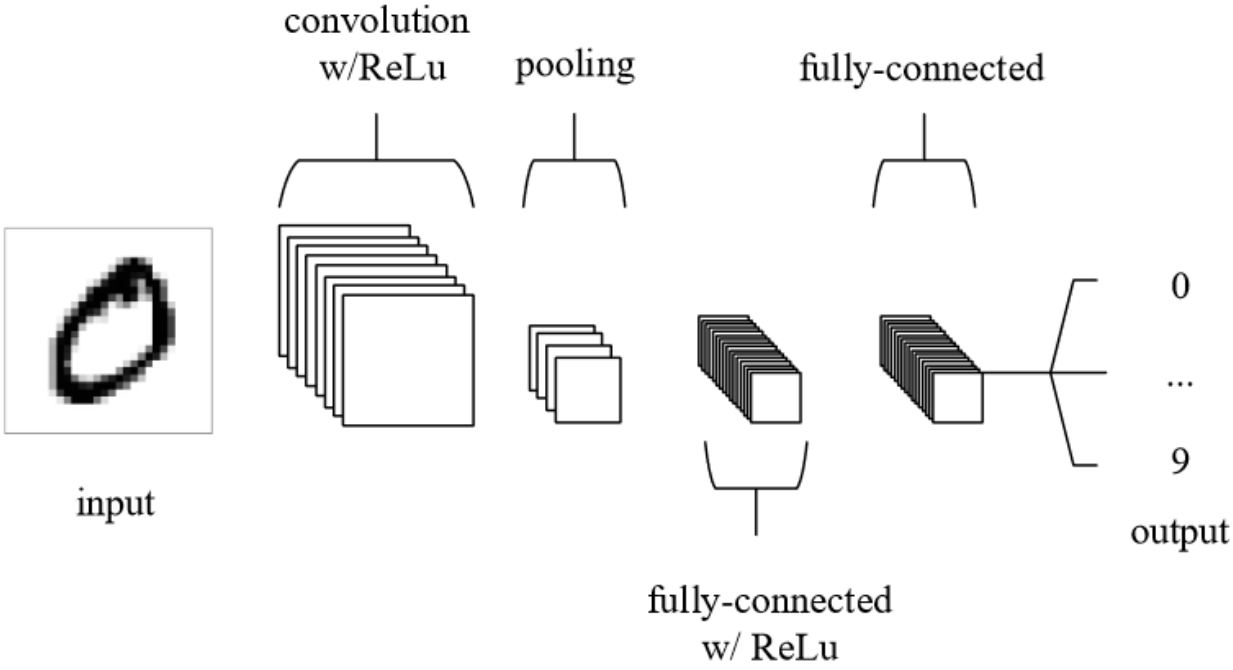

Figure 2.1 Conceptual schematic of a CNN showing the three types of layers [22]

Two key parameters modulating the convolution process; *stride* and *padding*. Stride refers to the number of pixels by which the filter moves across the input. Larger strides reduce the spatial dimensions of the output feature maps. Padding involves adding extra pixels around the input's border, typically zeros, to control the spatial dimensions of the output feature maps and to preserve edge information. In the context of our Imitation Learning work, we rely on stride, rather than pooling, to progressively reduce the input size. This way, we avoid a set of operations, reducing the computational load, while also ensuring that important features and their location

on the image are better preserved. The latter is crucial in determining actions based on image input as opposed to the canonical task of classification where the location of certain features is less important.

Ultimately, CNNs take advantage of parameter sharing, as the same filter, i.e. set of weights, is applied across different parts of the input. Each neuron in a convolutional layer is connected to a local region of the input, called the receptive field. This local connectivity helps the network to learn spatial hierarchies of features and, crucially, significantly reducing the number of parameters compared to fully connected networks, thereby achieving high performance with lower computational loads.

2.1.2 Variational Autoencoders

Variational Autoencoders (VAEs), introduced by Kingma and Welling in 2013, are a class of generative models that combine ideas from variational inference and neural networks. Unlike traditional autoencoders that only compress data, VAEs aim to learn a low-dimensional latent representation of high-dimensional data while simultaneously learning to generate new data samples from this latent space.

A VAE consists of two main components, an encoder and a decoder, both of which are modelled as DNNs. The encoder maps the input data x to a latent space, represented by a set of latent variables $\boldsymbol{z}$. In contrast to deterministic autoencoders, the encoder in a VAE outputs parameters of a probability distribution instead of a single vector. This distribution $q_{\varphi}(\boldsymbol{z} \mid \boldsymbol{x})$ where $\boldsymbol{\phi}$ represents the parameters of the encoder, is typically selected to be the Normal distribution with $\mu(\boldsymbol{x})$ and variance $\sigma^2(\boldsymbol{x})$ which are the outputs of the encoder. For practical reasons, the variance is typically modelled as a diagonal matrix, implicitly assuming that each latent dimension is independent from the rest, and the output of the encoder is $\log \sigma^2(\boldsymbol{x})$. The decoder reconstructs the input data from the latent representation. It maps the latent variables z z back to the data space, providing a distribution $p_{\theta}\ (\boldsymbol{x} \mid \boldsymbol{z})$ where θ denotes the parameters of the decoder.

The objective of the VAE is to maximize the likelihood of the observed data under the model. This however would require the marginal likelihood, or evidence, to be computed:

$$p_{\theta}(\boldsymbol{x}) = \int p(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z}) d\boldsymbol{z} \tag{2.7}$$

This computation is intractable due to the integral over the latent space. Instead, VAEs optimize a tractable lower bound known as the Evidence Lower Bound (ELBO). This is given as:

$$\mathcal{L}(\varphi, \theta; \boldsymbol{x}) = \mathbb{E}_{q_\varphi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})\right] - KL\left(q_\varphi(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p(\boldsymbol{z})\right) \tag{2.8}$$

The two terms of the ELBO lend themselves to a straightforward interpretation from a practical standpoint. The first term, $\mathbb{E}_{q_\varphi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})\right]$, measures how well the decoder reconstructs the input data from the latent variables. It encourages the model to produce samples that are similar to the input data and can thus be considered a reconstruction term. The second term, $KL\left(q_\varphi(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p(\boldsymbol{z})\right)$, is the Kullback-Leibler (KL) divergence between the approximate posterior $q_\varphi(\boldsymbol{z} \mid \boldsymbol{x})$ and the prior $p(\boldsymbol{z})$. This term regularizes the latent space to be close to the prior distribution, typically chosen to be a standard normal distribution $\mathcal{N}(0, \mathrm{I})$. It ensures that the latent space does not deviate too much from a simple distribution, ensuring a smooth manifold for the latent space that facilitates the sampling process.

The optimisation of the ELBO requires computing the gradient of (2.8) with respect to θ, φ. This however is not well defined given that the gradient of the expectation term $\mathbb{E}_{q_\varphi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})\right]$ with respect to ϕ is not straightforward to compute since z is produced stochastically by sampling the Normal distribution with mean and variance produced by the decoder. A way around this obstacle, is the use of the *Reparameterization Trick,* whereby the random variable $\boldsymbol{z}$ is expressed in terms of a deterministic function of $\boldsymbol{x}$ and an auxiliary variable ε with a fixed distribution, which is usually set as the standard normal distribution $\varepsilon \sim \mathcal{N}(0, \mathbf{I})$.

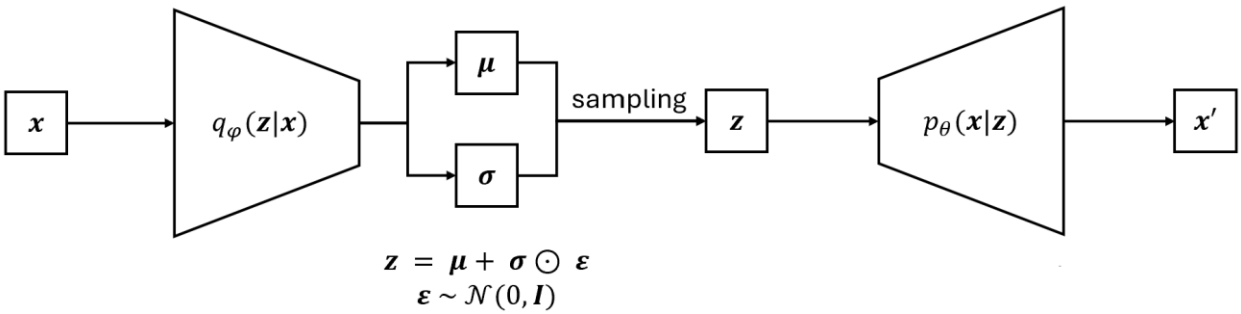

Figure 2.2. Conceptual schematic of a VAE operation

One challenge in training VAEs is that sampling from q*(z|x)* introduces a non-differentiable operation in the computational graph, which prevents backpropagation. The reparameterization trick addresses this issue by reformulating the sampling process in a differentiable manner. This reparameterisation can be written as:

$$\boldsymbol{z} = \mu(\boldsymbol{x}) + \sigma(\boldsymbol{x}) \odot \varepsilon \tag{2.9}$$

Where $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ are produced by the encoder network, and $\odot$ denotes element-wise multiplication. By reparameterising z in this way, the gradients can be passed through the deterministic parts $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$, making the model differentiable with respect to the encoder's parameters ϕ. Thus, gradients can flow through the sampling operation, enabling end-to-end training of the VAE using standard backpropagation techniques. Figure 2.2 depicts the typical architecture of a VAE.

2.1.3 Vector Quantisation

Vector Quantisation in the context of Representation Learning was originally introduced in [23], in the form of a so-called Vector Quantised Variational Autoencoder (VQ-VAE) illustrated in Figure 2.3, and has been widely adopted as a technique enabling the transformation of continuous-valued vectors into discrete codes. This process is essential for applications requiring discrete latent variables, such as generative models and draws inspiration from the simple observation that a mental description of an image would require far less information than the amount afforded by a high-dimensional continuous vector. VQ was a crucial component of the first image generative models to achieve world-wide prominence such as Dall-E [24].

The approach makes use of a *codebook*, i.e. a set of discrete embeddings, also referred to as *code vectors* $\{\boldsymbol{e}_i\}_{i=1}^{K}$, where K is the codebook size, i.e. the number of code vectors within it. Each code vector $\boldsymbol{e}_i$ has the same dimensionality as the latent variables. During training, the codebook is updated to capture the distribution of the latent variables effectively, ensuring that the discrete representations are both informative and compact.

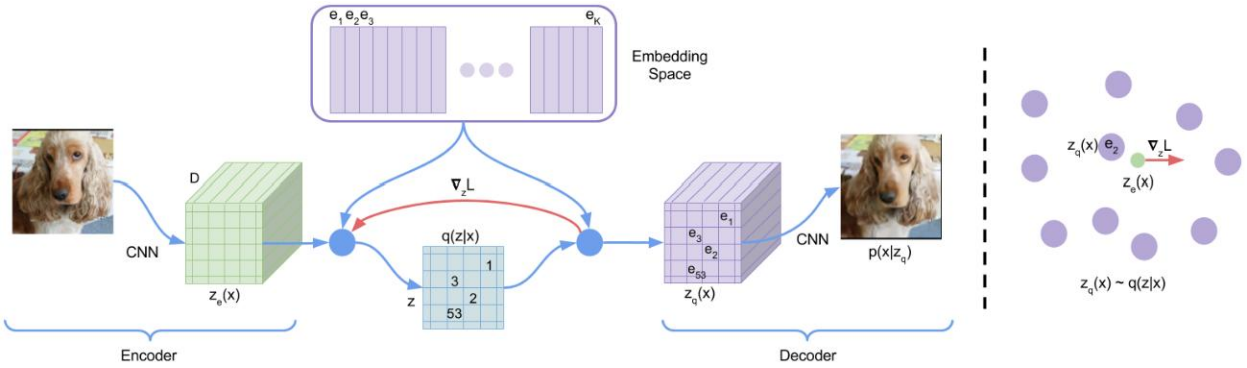

Figure 2.3 The architecture of the VQ-VAE [23]

The quantisation process involves an encoder and a decoder. In the encoding phase, a continuous latent variable $\boldsymbol{z}$ is mapped to its nearest code vector. This nearest-neighbour search is performed using the Euclidean distance metric:

$$k = \underset{i}{\operatorname{argmin}} \parallel \boldsymbol{z} - \boldsymbol{e}_i \parallel_2 \tag{2.9}$$

Once the nearest code vector is identified, $\boldsymbol{z}$ is quantised to $\boldsymbol{e}_k$. In the decoding phase, the quantised representation is processed to reconstruct the original input or generate new data. This discrete code provides a compact and efficient representation of the latent space, facilitating various downstream tasks.

2.1.4 Mixer Models

The concept of mixing for vision applications was first introduced in [25] which presented a novel approach to vision tasks by relying solely on MLPs, diverging from traditional convolutional neural networks (CNNs) and the more recent Vision Transformers (ViTs) [26]. The core idea is to perform spatial and channel mixing to capture complex patterns in image data using a special model architecture made entirely of MLPs, the MLP-Mixer. The MLP-Mixer architecture consists of a series of layers that alternately apply MLPs for *token mixing* and *channel mixing*. The model operates on image patches drawing inspiration from the ViT architecture.

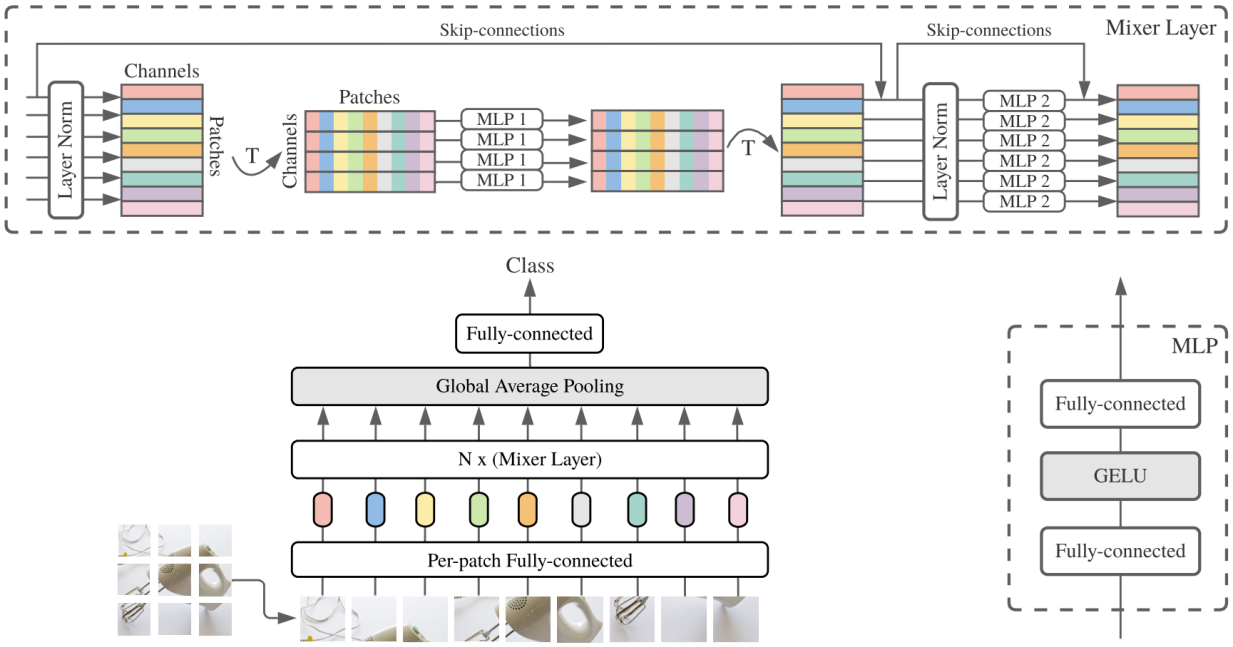

Figure 2.4 The MLP mixer architecture [25]

As seen in Figure 2.4, The MLP-Mixer operation relies on the input image being divided into a grid of non-overlapping patches. Each patch is then linearly embedded into a fixed-dimensional vector thus producing a sequence of patch embeddings. The embeddings are then processed by sequences of Mixer layers, alternating between token mixing and channel mixing MLPs. The token mixing MLPs operate across the patch direction, allowing for "communication" between different patches, i.e. tokens keeping the features of each patch independent. Conversely, the channel mixing MLPs operates across the feature dimension within each patch. The processed patch embeddings from the final Mixer layer are used for classification or other downstream tasks.

The MLP-Mixer model stimulated a line of research for new models operating on patches, revisiting older concepts. The ConvMixer is such a result, introduced by which introduces CNNs back into the [27] architecture with a view to leverage the weight sharing and resulting efficiency of the convolution kernel concept. The ConvMixer, illustrated in Figure 2.5, has shown remarkable results, significantly outperforming much more complex architectures despite its exceptional simplicity and efficiency. Like MLP-Mixer, ConvMixer requires the image to be split into patches and each patch transformed into an embedding. It then proceeds with the mixing process using *depthwise* and *pointwise* convolutions. The former achieve spatial mixing through grouped convolution with groups equal to the number of channels while the latter achieves channel mixing using kernels of size 1x1.

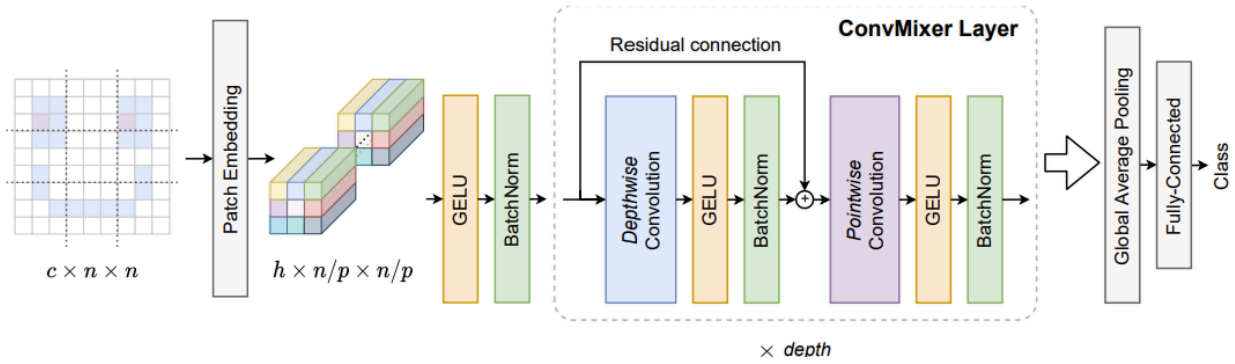

Figure 2.5 The ConvMixer architecture [27]

2.2 Related Work

This section provides an overview of the current state of the art in the field of Imitation Learning. Within this scope, tangential fields of Explainable AI and Representation Learning and their links to Imitation Learning are also analysed.

2.2.1 Inverse Reinforcement Learning

Imitation Learning often takes the form of Inverse Reinforcement Learning (IRL) [28], where the demonstrations are used to learn a reward function that can subsequently be used by a RL algorithm. Finn et al. [29] were among the first to demonstrate a functional IRL method that could operate directly on images, employing a cost learning algorithm based on policy optimization. A visual-based IRL pipeline for robotic manipulation was implemented by Das et al. [30], however their approach involves learning a dynamics model and a processing step that detects key points on the robot configuration. Recent work has shown significant improvements in sample efficiency using Optimal Transport principles [31] but these improvements were demonstrated in simple manipulation tasks only involving pushing rather than grasping. A novel reward learning mechanism has been proposed in [32] using time contrastive networks, where the reward is the inverse of the distance between corresponding embeddings of expert and agent observations.

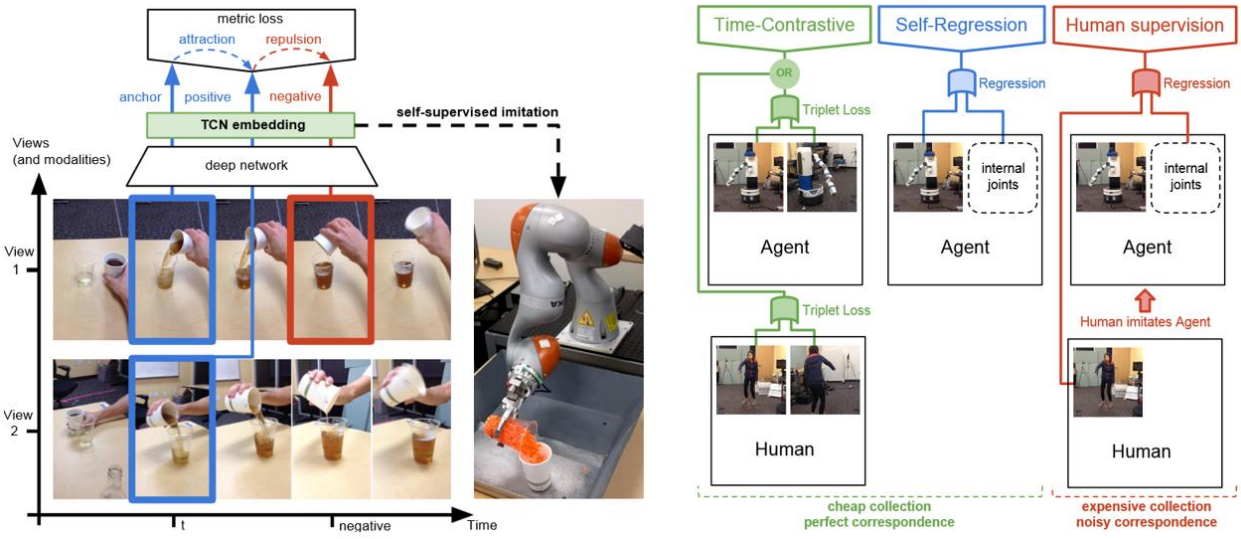

Figure 2.6 Time-contrastive network architecture. A reward function is learned through contrastive learning using pairs of aligned videos. Negative and positive samples are drawn based on temporal distance. [32]

An alternative prominent class of IL methods is Adversarial Imitation Learning (AIL), originally introduced by Ho et al. [33]. This approach draws inspiration from IRL and Generative Adversarial Networks (GANs) [34]. The GAN's Discriminator network is trained to distinguish between samples derived from agent and expert trajectories and its loss is then used as a reward function to drive an RL procedure. Approaches such as Primal Weisserstein Imitation Learning [35] have been successful in robotic manipulation directly from pixel-level input on a simulated door opening task, by using the Weisserstein distance, a key component of GANs, in its primal formulation. Still, to attain sufficient performance this approach requires over 1 million steps of interactions, i.e., more than an order of magnitude higher than the methodologies presented within this thesis.

2.2.2 Behavioural Cloning

By using demonstrations as a proxy to a reward function, IRL and AIL require the agent to be trained by interacting with the environment while still not being able to act close to an optimal and/or safe behaviour. A line of work in IL that avoids this constraint is Behavioural Cloning (BC) originally presented in [36]. This approach directly casts the IL problem as a supervised learning problem, i.e., predicting an action $a = f(o)$ where o is an observation returned by the environment, and f is any Machine Learning model regressing a on o. Despite the presence of distributional shift, i.e., potential instabilities because the i.i.d. requirement for the training data does not hold since past observation-action pairs directly influence the present and future, BC has been shown to perform remarkably well in robotic manipulation by visual inputs. Rahmatizadeh et al. [37] have implemented an visual-based approach to learning five different tasks using BC. This approach requires four different neural networks and several manual adaptations to the pipeline while it relies on >900 demonstrations for certain tasks. Florence et al. [38] proposed Implicit Behavioural Cloning (IBC), where action inference is performed by minimizing the loss of an energy-based model, trained on observation-action pairs, with respect to the action. IBC shows promising results in tasks where optimal trajectories can have multi-modal distributions, training on pixel-based data has only been tested on a block sorting task with a reduced, 2D action space. Other works have been capitalizing on the impressive performance of transformers in other ML-related fields such as Natural Language Processing. The Trajectory Transformer [39] and the Behaviour Transformer (BeT) [40] are two prominent examples. The former follows a pure sequence modelling approach, capturing distribution over trajectories. However, it is an Offline RL algorithm and thus requires a notion of reward. The latter

employs an action discretization technique, using a Generative Pretrained Transformer [41] to predict the action category and the residual.

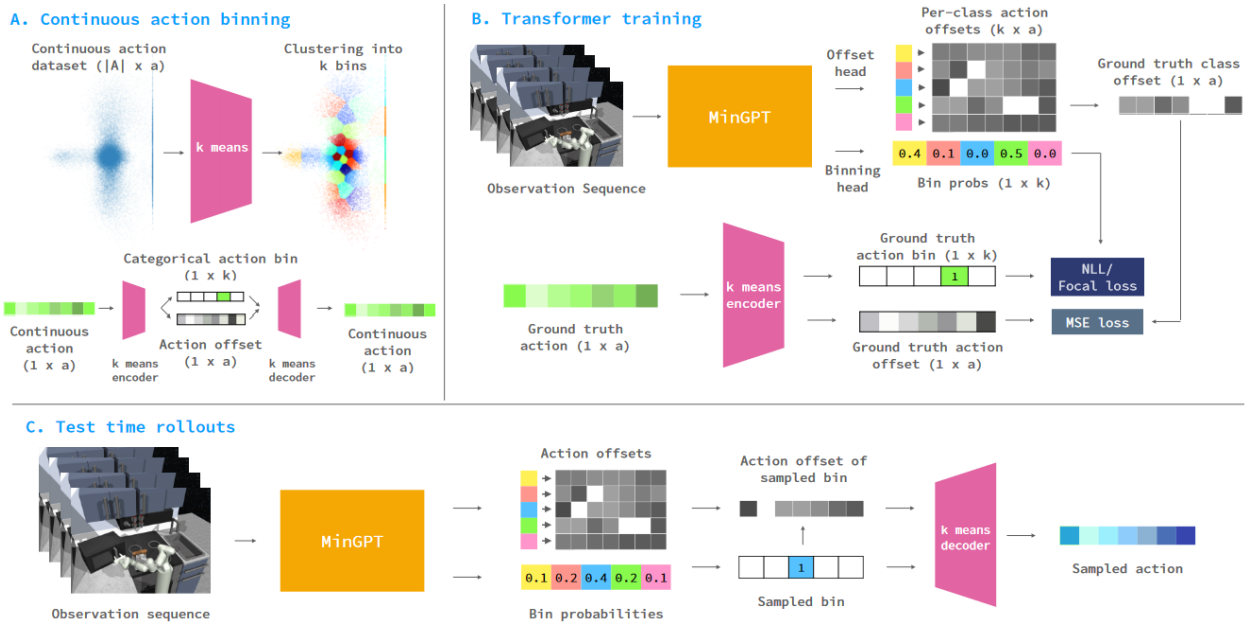

Figure 2.7 The Behaviour Transformer architecture. The implementation includes discretising the action by splitting each vector into a discrete component via a simple k-means process while keeping the residual. A standard, small-scale Generative Pre-training Transformer is used as the action decoder [40].

Zhao et al. [42] have proposed the Action Chunking Transformer for learning complex tasks involving bimanual manipulation. This approach requires four different cameras and model with >80m parameters, an order of magnitude larger than ours. Finally, the Perceiver-Actor architecture [43] shows promising results in language conditioned goal-oriented tasks but it requires registered RGB-D cameras, and it operates on a voxelised representation inducing high computational cost.

Several recent approaches to BC leverage Denoising Diffusion Probabilistic Models [44], drawing inspiration by their recent successes in modelling high-dimensional data including images and videos. These are generative models that map Gaussian noise to some target distribution, usually conditioned on some context-specific embedding. Pearce et al. [45] implement a BC pipeline by using a diffusion-based generative model that is conditioned on the current observation embedding, and maps gaussian noise to the next action. The approach has only been tested on simulated environments and videos. Chi et al. [46] introduced Diffusion Policy, a complex pipeline combining Feature-wise Linear Modulation [47] with a diffusion model that instead of predicting

actions directly, it predicts the gradient field of action energy scores that is then used to obtain a series of actions. This results in a powerful model which however comes at a great computational cost, involving over 250M parameters and requiring days to train on state-of-the art GPUs while requiring a dual camera setup. Due to the iterative nature of diffusion models, these approaches are much slower and require a lot of empirical tweaks for them to be deployed on real systems.

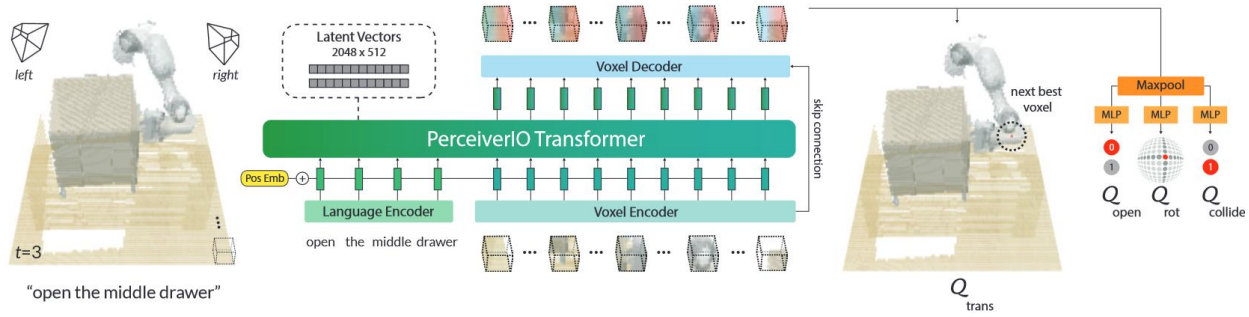

Figure 2.8 The Perceiver-Actor architecture. The approach implements a language-conditioned behavior-cloning agent trained with supervised learning to detect actions. The input is a language goal and a voxel grid reconstructed from RGB-D sensors. The models operate on 3D patches of the voxelised grid [43].

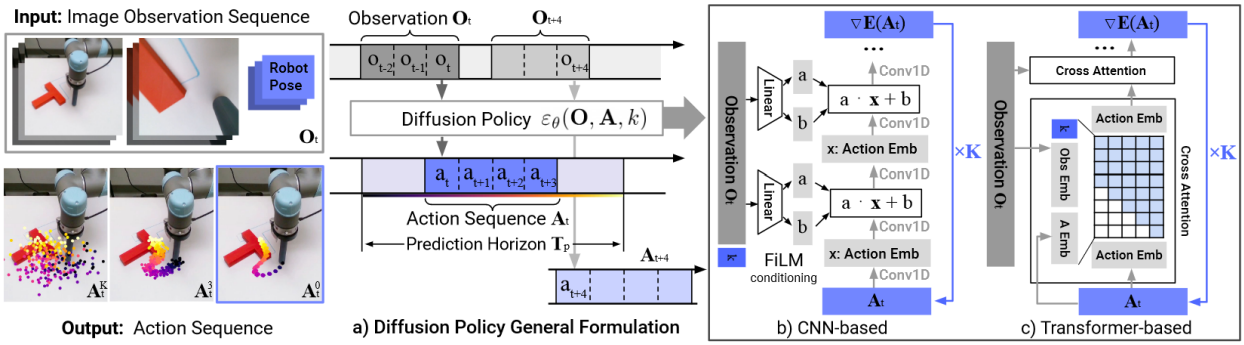

Figure 2.9 The Diffusion Policy architecture. The diffusion model refines noise into actions via a learned gradient field. This formulation is considered to stabilise training, enabling the learning of multi-modal action distributions, and accommodating high-dimensional action sequences [46].

2.3 Few-shot Imitation Learning

We use the term few-shot to describe a class of algorithms that can learn a task from a handful of demonstrations or less, down to one or, in extreme cases, zero expert trajectories. This sort of data efficiency is of paramount importance in the context of robotic manipulation as demonstrations are almost always expensive and cumbersome to collect.

Initial attempts to tackle the problems of data efficiency in Imitation Learning for non-linear settings were based on tiered approaches where local policies were chained or hierarchically combined under a certain context [48], [49], [50], [51]. One of the first attempts to tackle few-shot Imitation Learning from raw pixels was [52] which introduced Meta-Imitation Learning (MIL), in which a policy is learned that can be quickly adapted, via one or few gradient steps at test time, in order to solve a new task given one or more demonstrations, leveraging Model Agnostic Meta Learning [53] algorithms. This gave rise to a broad range of methods based on Multi-task and Meta-Learning techniques [54], [55], [56] which enable robots to generalize to new tasks after having been trained in an initial task set. As a result, few-shot in this context refers to the novel tasks being learned; the original tasks may still require quite a number of expert trajectories.

Another line of work that does not involve meta-learning, tackles the few-shot challenge by using the demonstrations as meaningful priors and allowing the agent to self-explore in a Reinforcement Learning fashion where the reward is essentially a distance between the current trajectory and the expert one at different phases. Pathak et al. [57] introduce a zero-shot technique that enabled a robot to tie a rope knot while using manual annotation of key trajectory points of the demonstration as individual goals. WHIRL, proposed in [58], attempts to achieve one-shot imitation directly from videos by leveraging image processing techniques to remove the agent from the videos and allowing the agent to explore the environment utilizing specific motion primitives. Such approaches indeed require an extremely low number of demonstrations but the requirement to allow the agent to freely explore is not always practical in industrial applications due to safety and cost constraints.

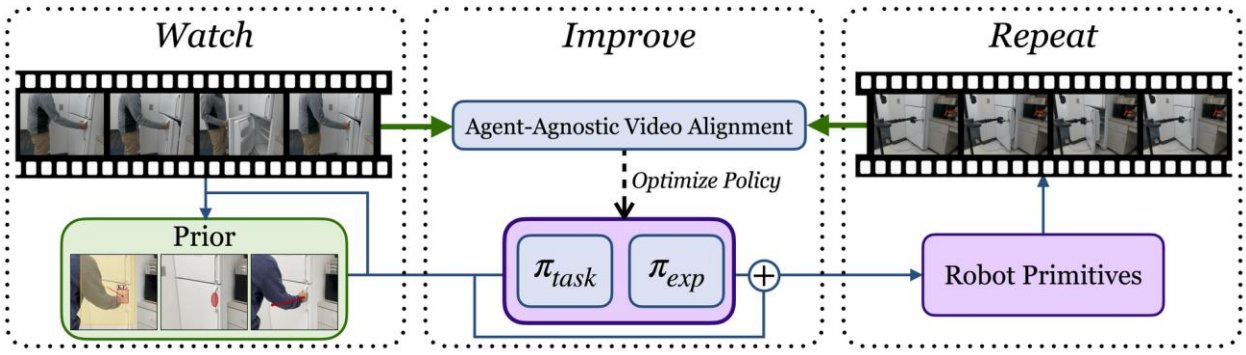

Figure 2.10 The WHIRL's approach three main components. The robot first obtains human priors such as hand movement and object interactions by watching the expert and then post processing the image to remove the expert from each frame via inpainting. These priors are repeated by

interacting in the real world, trying to achieve task success and explore around the prior. The task policy is improved leveraging an agent-agnostic objective function which aligns human and robot videos [58].

Another line of work towards one-shot learning is that of [59], dubbed coarse-to-fine IL, which attempt to derive an IL approach that imitates the visual servoing part which is essential in almost all IL methods pertaining to robotic manipulation, with the object handling part of the trajectory following a scripted policy, i.e. by replaying the original expert demonstration's velocities from the grasping point onwards. Indeed, a vast class of tasks can be broken down in a similar manner and harvesting tasks are no exception. An extension of this work attempts to directly learn to estimate the pose of the target object by providing strong inductive biases to the learning approach leveraging geometrical transformations [60]. This approach, like [59], makes a strong assumption about the structure of the scene, namely that the target object to be manipulated is singular and no distractors exist in the camera's field of view. Another approach that loosens this assumption, allowing for the presence of distractors is that of [61] which employs a sophisticated pipeline that first segments the image to determine the presence of the target object and then processes the image to determine the pose of the target object guiding the visual servoing sequence by trying to match the scale of the detected object with that of the reference trajectory. Although this accomplishes one-shot learning without the use of auxiliary data, it is not practical in our application as we are dealing with scenes where distractors are almost the same as the target object and the target object can vary in scale.

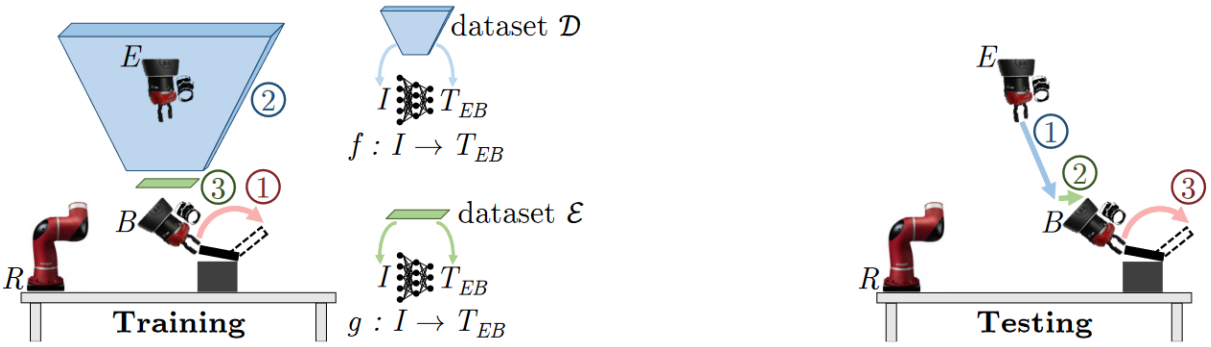

Figure 2.11 The coarse-to-fine IL approach. The task is deconstructed into a Visual Servoing and a Manipulation subtask. The former is learned based on a dataset collected by moving the end-effector to various positions above the target, while the latter is simply replayed from the single expert demonstration [59].

2.4 Explainable AI

Recent years have seen the rise of complex decision-making systems such as Deep Neural Networks (DNNs), which are less transparent than the simpler early AI systems. Deep Learning (DL) models, including DNNs, have achieved significant empirical success owing to their extensive parametric space and powerful learning algorithms. With millions of parameters and numerous layers, DNNs are often called black-box models [62]. These tremendous achievements have enabled a rapid expansion of their usage across all industrial and commercial sectors; indeed one would struggle to identify one aspect of human activity that is not currently being redefined by the use of AI. The opaque nature of such systems however gives rise to significant ethical concerns and issues of trust which have prompted European policy makers to create a framework for Trustworthy AI [63]. A fundamental constituent of Trustworthy AI is Explainable AI (XAI).

XAI approaches can be categorised into ante-hoc and post-hoc [64] where the former refers to using inherently transparent systems such as Decision Trees or Bayesian Models, also frequently termed Intepretable AI, and the latter encompasses methods to obtain insights into the decision process of non-explainable (i.e., black-box) models. Another taxonomy splits XAI techniques into model-specific and model-agnostic with model-agnostic being applied to black-box trained agents as they are independent on the constituents of a model [65]. Model agnostic models are mainly divided into two categories, attribution- and surrogate-based; the former seek to assess the effects of certain aspects of the input most frequently applying perturbations while monitoring the outputs and the latter focus on locally approximating the model under inspection with simpler surrogate models that are easier to explain. Popular model agnostic, post-hoc XAI algorithms include SHAP [66] and LIME [67] respectively. One of the most prominent model-specific schemes is the Layerwise Relevance Propagation [68].

In the context of Reinforcement and Imitation Learning, the need for transparent techniques that move beyond the conventional “black-box” models has attracted strong research interest in recent years [69]. Several works have approached this problem from the policy generation point of view. Initial attempts focused on evolutionary algorithms that produced interpretable policies using fuzzy rules [70], or basic algebraic formulas [71]. Other approaches utilized Decision Trees either distilled from neural network-based policies [72], or evolved through genetic programming [73]. Other important works include [74] which implements Imitation Learning by using the expert trajectories as a basis to extract symbolic reward functions and then relying on Reinforcement Learning, while [75] leverages Signal Temporal Logic to extract formulas that

optimally explain expert trajectories. Another line of research aimed at building causal models, such as directed acyclic causal graphs capturing causal relations to explain environment dynamics and create interpretable policy controllers [76],[77],[78] A crucial limitation of all these works is that they rely on extremely simplified representations of the environment, containing full knowledge of the true state, and their practicality is limited to very small action spaces. Therefore, they are only tested in simplistic environments, such as cart-pole, mountain-car or simple 2D games.

Although these approaches offer robust and mathematically rigid groundwork explainable models, their scalability is limited which is why they have only been demonstrated in tasks with low dimensionality. Visual-based Imitation Learning is significantly more complex. InfoGAIL [79] is among the first explainable techniques to operate on visual inputs, focusing on autonomous driving, however interpretability in this work's context refers to disentangling action modes using a latent variable rather than uncovering some part of the model's decision process per se. There is relatively scarce work in explainable Imitation Learning for robotic manipulation beyond [80] which focuses on a drink pouring task and achieves explainability by imposing a hierarchical structure to the task learning rather than an end-to-end learning approach.

A concept of self-interpretable agents was introduced in [81], where the focus of the interpretability was shifted to the environment representation instead of the policy generation. In this approach, an end-to-end learning pipeline, operating on pixel-based input was introduced, where a self-attention module was responsible for identifying the most important patches of the input image, whose coordinates were then fed to an LSTM that produced the action. The key limitations of this pipeline were that the operation for extracting the important patches is inherently non-differentiable and thus made the use of evolutionary algorithms necessary which severely limited scalability.

2.5 Representation and Object Centric Learning

The vast majority of IL approaches largely rely on well-trusted Computer Vision models such as ResNets [82] as the backbone of the RepL module, often enhanced through techniques such as Feature-wise Linear Modulation [83]. Indeed, when success rate in task solving is the single metric an IL pipeline is judged on, this is a valid choice as explained in [84] which showed that the performance of IL pipelines was largely unaffected by different choices of RepL models.

Previous work in IL leveraging Vector Quantization (VQ) on observations, rather than actions, is sparse. VQ is used in [85] to obtain object-centric representations but the result is processed on a semantic level and the approach focuses on 2D video game playing and self-driving simulation. Another notable work is [86], where VQ is used in a hierarchical IL approach, by quantizing subgoals again in 2D game playing. Our work makes use of VQ directly on visual-based input showing that it can significantly increase the performance of the downstream IL model.

Object-centric learning is a promising field attempting to create representations that focus on objects in the scene. One the first prominent works in this line was [87] which implemented a learning module for determining relevant objects in the scene using a subset of the demonstration trajectories using a separately trained region proposal network. In similar vein, [88] use point detectors for object tracking and embed these in graphs to capture spatial relationships. Such approaches require manual labelling of scene-specific data. To avoid this, recent work capitalizes on the rapid progress of foundational Computer Vision models whose pre-trained versions can be relied on for highly accurate object detection due to their exposure to vast amounts of data during training [89]. Nevertheless, the need to define the concept of "objectness" still remains while there is no guarantee that the specific task scene the robot operates within as well as its point of view will be similar to the dataset the object detection backbone is trained on which can lead to poor performance.

Recent works have investigated approaches for learning object-centric representations in a self-supervised manner directly from the data at hand [90], [91]. This mechanism, termed Slot-Attention attempts to extract a stack of pixel-level semantic masks corresponding to objects on the scene with no previous notion of "objectness". So far these approaches have been tested on artificial datasets with a view to produce representations suitable for Visual Question Answering tasks rather than IL.

3 Mushroom Harvesting Human Expert Demonstration Collection

3.1 Data Streams and Recording Equipment

Expert demonstrations comprise two main data streams, the tactile sensing and the RGB-D video stream. These streams along with the respective pieces of equipment used to record them are described below.

3.1.1 RGB-D Video Stream

The RGB-D stream provides visual and depth information which will be crucial for obtaining high-level semantic-rich information. Four RGB-D sensors were installed on separate locations to obtain a comprehensive view through multiple viewpoints. The RGB-D sensors used in the expert demonstration recording sessions were four instances of the ZED 2 camera. The layout of the four sensors around the mushroom bed is illustrated in Figure 3.1 below while a picture of the setup is shown in Figure 3.2 in the next page.

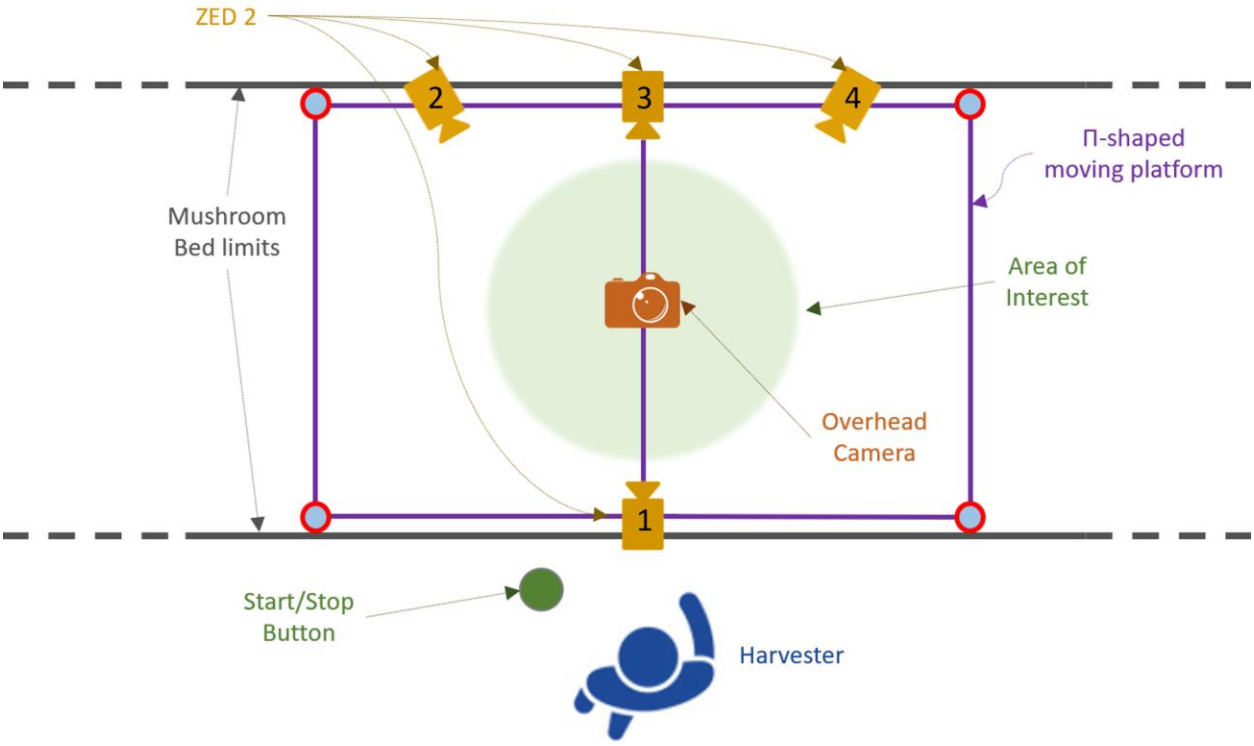

Figure 3.1. RGB-D camera positions.

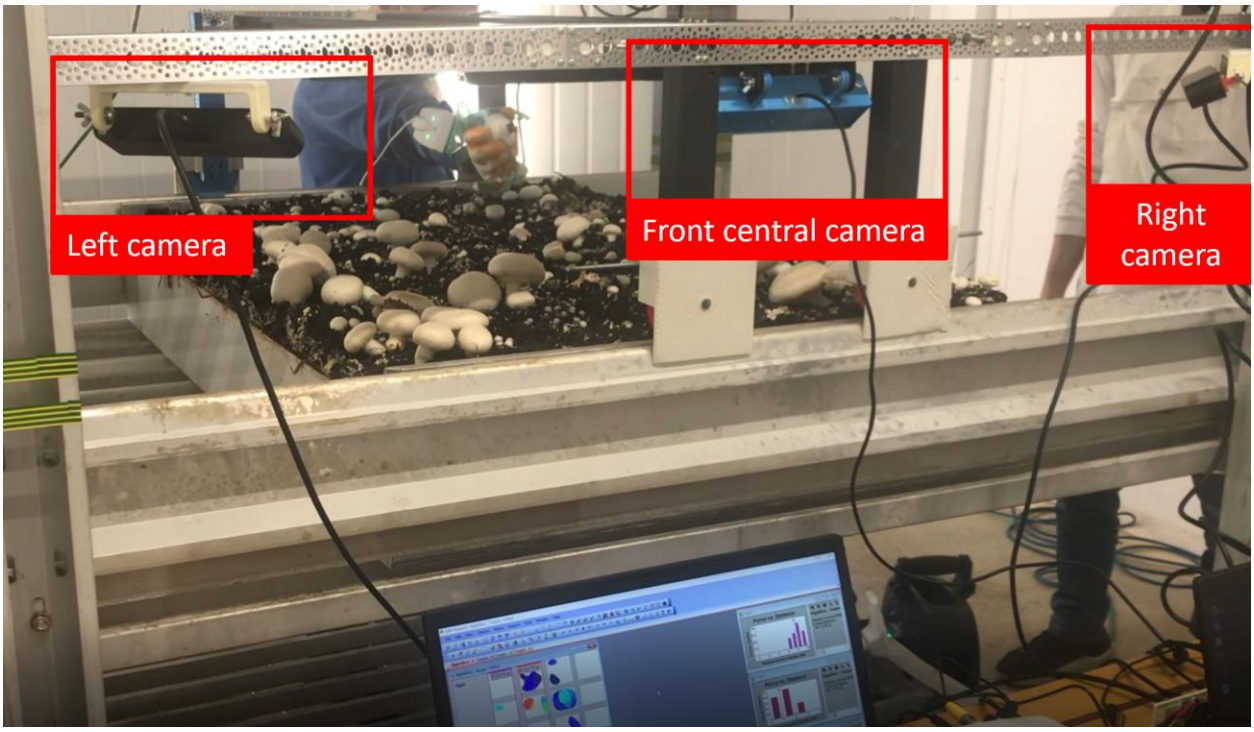

Figure 3.2. Expert demonstration collection setup.

3.1.2 Tactile sensing stream

The main tactile sensing recording system used was the Tekscan Grip System[1]. The system provides >300 sensing elements distributed across the surface of the user's hand providing high-resolution pressure mapping information as seen in Figure 3.3. In the context of mushroom picking demonstration recording, we focused on the sensing elements placed on the Thumb, the Index and the Middle Finger.

Figure 3.3 Pressure map profiles tracked by Tekscan Grip System

[1] Datasheet available https://www.tekscan.com/products-solutions/systems/grip-system

To determine the optimal positions for force and pressure sensing elements to be mounted on the finger, we observed the markings left by mushrooms on regular gloves after a number of pickings. These are illustrated in Figure 3.4. In this way, we were able to gain insight on the points of the finger that are most frequently in touch with the mushroom.

Figure 3.4. Used glove with stains indicating finger regions most frequently contacting the mushroom surface.

Following this observation, the sensing elements were mounted on a new set of mushroom pickers' gloves as shown in Figure 3.5. Figure 3.6 illustrates a mushroom picking experiment carried out by an expert picker.

Figure 3.5. Tekscan Grip System sensing elements mounted on the mushroom picker's gloves.

Figure 3.6. Mushroom Picking trial using the sensorised glove.

Figure 3.7 Force sensors mounted on the expert picker gloves.

In order to investigate the domain shift between the measurements of the Tekscan Grip System and the tactile sensors envisaged for the soft robotic gripper developed within the project, a limited number of trials were conducted using the sensors developed by MITSUI Chemicals. Unlike the Tekscan Grip System sensing elements, which provide pressure maps, these are 6-axis tactile sensors measuring force and torque at a single point. Figure 3.7 above shows these sensors mounted on the mushroom picker's glove.

3.2 Data acquisition hardware

For the camera and the glove recordings to be synchronized, the two streams should start simultaneously. To achieve that, a pulse signal was generated at the beginning of the recording procedure by the Tekscan hub output port. A GPIO pin on the Jetson devices was used to read this pulse, together with a pull-down resistor as shown in the Figure 3.8. The ZED2 cameras are connected to the Jetson via USB 3.0, are initialized and are waiting in standby mode for the pulse signal to initiate the recording. The output is an SVO file from every camera. The executable that interacts with the ZED cameras was developed in C++ using the ZED2 SDK. Lastly, it is important to mention that 4 cameras were used. The Jetson Xavier NX could handle 2 ZED2 cameras while one jetson nano per camera was needed for the last two cameras. Figure 3.9, illustrates how the Jetson devices and the cameras were connected during the data acquisition trials.

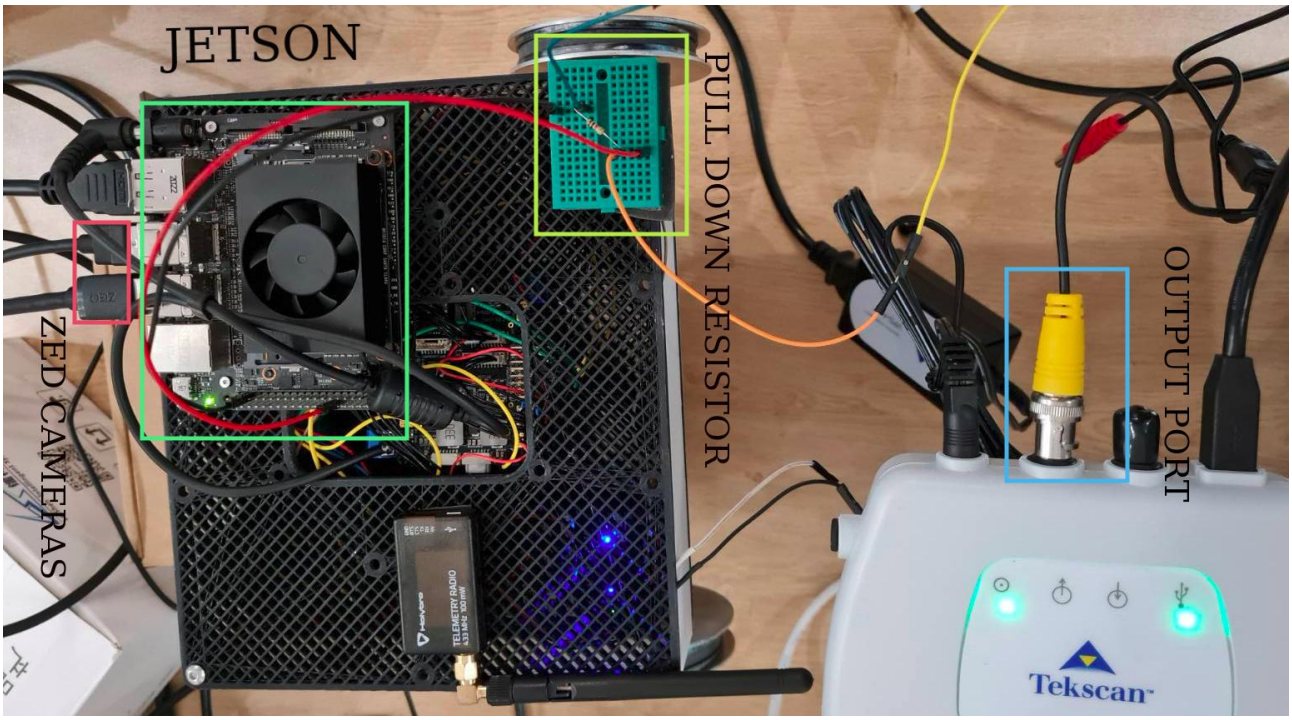

Figure 3.8 The ZED camera is connected to the Jetson. A trigger signal initiates the recording procedure. This signal is sent from the Tekscan output port and received at a Jetson GPIO pin.

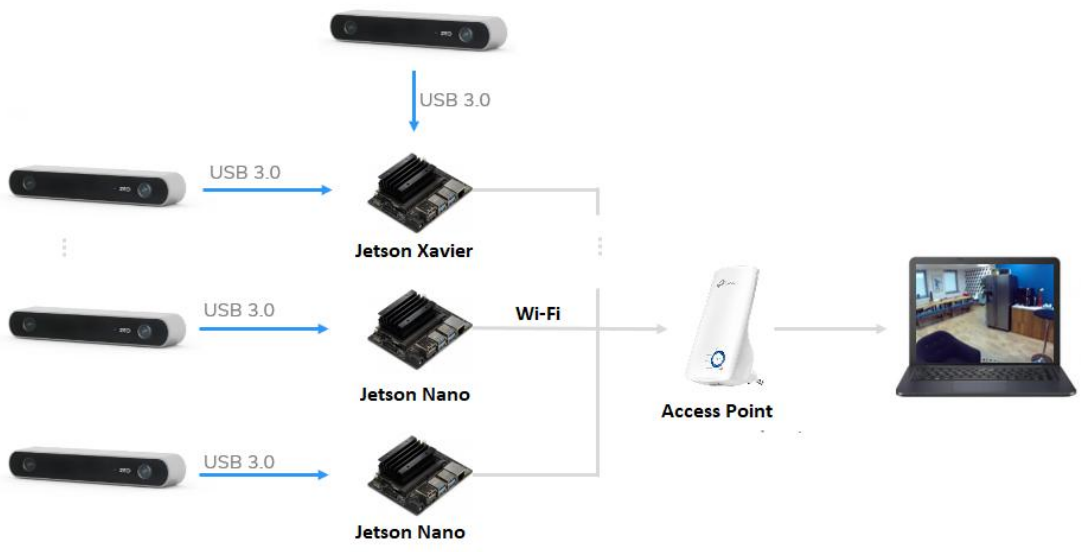

Figure 3.9 Topology of the Jetson devices, the ZED2 cameras and the PC that records the camera streams.

3.3 Data Overview

3.3.1 RGB-D Data

Indicative screenshots of the RGB-D streams are provided in the next figures.

Figure 3.10 RGB-D camera 1: Left Image, Right Image and Depth Array

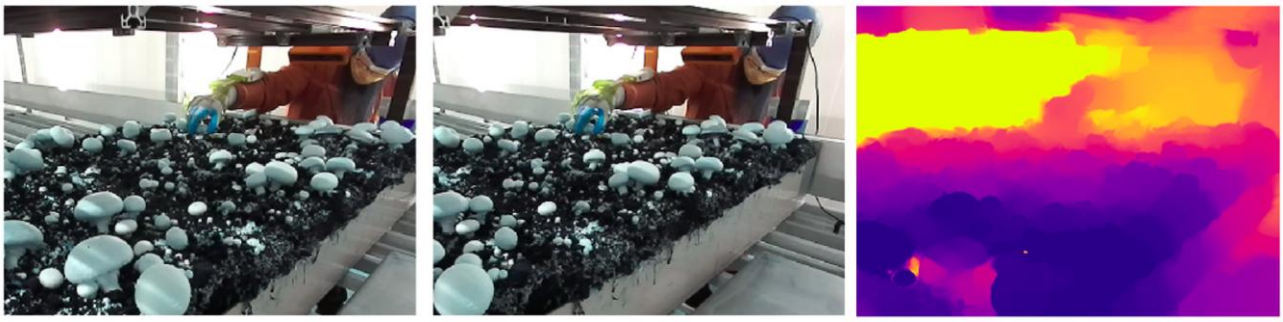

Figure 3.11 RGB-D camera 2: Left Image, Right Image and Depth Array

3.3.2 Tactile Data

The following figures illustrate the aggregated pressures retrieved from the Tekscan Grip System. The aggregation was performed by way of averaging the pressures recorded at each region of the finger. Figure 3.12 illustrates the time-series of the aggregated pressures on the tips of the three fingers (thumb, index and middle finger) while Figure 3.13 depicts the corresponding pressures at the finger mid-sections. These time-series are obtained for the same picking trial as the one illustrated in Figure 3.10 and Figure 3.11.

In this picking trial, it seems that the pressure from the thumb tip is dominant while the pressures from the index and middle fingers is relatively evenly distributed between the tip and the mid-section. The picker initially touches the mushroom around the 10th second of the recording and the twisting and tilting motion starts between the 11th second. The peak observed around the 12th second marks the successful breaking of the roots and from that point on the mushroom is being lifted.

The illustration of force measurements retrieved by the force sensors are provided in Figure 3.14. As observed in Figure 3.12 and Figure 3.14, pressure and force measurements by the two systems provide similar time-series profiles.

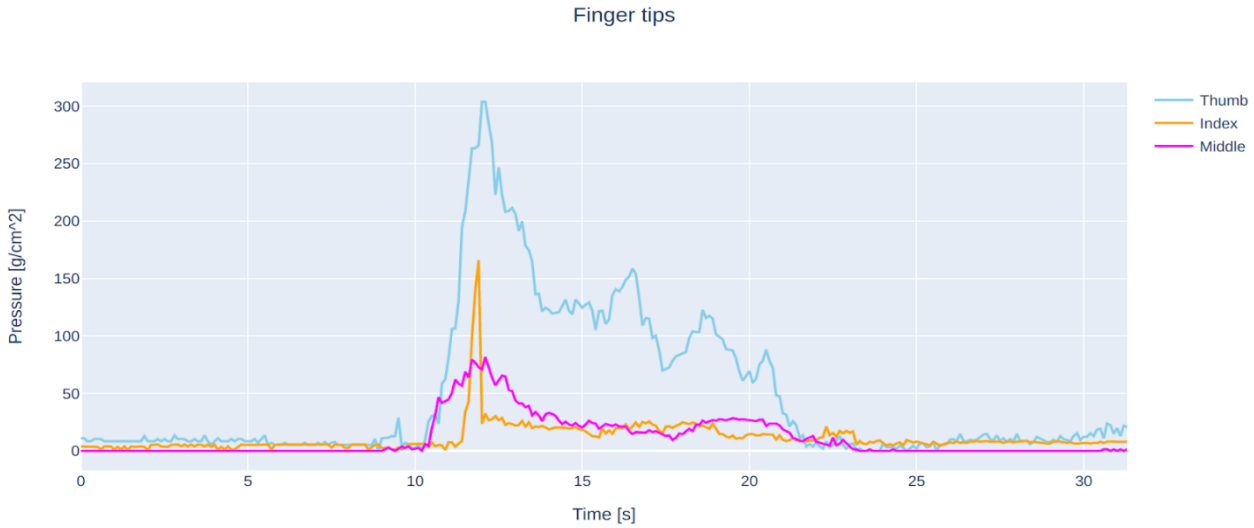

Figure 3.12 Aggregated pressures on fingertips (Tekscan Grip System).

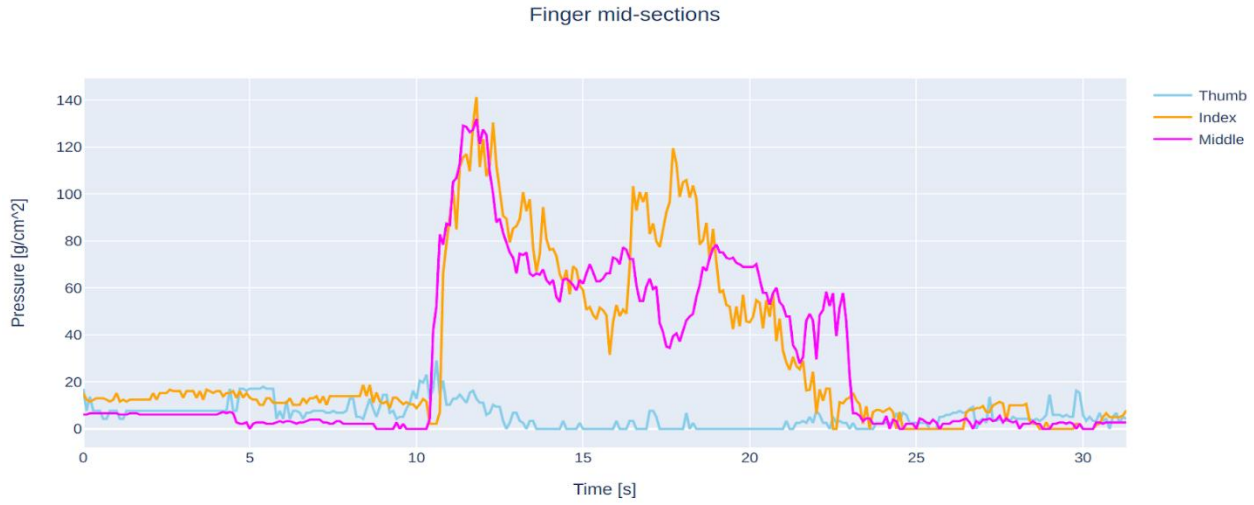

Figure 3.13 Aggregated pressures on finger mid-sections (Tekscan Grip System).

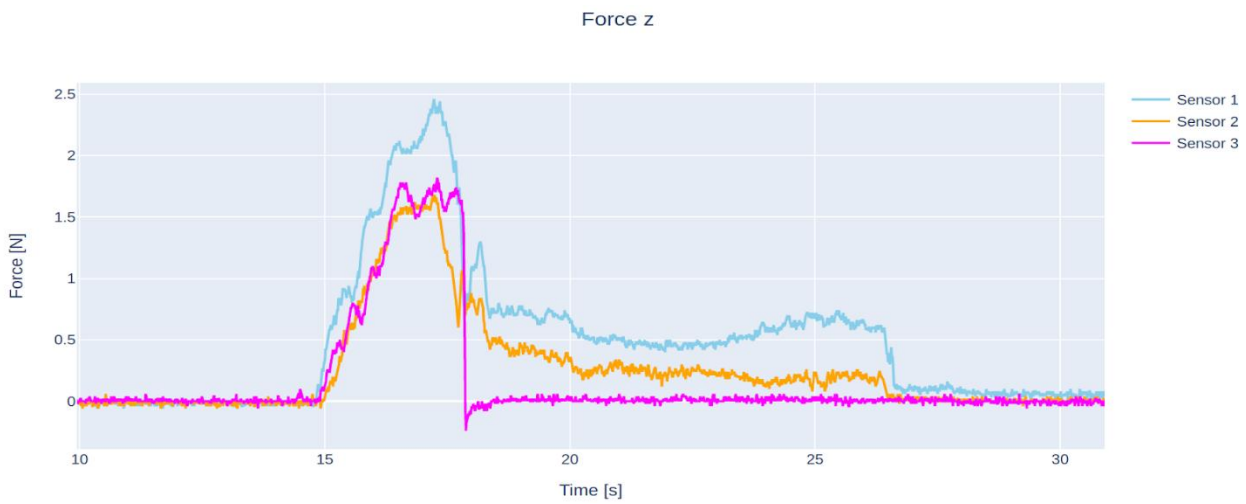

Figure 3.14 Forces on finger tips (Mitsui sensors).

3.3.3 Synchronised Sequences

For a comprehensive overview of the picking process both in terms of the visual and the pressure maps streams, Figure 3.16 illustrates four screenshots, each belonging to a different state, of the picking attempt considered throughout Figure 3.12, Figure 3.13, Figure 3.17 and Figure 3.18 with the pressure maps reading overlaid in the bottom right corner of each image. The correspondence of the each overlayed patch with the finger regions is provided in Figure 3.15. As seen in Figure 3.16, there is a clear distinction between the pressure patterns observed during the different states, with measurements being increasingly apparent during the "Touching" and "Outrooting" states and gradually receding towards the lifting phase.

Figure 3.15 Overlay - finger region correspondence.

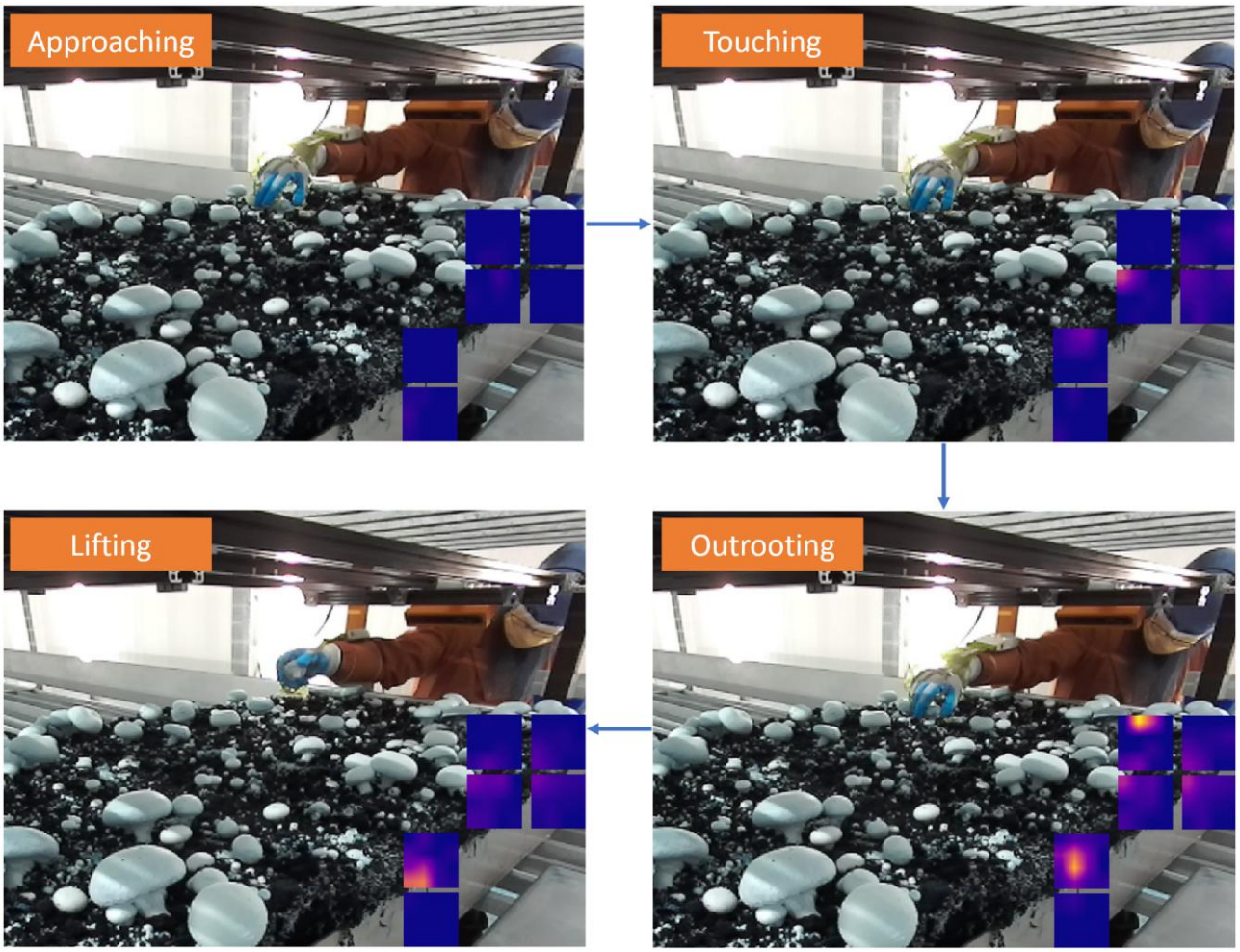

Figure 3.16 Mushroom picking sequence with pressure maps overlay

3.4 Data Annotation

By examining the pressure time-series in combination with the visual streams we have performed an annotation of the recording based on four states, namely “Approaching”, “Touching”, “Outrooting” and “Lifting”. The state annotation corresponding to the picking trial considered in Figure 3.12 and Figure 3.13 is plotted in Figure 3.17 below. Figure 3.18 illustrates the segmentation of the index tip pressure profile into the four states.

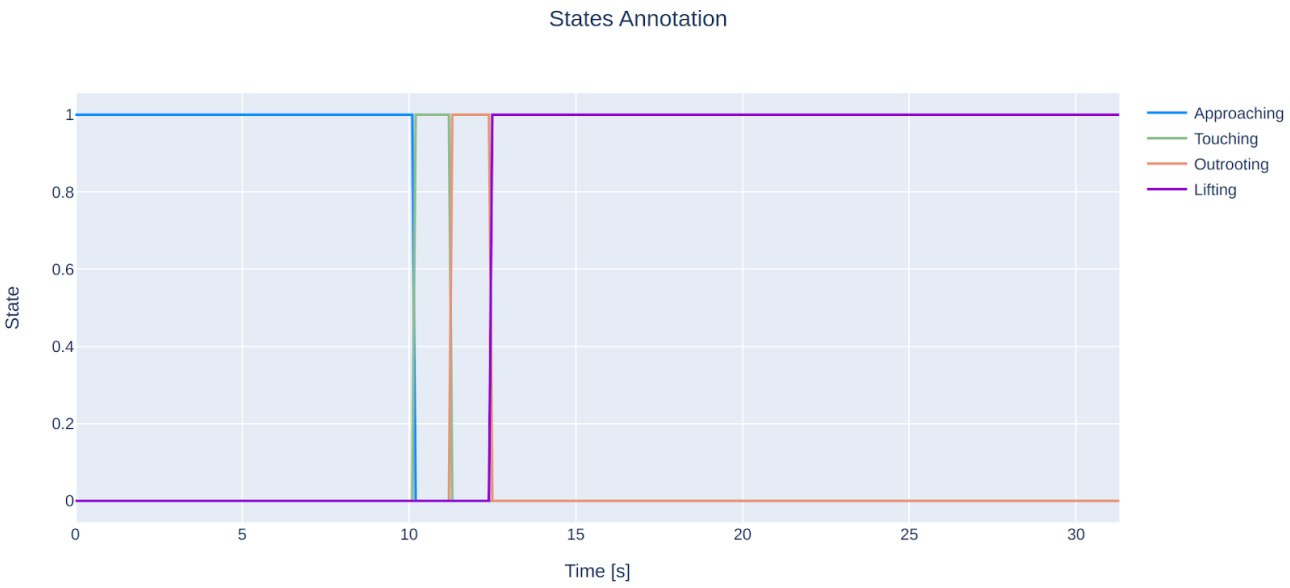

Figure 3.17 State sequence for recording presented in Figure 3.12 and Figure 3.13.

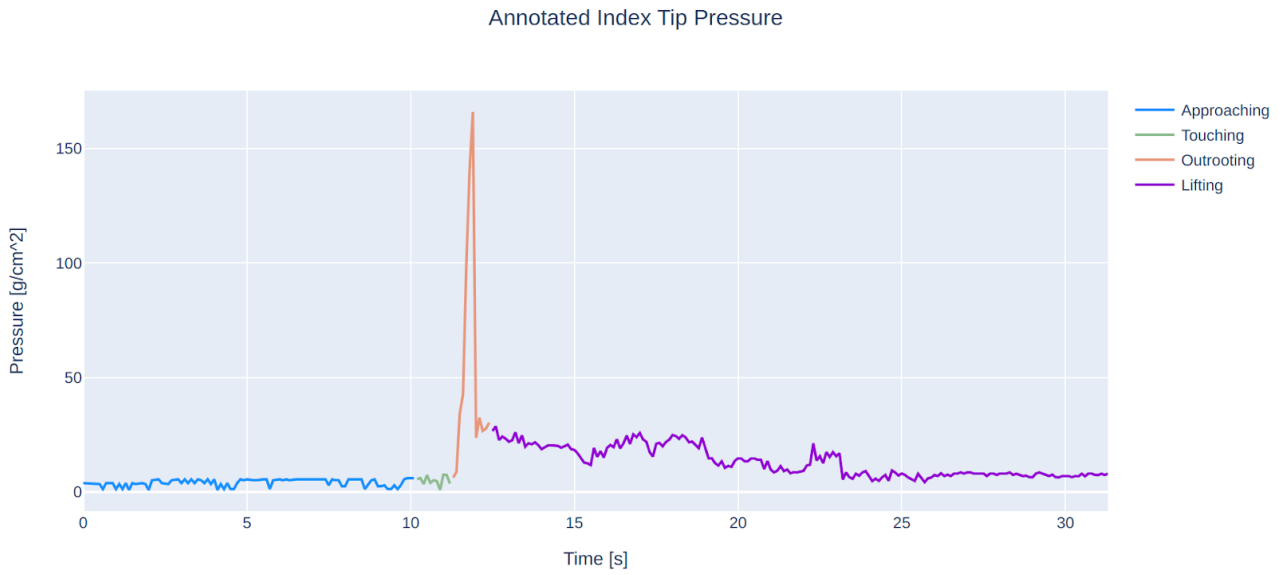

Figure 3.18 Segmentation of the index tip pressure profile.

A total of 30 demonstration attempts were recorded. By examining the expert demonstration recordings, we were able to determine the core characteristics of the motion and force profiles during mushroom picking. The following observations were made:

1) The most prominent motion pattern during picking is that of twisting the mushroom around its principal axis.
2) Tilting the mushroom is employed by the picker only when twisting as far as possible given the human finger configuration allows is not enough to break the mushroom roots which happens in less than 10% of the demonstrations.
3) The force applied during the outrooting phase is between 1.5 and 3N. This is to ensure that no blemishes are caused to the mushroom.

Based on the above observations an array of different Imitation Learning techniques were designed and implemented across different environments, namely a physics-based simulation, a robotic arm with 3D printed mushroom mock-ups and a soft gripper.

4 Behavioural Cloning for Mushroom Picking in Simulated environment

As a first step towards Imitation Learning of mushroom picking, we consider an implementation of an Imitation Learning method for solving a simulated version of the composite manipulation task involving force interactions, published in [92] (© 2023 IEEE).

4.1 Imitation Learning Architecture

Our Imitation Learning Architecture comprises two main components; a Representation Learning (RepL) module that allows for the high-dimensional input to be cast into a latent space to obtain a low dimensional embedding, and the Behavioural Cloning (BC) module which predicts the next action given the embedding of the current observation. We use two different datasets to train our models. The first dataset $D_r = \{T_1^r \dots T_N^r\}$ is a collection of Nrandom trajectories. These are created by sampling random actions by rolling forward a Ornstein–Uhlenbeck process [93]. This dataset is cheap to collect as it does not need and expert agent while still providing important information about the observation structure which is useful to pretrain the RepL module as explained later. Such random trajectory collection has been implemented to stimulate exploration [94]. The second dataset, $D_e = \{T_1^e \dots T_M^e\}$ contains M expert demonstrations, driven by a precise rule-based policy that has direct access to the location and orientation of the target, i.e. the mushroom to be picked and the proprioception measurements of the robot. Each trajectory $T_i = \{(o_k, a_k),\ k = 1 \dots K_i\}$ is a sequence of observation-action pairs as is the case in a standard IL setting. In our case, each observation o is a depth image I, i.e. we only feed one frame per step to the model. This is feasible because gripper actions are defined in terms of linear and angular velocities of the end-effector on the task space as will be explained later. Each depth image is a normalized image, with pixel intensities mapped to $[0,1]$, captured by a camera located close to the end-effector. Thus, we are adopting an eye-in-hand approach.

The RepL module is implemented as a Variational Autoencoder (VAE) [95]. The VAE's encoder, $g(o)$, receives and observation o and outputs two vectors, $g^{\mu}(o)$ and $g^{\sigma}(o)$ which are treated as the mean and the diagonal covariance matrix of normal distribution. $g^{\mu}(o)$ is considered an embedding of o in a latent space captured by the VAE. The decoding step requires sampling the normal distribution and then running the sample through the decoder which produces a reconstruction of the original input $\tilde{o}$. The model loss for each observation sample is defined as:

$$l_{VAE}(\boldsymbol{o}) = (\boldsymbol{o} - \tilde{\boldsymbol{o}})^2 + D_{KL}\left(N\left(g^{\mu}(\boldsymbol{o}), g^{\sigma}(\boldsymbol{o})\right), N(0, \mathrm{I})\right) \tag{4.1}$$

where D_{KL} is the Kullback – Leibler divergence. The overall model is trained by minimizing the mean of the above loss over the entire dataset:

$$L_{VAE} = \frac{\sum_k l_{VAE}(\boldsymbol{o}_k)}{K} \tag{4.2}$$

where K is the total number of observation action pairs in the dataset.

The Behavioral Cloning module comprises a Multi-layer Perceptron that receives as input the embedding $u = g^{\mu}(o)$ and outputs an action prediction $\tilde{a}$. This model is trained to minimize the Mean Squared Error loss:

$$L_{BC} = \frac{\sum_k (\boldsymbol{a}_k - \tilde{\boldsymbol{a}}_k)^2}{K} \tag{4.3}$$

The training of the entire model proceeds in three steps. Firstly, the RepL module is trained on a dataset of random trajectories. The purpose of this is to provide the RepL module with observation instances of sufficient diversity and avoid cases where a new observation lies too far from what the RepL module can properly capture in its latent space. Then, a second training stage of the RepL module takes place. This time it is trained with expert trajectories. Finally, the RepL and BC modules are trained jointly on the expert data. This means that the gradients from the L_{BC} loss flow back to the encoder, updating the weights of $g^{\mu}(o)$.

This staged training approach, illustrated in Figure 4.1, keeps a balance in allocating the representational power of the RepL module to the information contained in the images that is meaningful to predict the actions. During the first two stages, the encoder is trained to minimize L_{VAE}, i.e. to capture information that can be used to reconstruct the image while ensuring that the embeddings are smoothly distributed. In the last stage however, L_{VAE} is disregarded and the encoder is adapted so that it encodes information that is important to predict the optimal action given the observation. Reconstructing the observation and choosing the action are two distinct tasks, potentially requiring the encoder to allocate representational power to different parts. Chen et al. [96] have confirmed how this fact can lead to suboptimal RepL for IL by contrasting the embeddings obtained by using VAEs in a standard classification setting compared to an IL setting, showing that similar embeddings can underpin different actions.

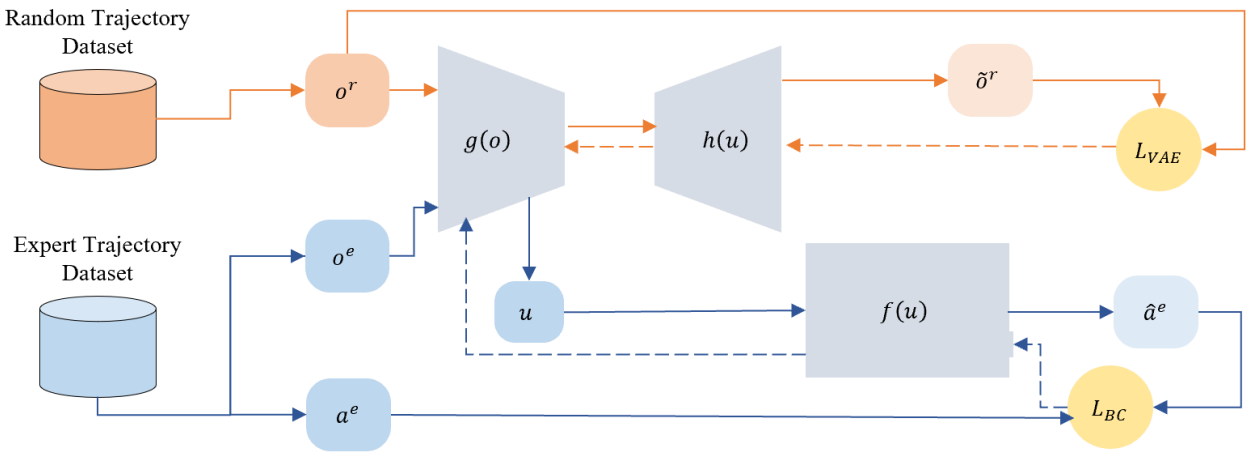

Figure 4.1 Architecture and flow of data (solid lines) and gradients (dashed lines) during training. Orange flow: The pretraining sequence, where the RepL module is trained on random trajectories. Blue Flow: Joint training of both the RepL and the BC modules where the gradients from the L_{BC} flow all the way back to the encoder of the RepL module © 2023 IEEE.

4.2 Experiments

We rely on a custom simulation framework for artificially generating the required training data and for evaluating the imitation-learning pipeline. The simulation is built on top of the PyBullet rigid multibody dynamics physics engine [97], commonly used in the robotics and control community, as the core for modelling the mushroom-root and gripper system. Modelling the mushroom poses a significant set of challenges as it includes elastic body dynamics, material failure mode and intricate complex collision and friction interactions which are notoriously hard to accurately simulate and in a reasonable execution time. Our mitigation strategy relies on a surrogate 4 DOF mass-spring model comprised of a spherical and prismatic joint kinematic structure for the mushroom root motion, which assumes linear isotropic elastic material properties of a solid cylinder for capturing the dominant system dynamics. For computing the stress threshold necessary for material failure and the consequent mushroom detachment from the ground von Mises criterion is utilized. Lastly, instead of incorporating the complete robot model we only include the gripper by defining a fixed constraint in the flange frame in 3D space, as also suggested in [98]. The end-effector is a model of the gripper of the Panda robot by Franka Emika GmbH, one of the most widely used collaborative robots in academia and industry alike. Figure 4.2 shows a snapshot of the simulation environment.

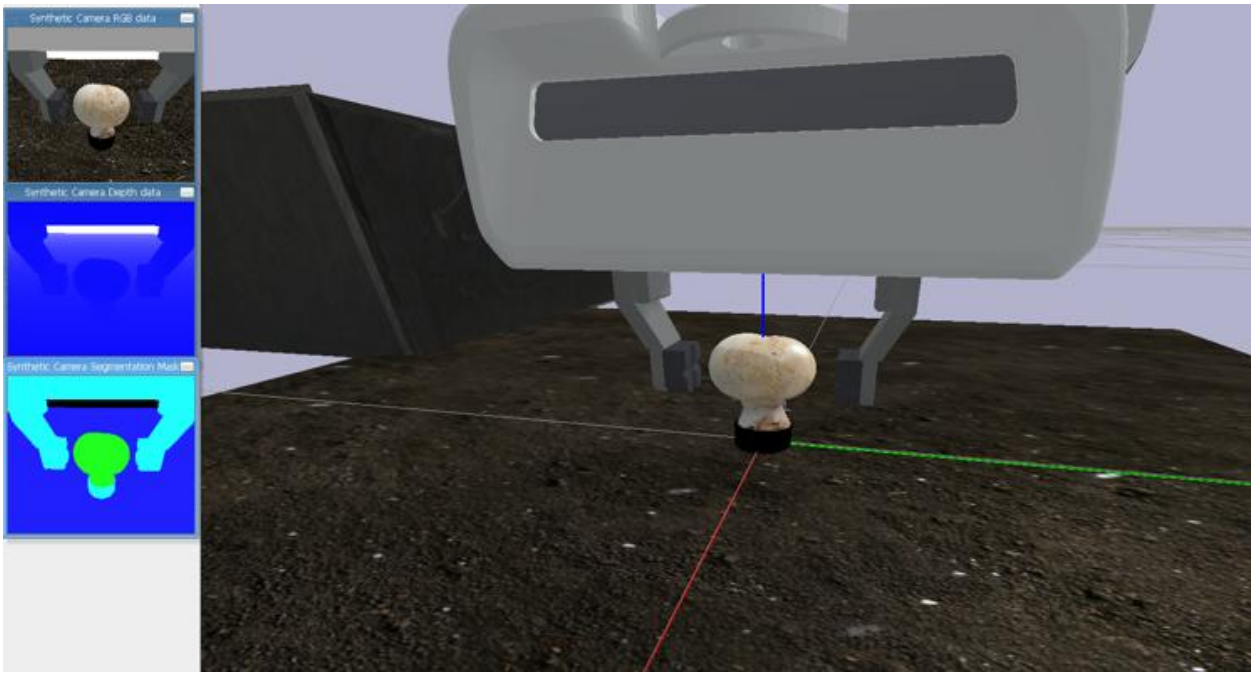

Figure 4.2 PyBullet debug window rendering for the gym simulation environment for mushroom harvesting © 2023 IEEE

The stiffness parameters for the 4DOFs of the root model are defined as follows assuming linear isotropic properties and given the Young's modulus E and the Poisson ratio v. The axial, rotational and bending stiffness components are given as follows:

$$K_{ax} = \frac{EA}{l} \tag{4}$$

$$K_{rot} = \frac{GJ}{l} \tag{5}$$

$$K_{bend} = \frac{3EI}{l} \tag{6}$$

where G is the shear modulus is calculated by:

$$G = \frac{E}{2(1+v)} \tag{7}$$

and l is the length of the stem, A the axial surface area, J the second polar moment of area, and I the second moment of area.

A critically damped component is also defined for all 4 DOFs for ensuring stability. The axial and shear stress component under axial force p, bending force f and twisting torque τ are computed by:

$$\sigma_{ax} = \frac{p}{A} + \frac{flr}{I} \tag{8}$$

$$\tau_{zx} = \frac{\tau r}{J} \tag{9}$$

where r is the radius of the mushroom stem and τ. The von Mises stress factor is given by:

$$\sigma_{VM} = (\sigma_x^2 + 3\tau_{zx}^2)^{1/2} \tag{10}$$

Failure and therefore mushroom yielding will occur when the von Mises stress factor exceeds the yielding stress factor S_y, namely when $\sigma_{VM} > S_y$. The Young's modulus, Poisson ratio and Yielding stress factor where selected based on harvesting experiments carried out, also confirmed by literature [99]. The selected values are provided in Table I.

Table 4.1 Key Simulation Parameter Values

Parameter	Value range
Young's Modulus	0.05 – 0.15 MPa
Poisson Ratio	0.3 – 0.5
Yielding stress	14,000 kPa

The simulated mushroom dynamics ensure a highly realistic outrooting behaviour. Concretely, by setting a squeezing force threshold value at 1N, the mushroom cannot be outrooted by pulling alone. An attempt to pull the mushroom with this force would result in the mushroom slipping away from the fingers and thus the outrooting attempt would fail. Such a scenario is illustrated in Figure 4.3. On the other hand, with the same force threshold the mushroom can be outrooted successfully if a twisting motion is applied first which would result in increasing the stress on the roots according to the equations above. This is illustrated in Figure 4.4 which shows an outrooting attempt accomplished by the trained agent.

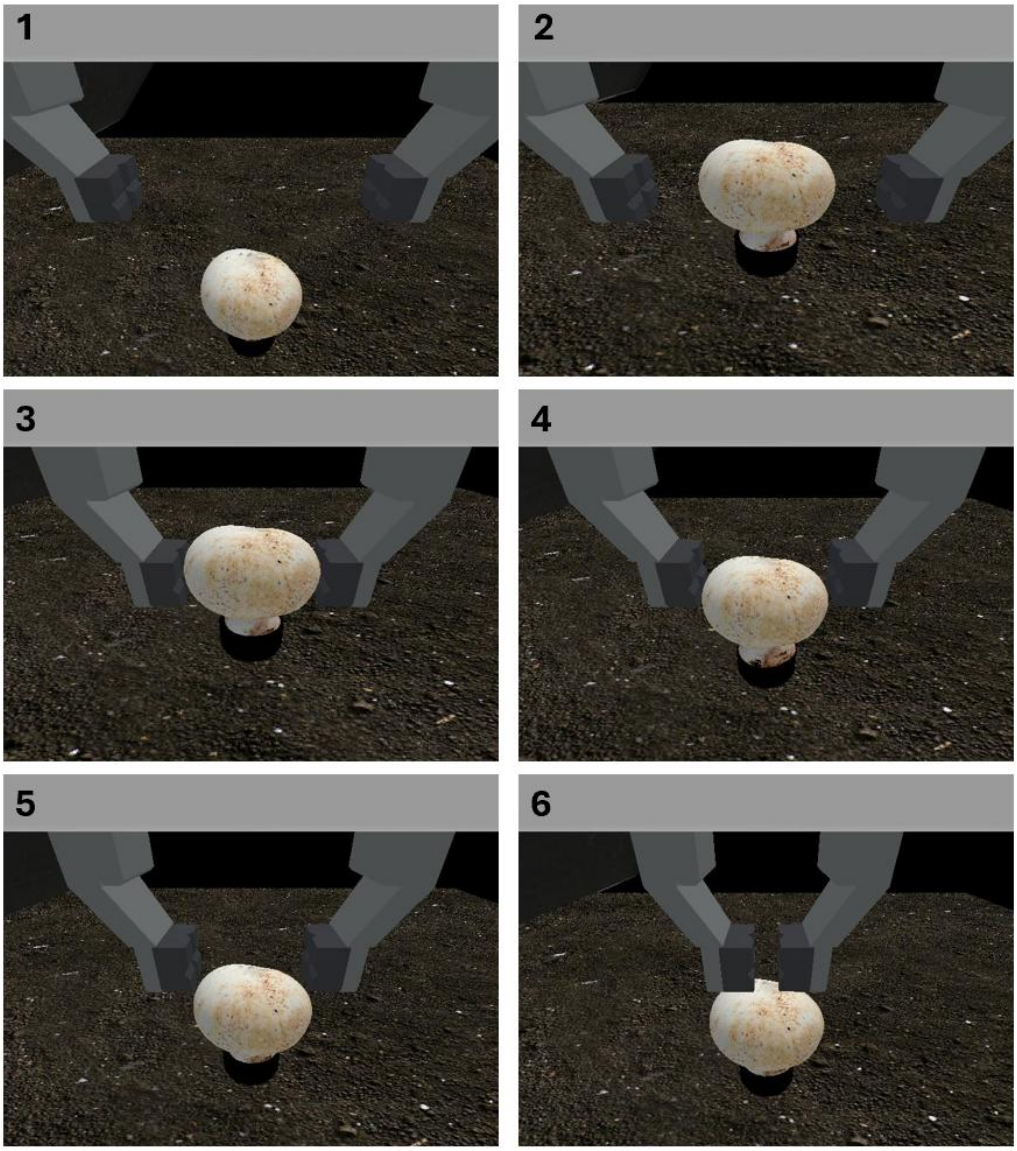

Figure 4.3 A sequence of a failed outrooting attempt; (1) the gripper starts at a random position, (2) moves fingers around the mushroom, (3) grasps the mushroom, (4-5) pulls the mushroom upwards without twisting resulting in slippage, (6) mushroom completely slips away

In our simulated mushroom harvesting setting, the observation o that the IL agent has access to is purely a grayscale or a depth image from a camera mounted close to the robot gripper. The action a the agent produces given an observation is an 8-dimensional vector; the six first elements of the vector contain the desired linear and angular velocities of the gripper while the last two elements determine the gripper control. Concretely, the seventh element is a binary value that triggers the gripper fingers to close while the last element is the desired force the gripper fingers should reach for the fingers to stop moving. This action representation,

particularly the part about the gripper, follows [98] as it allows for a direct transfer to a real robot. However, it is not without complications since it leads to a hybrid action space, i.e., containing both continuous and binary elements. Thankfully, we've observed that our IL approach is sufficiently stable to allow for a simple thresholding of the continuous relaxation of the binary value without hurting performance. Thus, our approach treats the action as continuous and then, during episode rollouts the binary element is thresholded at 0.1 to obtain a binary value.

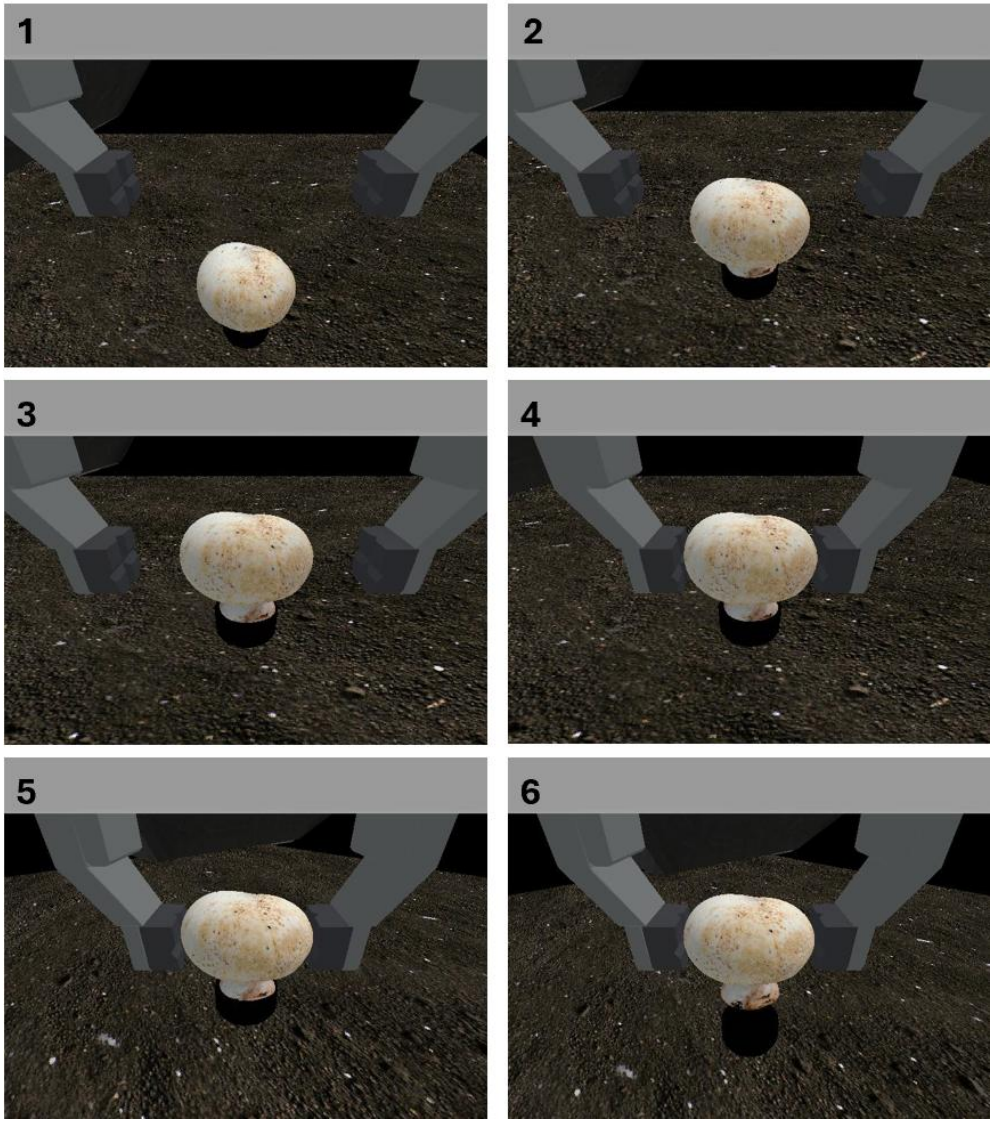

Figure 4.4 A sequence of a successful episode rollout by the trained agent. (1) the gripper starts at a random position, (2-3) moves fingers around the mushroom, (4) grasps the mushroom, (5) twists, and (6) pulls the mushroom upwards. © 2023 IEEE

Figure 4.4 provides an illustration of a mushroom harvesting sequence accomplished by a trained IL agent on grayscale images. Fig 5. illustrates the respective trajectory, also showing the trajectory of the expert demonstrator. It is shown that the agent closely mimics the expert, and it also performs the task slightly faster. The gripper z-position is telling; the gripper is lowered towards the mushroom, it stays in that low position for a time until the gripper closes, as seen by the squeezing force increase, and then the gripper moves up, pulling the mushroom with it.

4.3 Results

To assess the robustness of our approach, we have conducted experiments in both partially and fully randomized versions of the environment. In the partially randomized version, the size of the mushroom as well as the extrinsic and intrinsic parameters of the camera do not change across episodes. Thus, in this version, the only source of randomness between episodes was the starting point of the gripper. In contrast, in the fully randomized version of the environment, we randomly varied the extrinsic parameters of the camera, namely its yaw and pitch angle, its distance relative to the gripper as well as the intrinsic 2D coordinate frame centre. We specifically applied a randomness factor of ±5% across each of these parameters compared to the nominal values. We also varied the size of the mushroom by ±10%.

Figure 4.5 illustrates the trajectories followed and forces applied by the expert and the trained agent on the same simulation. As seen, the trained agent is able to closely mimic the expert albeit with a slight delay in terms of solving the episode. It is worth noting that the forces achieved are almost identical, even though the force information is not provided as an input to the policy network.

Furthermore, we tested our approach with two different types of images as observations. In the first case, we used grayscale images as the single source of information while in the second case we used artificially noise-corrupted depth maps. Adding noise to the depth maps is crucial to mimic a realistic setting. To produce the artificial noise, we used an empirically derived technique, where random noise is added close to the edges of the image, detected via a standard edge detection technique. Figure 4.6 compares a pair of an RGB and a depth map from an actual RealSense camera with a similar pair obtained by our simulation environment after artificial noise has been added to the depth map. Table III and IV summarize the results across all different tests. Success rates were measured on 50 different episode rollouts.

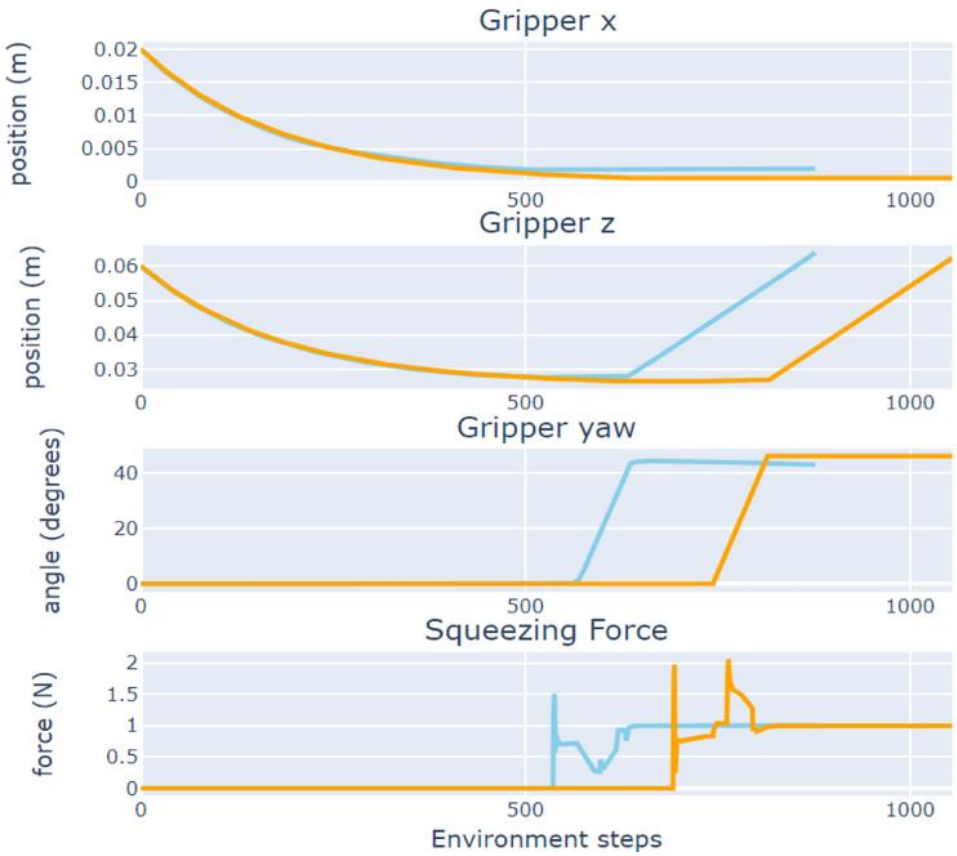

Figure 4.5 The trajectories (x, z position and yaw angle) and the squeezing forces observed on the gripper fingers of the trained agent (cyan) and the expert demonstrator (orange). © 2023 IEEE

Table 4.2 Results with Grayscale Image as Input © 2023 IEEE

Approach	Randomization	Success rate
Expert Only	Partial	90%
Expert Only	Full	58%
Random + Expert	Partial	96%
Random + Expert	Full	78%

As seen above, our approach successfully reproduces the mushroom harvesting sequence >90% of the time in the partially randomized environments. Pre-training on random trajectories can sustain a 78% success rate even in the case of significant camera perturbation and mushroom size variation, where pure training on expert demos reaches 58%. The high success rates in the corrupted depth map case show significant versatility and resilience to noise, particularly

considering the noise severity as shown in Fig 6. Pre-training did not improve the depth map agents while it also caused a slight drop of performance in the non-randomized case.

The computational cost of our approach is exceptionally light; the entire pipeline (RepL and BC) is trained in two passes, with about ~80,000 images of size 61x89 pixels per pass in under an hour on a Laptop GPU (GTX 1650 Ti). Inference takes less than 1.5ms on a Laptop CPU (Intel i5).

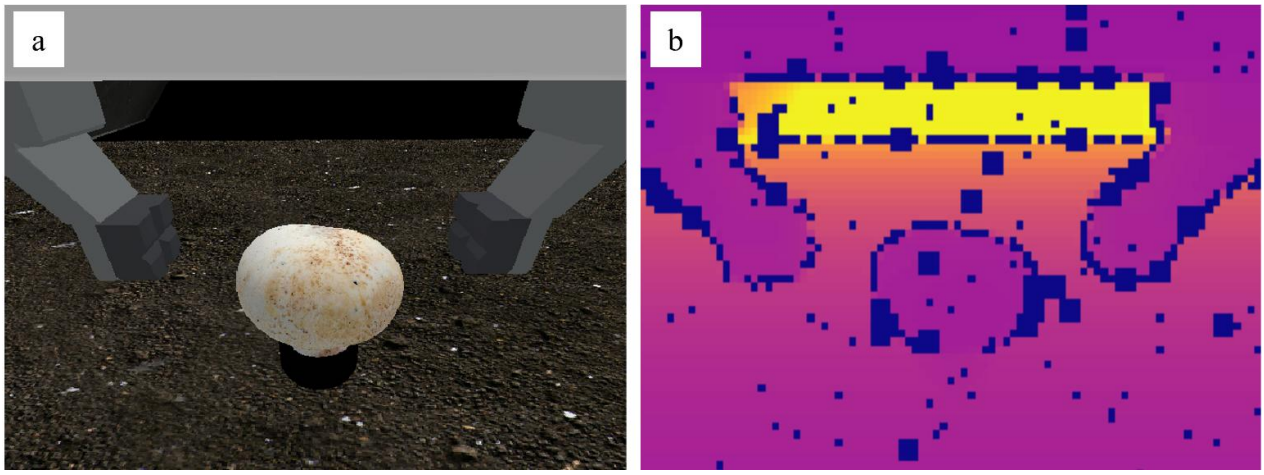

Figure 4.6 (a) RGB image, (b) depth map with artificial noise. © 2023 IEEE

4.4 Discussion

The approach described above demonstrates remarkable computational efficiency, particularly in its training time and computational requirements, enabling real-time inference on a standard Laptop GPU. This is a crucial feature that allows our pipeline to be used on local controllers comprising computational power in the order of single-board computers.

In terms of data efficiency, although our approach reaches acceptable levels thanks to leveraging random sampling for pretraining the auto-encoders, there is still significant room for improvement as there is a need for tens of thousands of steps which correspond to over 100 episodes. Although these numbers are easy to obtain in a simulation environment, it can be challenging to achieve in real settings.

Our approach's ability to solve a robotic manipulation task with force constraints directly from visual stream with such a simple architecture illustrates that BC holds significant promise particularly when compared with RL methods. The latter typically require millions of steps to attain sufficient success rates and requires extended exploration phase where the behaviours

rolled out by the agent cannot be considered safe [100]. Thus, even though our approach requires a considerable amount of expert data, fulfilling this requirement comes at a lower cost in terms of effort compared to that of ensuring safety for an exploring agent in a physical manifestation such as a robotic manipulator. Nevertheless, it should be noted that Offline Reinforcement Learning techniques can be highly competitive due to the existence of the critical state before grasping [101]. This, however, comes at the expense of having to carefully craft a reward function. Such a requirement is not straightforward to fulfill beyond simulated environments.

5 Behavioural Cloning for mushroom picking with a rigid gripper

Building on the previous architecture, we adjust transfer the approach to a real robotic arm with a rigid gripper

5.1 Imitation Learning Architecture

In contrast to [102], this end-to-end training approach, illustrated in Figure 5.1, does not require separate training of a Representation Learning and Behavioral Cloning modules. No representational power is allocated in reconstructing the observation; instead the encoder learns embeddings that are directly relevant to action prediction following the lessons of Chen et al. [96].

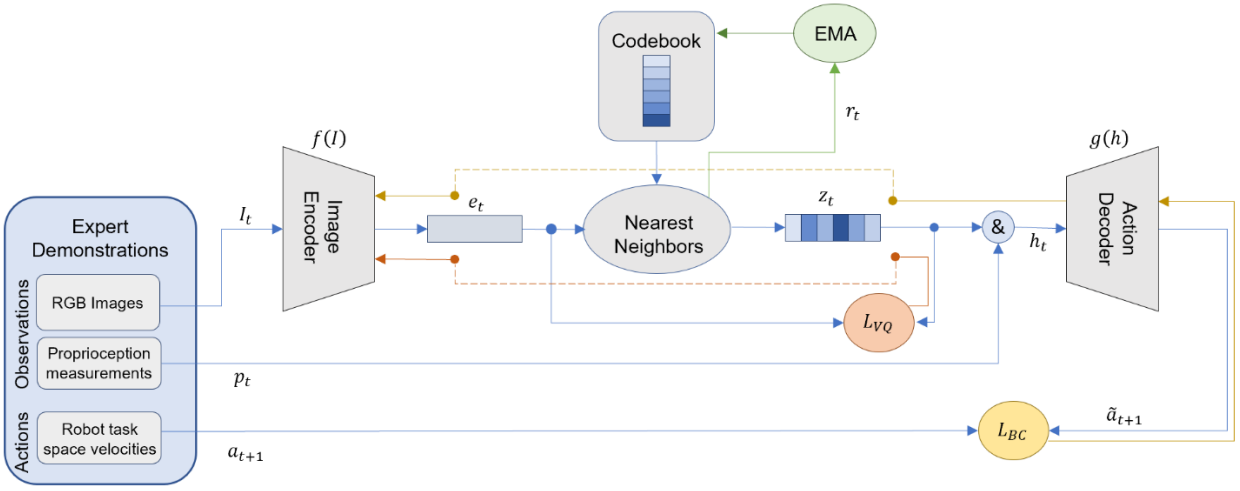

Figure 5.1 Architecture and flow of data (blue lines) and gradients (orange and yellow lines) during training. Dashed lines indicate gradient copying to accommodate non differentiable modules. The ampersand (&) symbol indicates concatenation.

Each action a is 5-dimensional vector; the first four elements contain the desired linear velocities v_x, v_y, v_z and the angular velocity ω_{yaw} of the gripper while the last element is a binary value that triggers the gripper fingers to close. The $\omega_{pitch}, \omega_{roll}$ angular velocities are set to zero throughout the task. This action representation, i.e. the task-space based velocities and gripper closing trigger follows [98] and it allows for straightforward transfer of the learning pipeline to different robotic arms. However, it is not without complications since it leads to a hybrid action space, i.e., containing both continuous and binary elements. Thankfully, we've observed that our IL approach is sufficiently stable to allow for a simple thresholding of the continuous relaxation of the binary value without hurting performance. Thus, our approach treats the action as

continuous and then, during episode rollouts the binary element is thresholded to obtain a binary value. Representing the action in terms of task-space velocities means that even one-step observation-action pairs are sufficient to learn a suitable policy. In our tests with Transformer-based models we also utilize multi-step sequences of observation-action pairs.

The Image encoder module f, parametrized by $\boldsymbol{\theta}_f$, applies a sequence of convolutional residual blocks which downsample each image sample $\boldsymbol{I}_t$ and then flattens the result into an embedding $\boldsymbol{e}_t = f(\boldsymbol{I}_t)$. The embedding $\boldsymbol{e}_t$ is passed through the Vector Quantization module q which yields a quantized embedding $\boldsymbol{z}_t = q(\boldsymbol{e}_t)$. This quantization process uses a learnable codebook of vectors $\boldsymbol{C} = \{\boldsymbol{b_i},\ i = 1 \dots N\}$. Each block of length B of e_t is matched with its nearest neighbor within $\boldsymbol{C}$ in the Euclidean distance sense, leaving a residual r_i. The residuals are used to update the codebook using and Exponential Moving average scheme:

$$\boldsymbol{b}_i^{\tau} = \boldsymbol{b}_i^{\tau-1} * \gamma + \boldsymbol{r}_i(1-\gamma) \tag{5.1}$$

where γ is an update coefficient between 0.9 and 1. To encourage the Image Encoder to produce embeddings with blocks that are close to the codebook vectors we implement a commitment loss:

$$L_{VQ}(\boldsymbol{e}_t, \boldsymbol{z}_t) = (\boldsymbol{e}_t - \ sg[\boldsymbol{z}_t])^2 \tag{5.2}$$

where $sg[\cdot]$ denotes stopping the gradient flow to account for the fact that the operation of finding the near neighbor is not differentiable per se. In practice, this means that in computing the gradient of L_{VQ}, z_t is considered independent of the θ_f parameters.

The Action Decoder module comprises a sequence of fully connected layers and outputs an action prediction $\tilde{a}$. This model is trained to minimize the Mean Squared Error loss:

$$L_{BC}(\boldsymbol{a}, \tilde{a}) = (\boldsymbol{a} - \tilde{\boldsymbol{a}})^2 \tag{5.3}$$

Different model alternatives can be implemented both for the Image Encoder and the Action Decoder. In our experiments the Image Encoder is fixed but we test with a Transformer-based Action Encoder instead of the dense layer network as well.

The training of the entire model proceeds in an end-to-end fashion where both modules, are trained jointly. The gradients from the expert data. The gradients from the L_{BC} loss flow back through the Action Decoder and the Image Encoder while those of L_{VQ} are only flowing through the latter. The L_{BC} gradients are side-stepping the discontinuity induced by the quantization module by being copied directly from the Action Encoder to the Image Decoder.

Introducing a VQ module between the encoder and the decoder is shown to significantly enhance the performance of the IL agent as explained in Section IV. This is possibly because it has a stabilizing effect against image variations induced by factors like differences in lighting across data collected during demonstrations and during actual trials.

5.2 Environment

The environment used to test our IL approach was designed to be representative of the actual mushroom harvesting process while staying practical in terms of the equipment and consumables required. Towards this end we designed and 3D-printed five different mushroom mock-ups with Polylactic Acid (PLA) material. Each mock-up had different stem length, cap size and orientation as shown in Figure 5.2.

Figure 5.2 Top- and side-view of the five mushroom models used in the picking experiments.

Our experimental setup encompasses the xArm 6 Robotic arm by Ufactory and a RealSense D435f camera by Intel. The camera was mounted above the end-effector on a custom 3D-printed mount, as seen in Figure 5.3. Although the camera is capable of capturing depth information, we did not use depth maps in our approach as depth sensing quickly becomes unreliable at low

distances from the image surface, which is bound to be the case in an Eye-In-Hand set-up. Most importantly, our focus was to explore the IL capabilities with sensorial streams that are straightforward and affordable to capture, requiring no calibration.

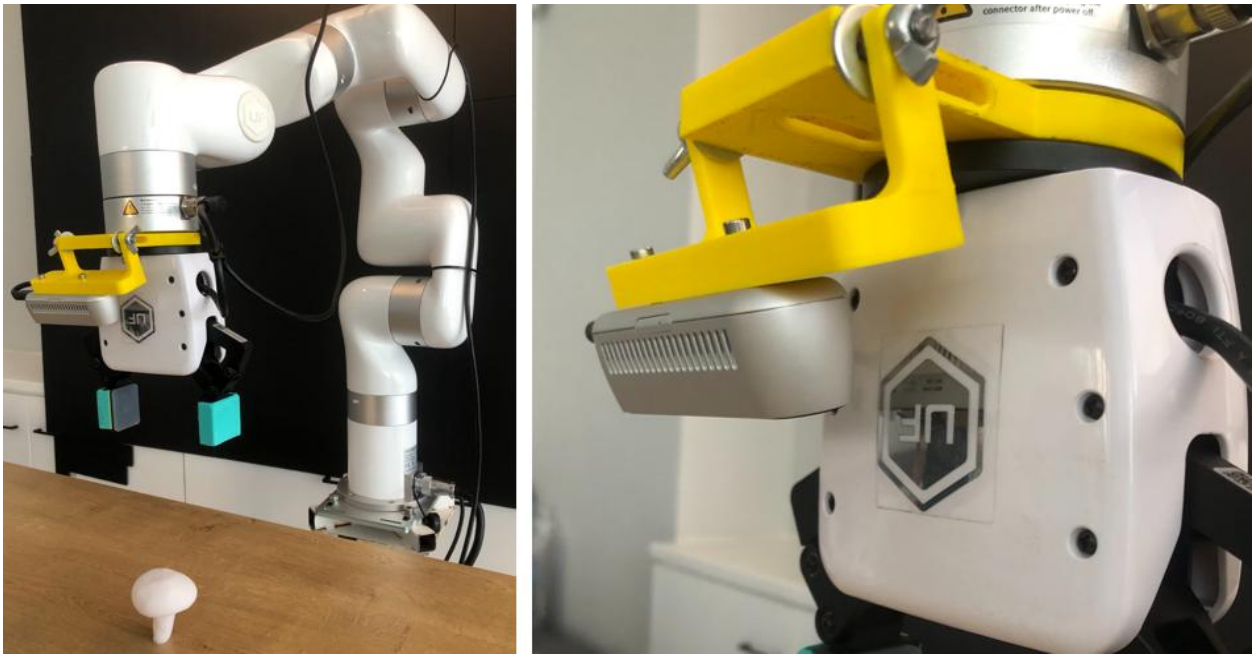

Figure 5.3 Our experimental setup; Left: the robotic manipulator (xArm6 by Ufactory) and an example of the mushroom 3D-printed mock-ups. Right: The 3D printed mount and the camera (Realsense D435f) used for imaging

Although our setup involves a real robot and realistic mushroom mock-ups it entails two obvious differences compared to an operational mushroom harvesting setting; (i) the PLA 3D-printed mushroom models are stiffer compared to their real counterparts and (ii) human mushroom harvesters must often pick mushrooms amongst dense clusters of other mushrooms rather than individually.

The stiffness discrepancy is considered minor particularly under the low forces employed by human pickers, which induces negligible deformation on real mushrooms. To further tackle this difference, we regulate the force exerted by the gripper to 5N and we attach the mushroom to the table using a putty-like self-adhesive on the bottom of the stem, emulating the root. The amount of self-adhesive has been chosen through experiments to achieve a key characteristic of the mushroom harvesting process; attempting to “outroot” the mushroom mockup simply by pulling upwards is impossible under the force-regulation condition, as the mockup slips from the fingers, but it is feasible by twisting it first, much like the situation with the real mushrooms. This

force regulation aspect introduces a significant challenge that distinguishes our setup from conventional pick-and-place setups routinely used for benchmarking IL approaches.

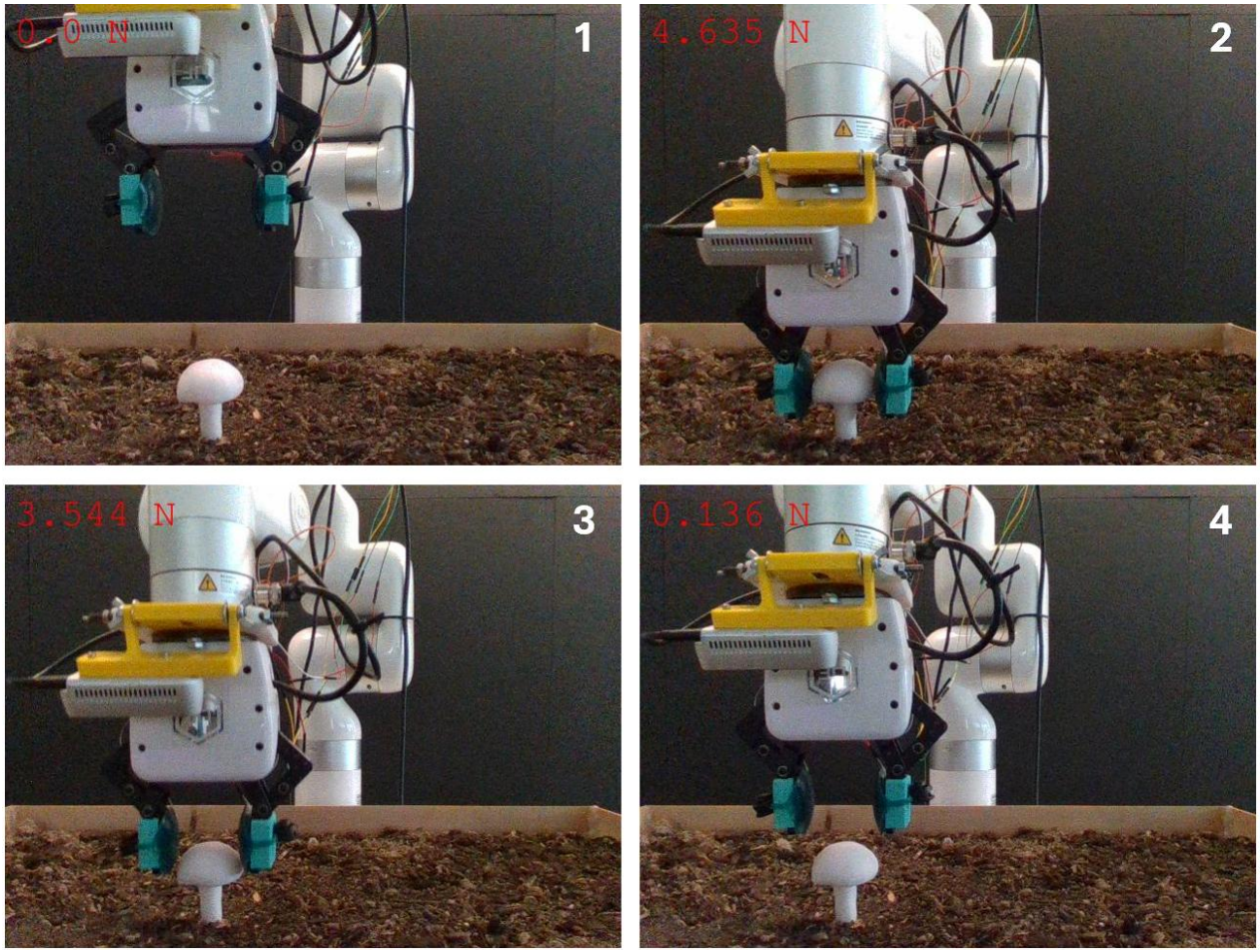

Figure 5.4 Failed attempt without twisting; (1) the gripper starts at the random position, (2) moves and grasps the mushroom mock-up, (3) starts pulling upwards causing mushroom mock-up to slip, (4) mushroom mock-up completely slips away

Harvesting amongst dense clusters involves a higher-level cognitive process like selecting the mushroom that must be picked based on expert knowledge and factors like mushroom size and surrounding mushroom topology. We view such a process to lie beyond the scope of our work. We focus on enabling learning of the motion combinations required to successfully harvest a single mushroom under force constraints and from ultra simple visual and proprioceptive input. We consider addressing this task to be the crucial element for automating the overall process.

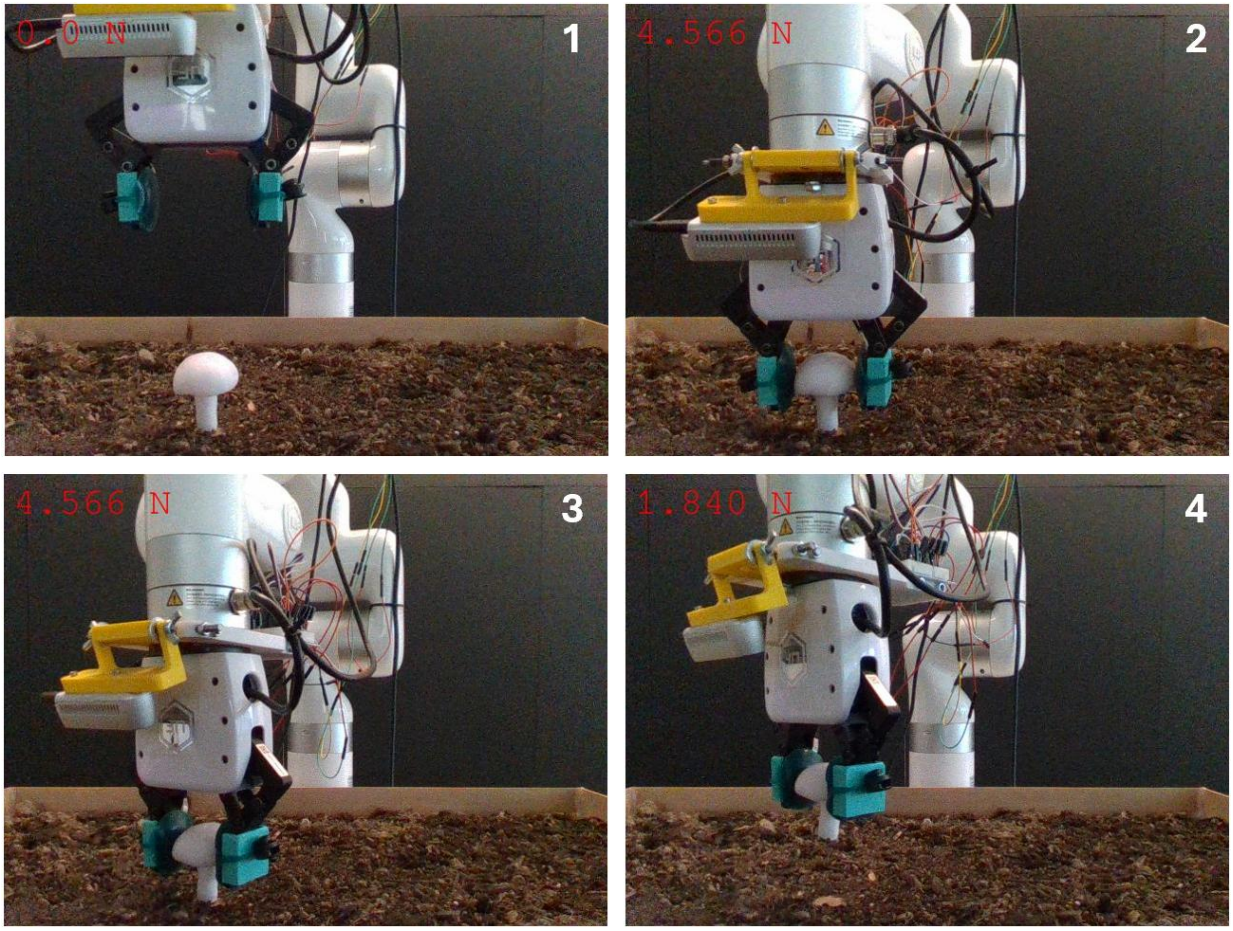

Figure 5.5 Successful attempt with twisting; (1) the gripper starts at the random position, (2) moves and grasps the mushroom mock-up, (3) twists the mushroom mock-up, breaking some of the adhesive bonds, (4) pulls mushroom mock-up upwards successfully

5.3 Experiments

We collected 400 demonstrations of mushroom picking using the robotic arm, by employing a precise, rule-based policy that has direct access to the 6DoF pose of the mushroom to be picked. Each trajectory was 17-20s long, thus the entire set amounted to a total of ~2hrs worth of training data. We used four of the mushroom models in these demonstrations, i.e. 100 demonstrations for each mushroom. The fifth mushroom has left aside to test the generalization ability to different mushroom heights and cap shapes. Each trajectory was collected with the gripper starting at a random position relative to the mushroom within a square of 120mm side on the xy-plane and with random mushroom orientation. The starting z-position of the gripper was kept fixed. The lighting conditions were intentionally not strictly dictated to probe the resilience of the approach to lighting perturbations.

We benchmark our core approach of convolutional residual blocks as an Image Encoder and a dense block of fully connected layers (ConvVQDense) against two state-of-the-art models; the Behavioral Transformer (VanillaBeT) [40], and a reduced version of the Diffusion Policy model (ConvDP) [46], where the Diffuser is reduced to 65M parameters, and the ResNet18-based Representation Layer is substituted by the simple convolutional residual blocks used in our pipeline. This adaptation was necessary to make the model practical on the same setup that the rest of the benchmarks are running on. In order to establish the merit of introducing a Vector Quantization module in between the encoder and the decoder, we also test five additional variants; an alternative BeT implementation where the initial Representation Module is changed with a trainable Image Encoder identical to the one of our core approach with (ConvVQBeT) and without a VQ module (ConvBeT), a Diffusion Policy model including a VQ module (ConvVQDP), a variant of our core approach where the VQ has been removed (ConvDense) and, finally, an implementation of our approach with a Variational Encoder (ConvVEDense) instead of a VQ module. In the latter case we use an Encoder including a Kullback – Leibler Divergence loss in lieu of the L_{VQ} loss as in the case of Variational Autoencoders (VAEs) [95] to further test the performance of the VQ against a competing embedding projector. In the cases without the VQ or the Variational Encoder, a fully connected layer is added to transform the dimension of the Image Decoder output in a dimension appropriate for the Action Decoder so that these two models' architectures are kept identical across all variants.

Our proposed architecture, ConvVQDense, is fairly lean, comprising to convolutional residual blocks with 20 and 10 3x3 filters respectively, applied with stride 2 for downsampling, for the Observation Encoder and two Fully Connected layers of size 360 each for the action Decoder. The VQ module in between has a codebook of 128 vectors of size 10 and outputs a flattened embedding of size 2940.

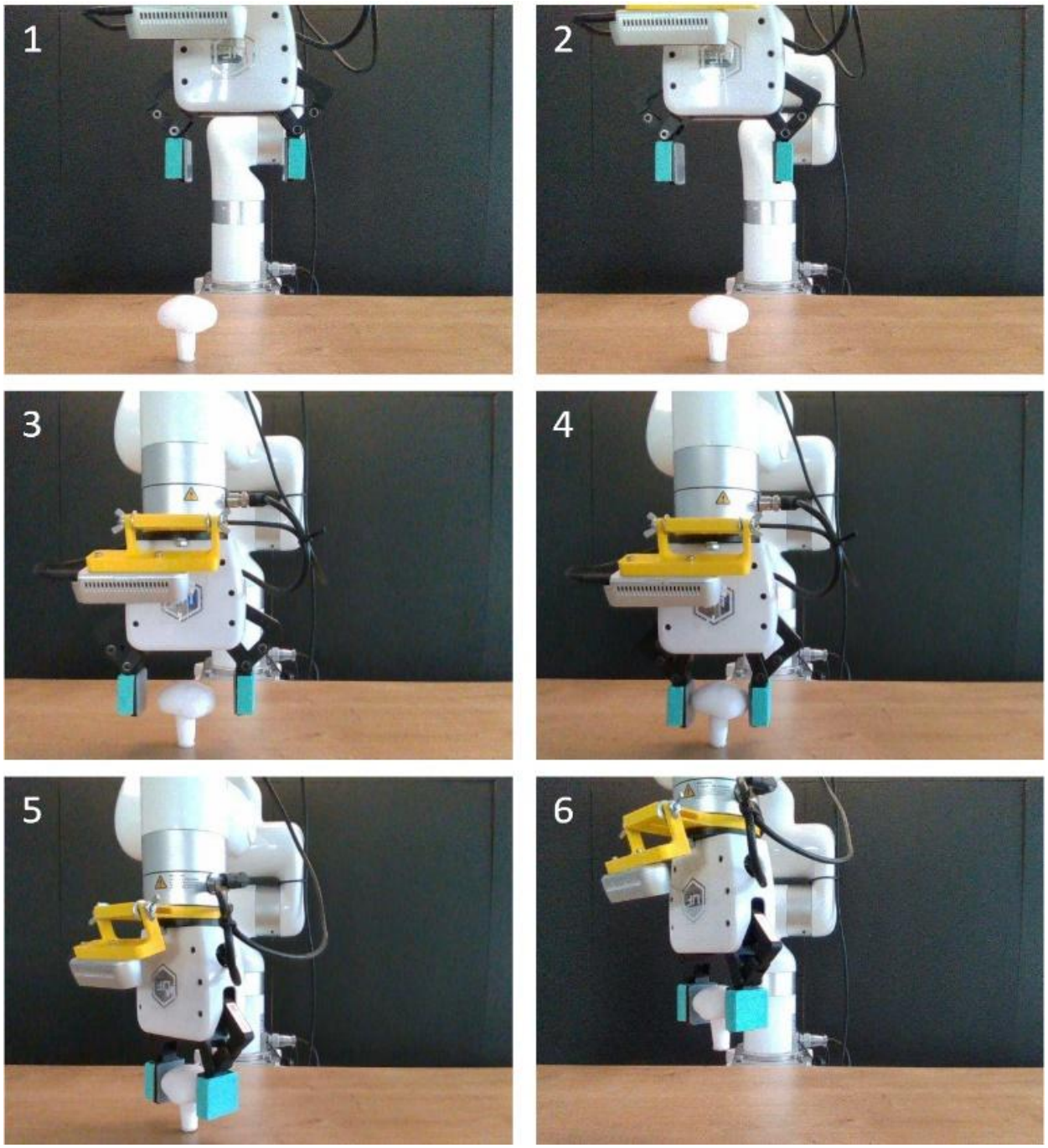

Figure 5.6 A sequence of a successful episode rollout by the trained agent. (1) the gripper starts at a random position; (2) moves above the mushroom; (3) reaches down; (4) grasps; (5) twists, breaking the adhesive bonds; and (6) pulls the mushroom upwards.

Figure 5.6 provides an illustration of a mushroom harvesting sequence accomplished by a trained IL agent with the ConvVQDense architecture. Figure 5.7 illustrates the respective trajectory, also showing the trajectory of the expert demonstrator with a similar starting location for the gripper.

It is shown that the agent closely mimics the expert solving the task in approximately the same level of steps.

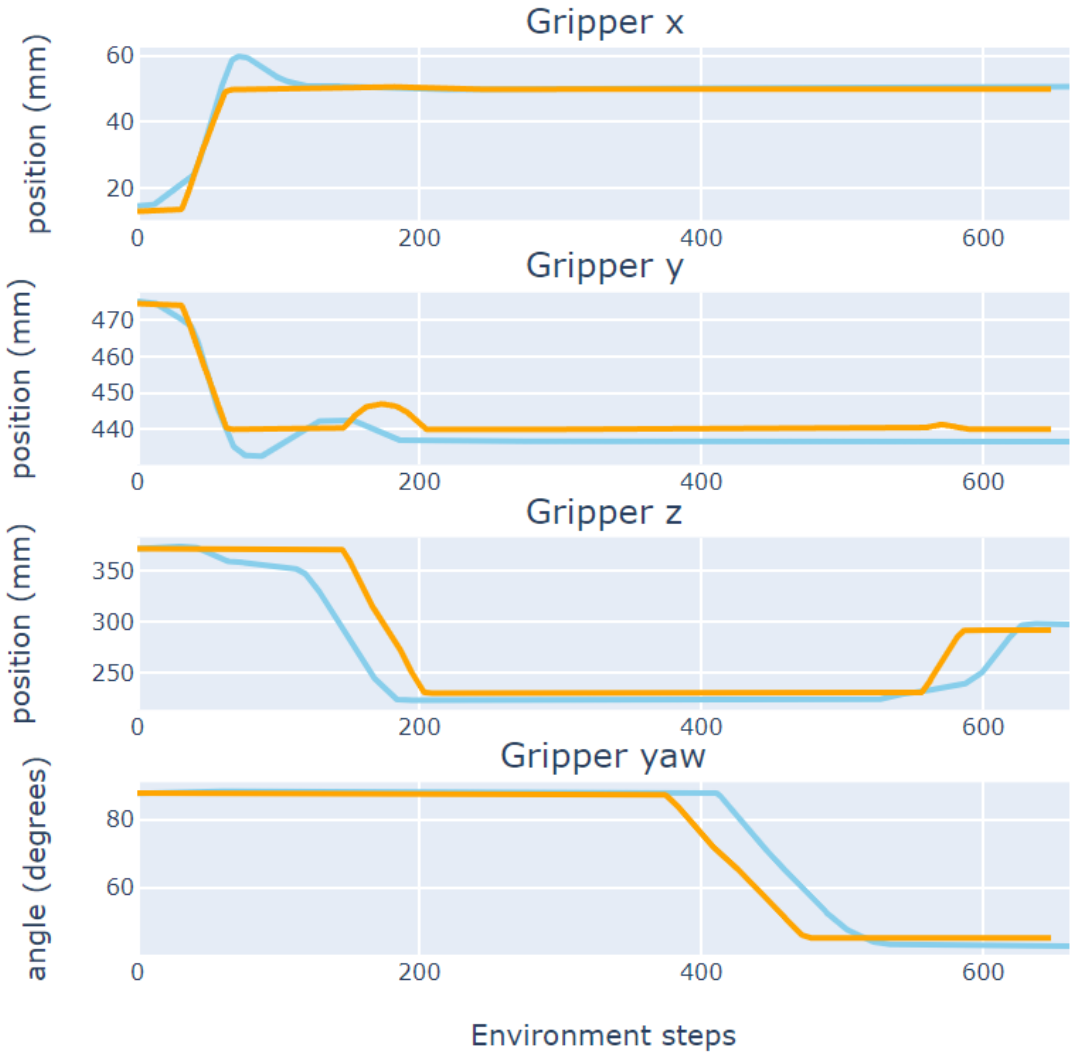

Figure 5.7 The trajectories (x, y, z position and yaw angle) observed on the gripper fingers of the trained agent (skyblue) and the expert demonstrator (orange).

5.4 Results

All six models described above were trained on the same expert demonstration dataset which comprised 244,000 steps in total. To accommodate the same control frequency across all models, despite the large variation in inference speed, we use a 3Hz control loop. This was necessary since inference for Diffusion Policy models exceeded 300ms per step. Since video and proprioception measurements were captured at ~30Hz, each trajectory was split into 10-13 subsampled trajectories. Approach-specific parameters such as action bins for the BeT were selected based

on Bayesian Optimization using the Optuna framework [103]. All models operated on single-frame inputs except for the Diffusion Policy models which got the last two consecutive frames as input. The VanillaBeT model uses a pretrained, frozen Resnet18 [104] as an Image Encoder, following [40]. The resulting models were evaluated over 50 trials, 10 for each of the different mushroom models. Episodes that ended with the gripper above 260mm from the table, holding the mushroom in less than 30s from the beginning of the episode were deemed successful. The success rate statistics were gathered by taking the average and the standard deviation across 3 different training seeds for each model pipeline.

The overall success rate of each model architecture over the entire set of trials is presented in Table I. Figure 5.8 outlines a breakdown of the performance for each mushroom, averaging across the three different training seeds for each model architecture.

Table 5.1 Comparison of our approach (ConvVQDense) against state-of-the-art models and variations thereof

Architecture	VQ/VE	Action Decoder	Success rate
VanillaBeT [40]	None	BeT	0.31±0.18
ConvBeT [46]	None	BeT	0.39±0.19
ConvVQBeT	VQ	BeT	0.45±0.17
ConvDP	None	Diffusion	0.44±0.19
ConvVQDP	VQ	Diffusion	0.57±0.12
ConvDense	None	Dense	0.61±0.18
ConvVEDense	VE	Dense	0.56±0.12
ConvVQDense	**VQ**	**Dense**	**0.90±0.08**

As seen above, introducing the VQ module invariably increases the performance of the respective model architecture, leading to an improvement of 13% for the Behavior Transformer, 30% for Diffusion Policy and 48% for the Dense network. Interestingly, in the latter case, substituting a VE instead of VQ, leads to slight decrease on the overall performance. It is also interesting to note that VQ also has a stabilization effect across training seeds as seen in the significantly smaller error margins for ConvVQDP and ConvVQDense. Overall, the VQ module ultimately enables a 90% success rate with a model pipeline of exceptionally low computational cost, using a sequence

of simple Fully Connected layers instead of Transformers or Diffusion models. Our pipeline requires just over 2ms for each inference step compared to >300ms for Diffusion Policy models [46].

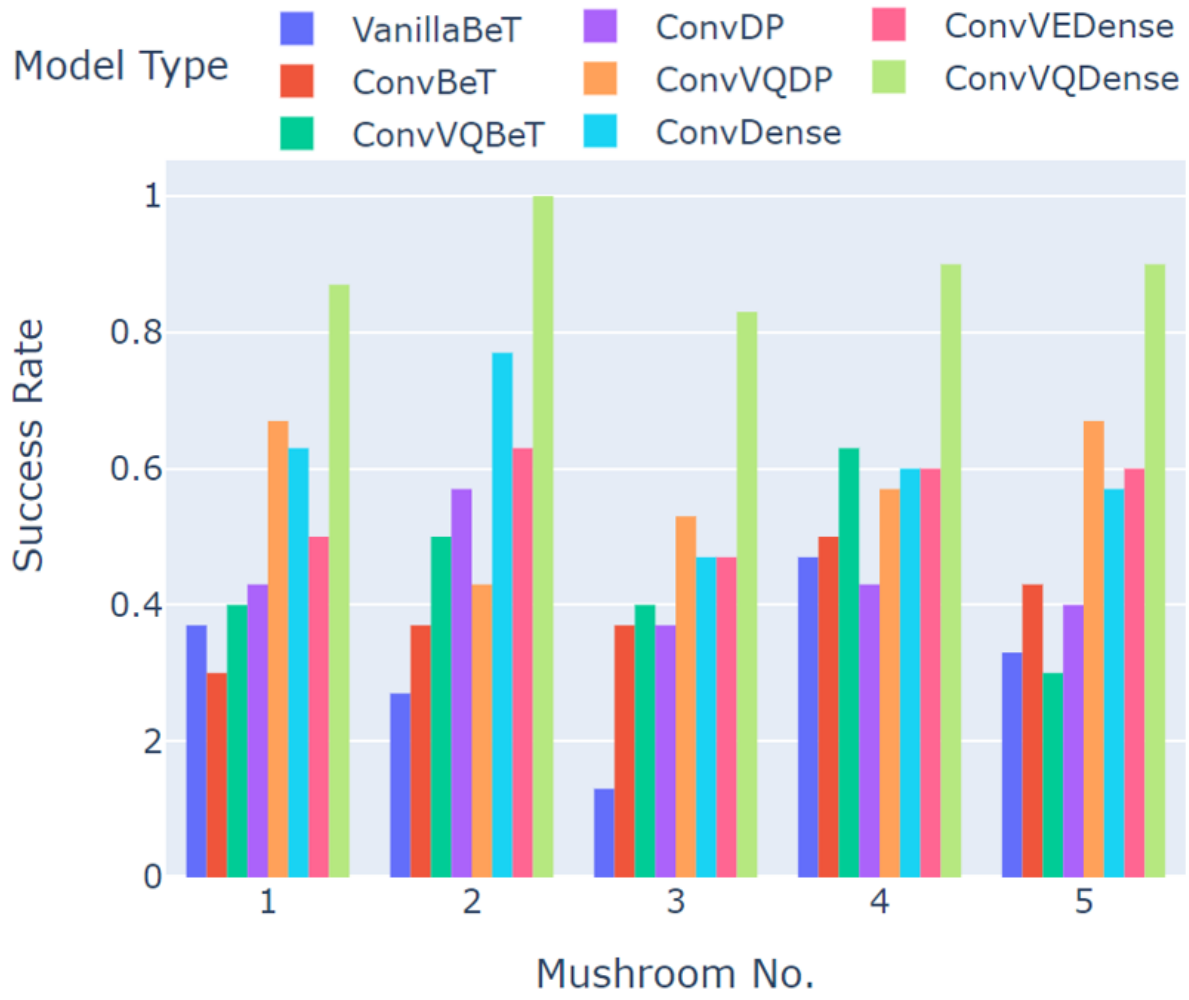

Figure 5.8 Grouped bar chart of success rates for each of the 8 models tested for each of the 5 mushroom models.

Drilling down into the performance of each approach for different mushrooms, seen (No. 1-4) and not seen (No. 5) during training, we observe that our approach, employing a VQ module, consistently outperforms all other approaches across all different mushroom cases. It is also worth observing that the introduction of a VQ module, besides increasing the performance, has the additional benefit of stabilizing it, leading to lower discrepancies across different training seeds.

To further investigate the beneficial effects of the VQ module we performed PCA analysis on the image embeddings passed as inputs to the Action Decoder with and without the VQ (ConvVQDense vs ConvDense). The first two principal components are visualized on the 2D plane in Figure 5.9. Quantized embeddings maintain a much more structured distribution as the episode progresses. In contrast, continuous embeddings tend to disperse forming broad clusters of points that often overlap even though they belong to time steps quite further away within the episode. This is bound to lead to the downstream BC model outputting similar actions in different phases of the task which is bound to be detrimental to the overall performance.

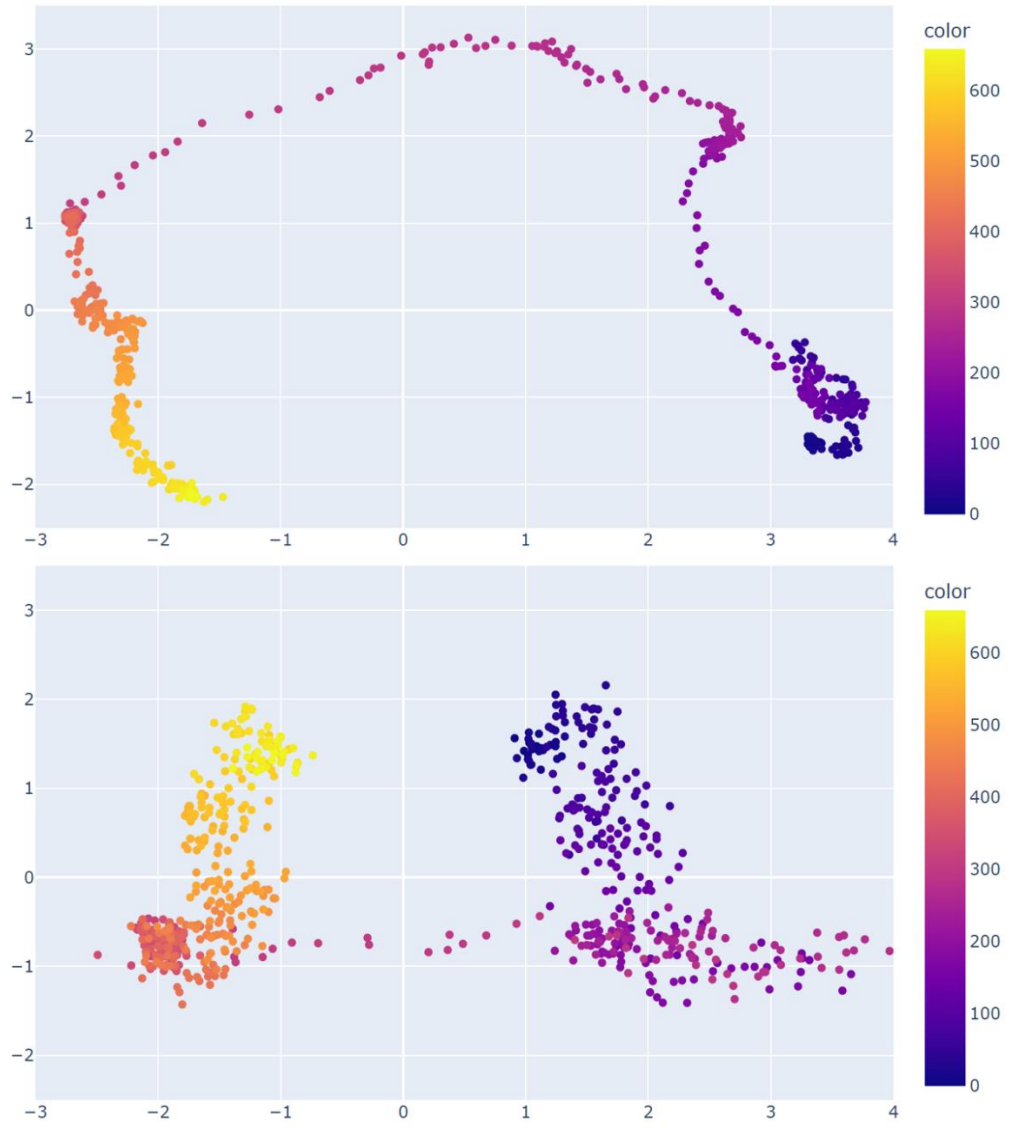

Figure 5.9 Top: embeddings with VQ, Bottom: embeddings without VQ. Color shade indicates the episode step index

5.5 Discussion

Our pixel-based IL approach leverages Vector Quantization, for learning harvesting motion combinations within an environment involving real robot and physical mushroom mockups, designed based on observations from real mushroom harvesting experiments. The approach is computationally lean achieving an inference duration of ~2ms per step on a moderate Laptop GPU (GTX 1650 Ti).

Our approach outperforms two of the most popular state of the art models, namely the Behavior Transformer [40] and a scaled down version of Diffusion Policy [46]. Most interestingly, our results show that introducing the VQ module into these models consistently increases their performance showing that Vector Quantisation can enhance the performance of BC models in general.

The low computational cost is thanks to our explicit choice of using convolutional and de-convolutional layers for producing low dimensional embeddings and decoding actions respectively as opposed to larger, far more computationally heavy models such as state-of-the-art Transformer or Diffusion networks like the ones we compare our approach against. In particular, our pipeline consists of a mere ~1.2m parameters, making it an order of magnitude leaner than the scaled down version of DP which consists of over 60m parameters, and two oreders of magnitude leaner than the original implementation comprising over 200m parameters making it impossible to run in real-time on standard GPU-enabled computers, necessitating action batching, i.e. predicting 16 actions at each inference step and rolling out 8 of these actions as environment steps. The low computational requirements coupled with the low cost of the sensorial streams required for our pipeline make for a practical method and system for IL in industrial setting.

Despite its promising performance our approach's generalisability will be constrained by its ability to handle domain shift, like change of lighting. This is a crucial issue with the entire family of BC models which is further aggravated by their inherent challenges with Covariance Shift [105], i.e. situations where the trained agent encounters observations not well covered in the training set leading to suboptimal actions which take the agent to state subspaces further away from those it was been trained on creating a vicious circle that leads to complete collapse of the model performance in certain trajectories. Future work to mitigate the effects of this domain shift will involve extensive Data Augmentation [106]; the model will be trained on large datasets containing derivative observation instances, namely artificial images that are synthesized by the

existing dataset by applying certain modifications, usually random cropping, rotation and noise addition. In our case more finely tuned modification strategies should be explored such as random region color adjustments that could qualitatively capture changes in lighting. As a next step, the implementation of a simple simulation environment similar to that of Chapter 4 combined with Domain Randomisation [107] can also be explored. Under this regime, the core components of the scene, i.e. the textures mushroom mock-up, the gripper fingers and the table can be synthesised through randomised processes to present significant changes between different episodes or even between episode steps. This would further expand the observation space the system is trained on, mitigating the effect of domain shift. Finally, more principled methods drawing on control theory, such as Stable-BC [108], that are used to reduce covariant shift can also be used for mitigating the effects of domain shift. This method introduces an auxiliary loss term that shapes the learned policy to converge towards the demonstrated behaviours instead of just minimising the action discrepancies. Although such work has mainly been tested on environments where the full state is available to the agent, appropriate loss functions could be formulated so that the domain shifted observation could be matched to the closest observation from the training.

6 One-shot Imitation Learning for Autonomous Mushroom Picking

Through experimenting with the previous methods, we recognised the need for a considerable amount of expert demonstration as a significant bottleneck for a practical system. We thus turned our attention to more data-efficient approaches. Toward this end, we developed a new approach following three core principles that are essentially man-dated by the constraints of operating within an agricultural robotics context: (i) that learning should be accomplished without an extensive number of demonstrations as these are cumbersome and often expensive to collect, (ii) that the sensorial streams enabling IL are easy and cheap to collect and process, and (iii) that the computational load of training and inference should be as low as possible so that the learned policy can be deployed at the edge on hardware with moderate capacity. The latter principle entails a preference towards IL methods that are affordable to train and require limited computational infrastructure to be deployed for real systems control. This work has been published in [109].

6.1 Imitation Learning Architecture

Our IL pipeline comprises four stages; an Image Encoder module that allows for the high-dimensional RGB images input to be cast into a low dimensional embedding, a VQ module that quantizes the embedding based on a learnable vector codebook, a Target Position Decoder that takes as input the quantized embedding, z_t which is concatenated with the encoder measurement of the z-coordinate of the gripper s_t, yielding h_t and predicts the next target position for the gripper $\widetilde{\boldsymbol{p}}_{t+1} = g(\boldsymbol{h}_t)$, and an Image Decoder that reconstructs the image input. The quantized embedding is produced as a 2D concatenation of selected vectors of the codebook which is in turn learned by minimizing an appropriate loss as explained in the next paragraphs. The overall architecture is shown in Figure 6.1.

To train the IL models we use a dataset, $D = \{T_1 \dots T_M\}$, containing M trajectories. Each trajectory in the dataset, $T_i = \{(\boldsymbol{o}_k, \boldsymbol{p}_k),\ k = 1 \dots K_i\}$ is a sequence of observation-action pairs as in the canonical IL setting. In our case, each observation o comprises two elements, namely an RGB image $\boldsymbol{I}$ and the z coordinate of the robot end-effector as obtained by the cartesian robot motor encoders, which is treated as a 1-dimensional vector s_t. Each RGB image $\boldsymbol{I}$ is normalized, with pixel intensities mapped to $[0,1]$, captured by a camera mounted in the palm of the robot gripper. Thus, we are adopting an eye-in-hand approach.

The Image encoder module f, parametrized by $\boldsymbol{\theta_f}$, applies a sequence of convolutional residual blocks which downsample each image sample $\boldsymbol{I_t}$ and then flattens the result into an embedding $\boldsymbol{e_t} = f(\boldsymbol{I_t})$. The embedding $\boldsymbol{e_t}$ is passed through the Vector Quantization module q which yields a quantized embedding $\boldsymbol{z_t} = q(\boldsymbol{e_t})$. This quantization process uses a learnable codebook of vectors $C = \{\boldsymbol{b_i},\ i = 1 \dots N\}$ producing z_t as a concatenation of vectors from C in the following manner. Each block of length B of $\boldsymbol{e_t}$ is matched with its nearest neighbor within C in the Euclidean distance sense, leaving a residual $\boldsymbol{r_i}$. The residuals are used to update the codebook using and Exponential Moving Average (EMA) scheme as in Eq. (5.1).

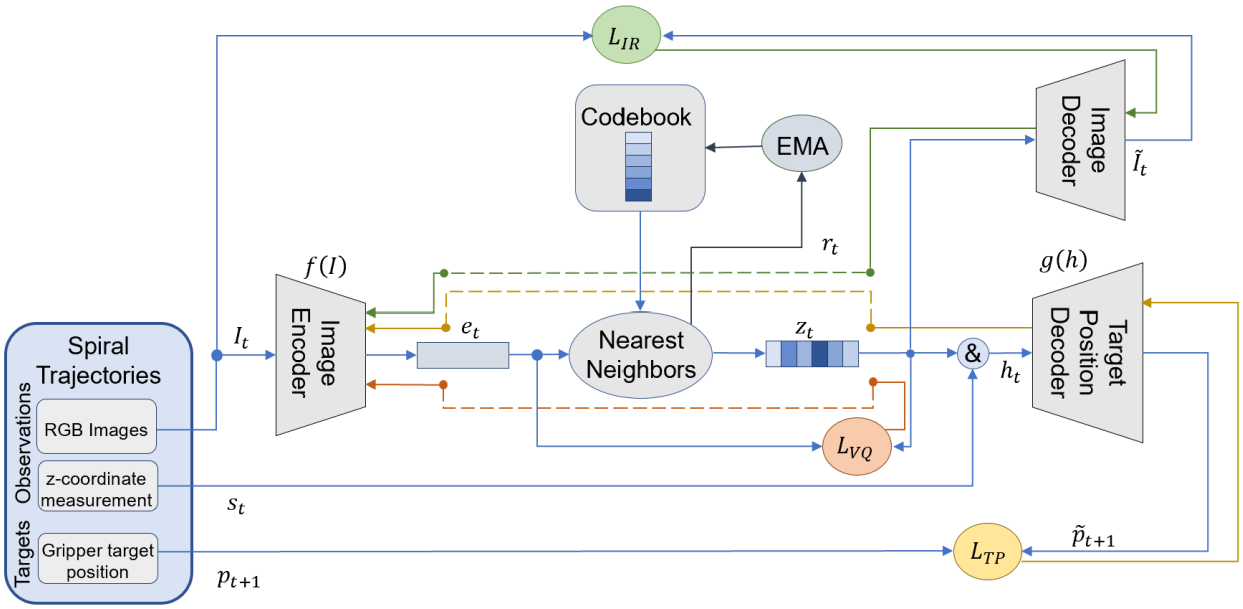

Figure 6.1 Schematic of Imitation learning driven visual servoing approach. The architecture integrates an Image Encoder for processing RGB images, an Image Decoder for frame reconstruction, a Vector Quantization (VQ) module for embedding quantization based on a codebook that is updated using Exponential Moving Average (EMA), and a Target Position Decoder for predicting the Target Position. The system is trained in a self-supervised manner, using a combined loss function that includes reconstruction loss (L_IR), VQ loss (L_VQ), and the Target Position loss (L_TP), facilitating accurate end-effector positioning based on visual and encoder inputs. Dashed lines denote gradient copying to account for the fact that the quantization operation is not differentiable per se.

The Target Position Decoder module comprises a lean Long-Short Term Memory (LSTM) Recurrent Neural Network [110] that predicts the target position $\tilde{p}$. This model is trained to minimize the Mean Squared Error loss:

$$L_{TP}(\boldsymbol{p_t}, \widetilde{\boldsymbol{p}}_{\boldsymbol{t}}) = (\boldsymbol{p_t} - \widetilde{\boldsymbol{p}}_{\boldsymbol{t}})^2 \tag{6.1}$$

The Image Decoder module is a series of deconvolution layers that outputs a reconstruction $\tilde{\boldsymbol{I}}$ of the input image.

$$L_{IR}(\boldsymbol{I}, \tilde{\boldsymbol{I}}) = \boldsymbol{I} \cdot \log \tilde{\boldsymbol{I}} + (\mathbf{1} - \boldsymbol{I}) \cdot \log(\mathbf{1} - \tilde{\boldsymbol{I}}) \tag{6.2}$$

The training of the entire model proceeds in an end-to-end fashion where all modules are trained jointly by minimizing the aggregate loss:

$$L = \sum_{i,t} L_{TP} + \beta L_{VQ} + \lambda L_{IR} \tag{6.3}$$

Where β is a coefficient modulating the relative weight of the vector quantization loss, also termed commitment loss, and λ is the weight of the Image Reconstruction loss. The summation is performed across all timesteps of all the trajectories.

The gradients from the L_{TP} loss flow back through the Target Position Decoder and the Image Encoder while those of L_{VQ} are only flowing through the latter. The L_{TP} and L_{IR} gradients are side-stepping the discontinuity induced by the quantization module by being copied directly from the Target Position Decoder and the Image Decoder respectively to the Image Encoder.

The Target Position Decoder estimates the target position relative to the robot's position in cartesian coordinates. Each such prediction is used to derive a simple velocity controller with a proportional gain K_p. The choice of the controller was driven by simplicity and practicality; we experimentally found that adding integral or derivative terms to the controller offered little benefit in terms of speed in reaching the target position and introduced risks of oscillation around the optimal position. Since the $g(\boldsymbol{h_t})$ estimation is bound to be noisy, we ignore small potential steady-state errors by considering the visual servoing finished when the distance between the current position of the gripper and the target position estimate is less than a threshold δ. The controller equation is thus given as:

$$\boldsymbol{v_{t+1}} = \begin{cases} K_p \cdot g(\boldsymbol{h_t}), & |\boldsymbol{m_t} - g(\boldsymbol{h_t})| \geq \delta \\ 0, & |\boldsymbol{m_t} - g(\boldsymbol{h_t})| < \delta \end{cases} \tag{6.4}$$

This end-to-end training approach, illustrated in Fig. 2, does not require separate training of the Representation Learning modules. The representational power of the embeddings is allocated in

capturing the necessary information to predict the correct target position as well as in reconstructing the original image to keep the embedding manifold as smooth as possible.

The model architecture and parameters of the proposed approach are presented in Table 6.1 below.

Table 6.1. The proposed model parameters.

Model Component	**Type**	**Layers/Parameters**
Image Encoder	Convolutional Neural Network	**Conv layer 1:** 20 channels, 5x5, stride 2 **Conv layer 2:** 10 channels, 3x3, stride 2
Vector Quantizer	EMA-based Quantizer	**Embedding Vocabulary size:** 1024 **Embedding dimension:** 10 **Embedding width/height:** 14*21
Target Position Decoder	Recurrent Neural Network	**Sequence length:** 5 **Hidden layer 1:** 1024 **Hidden layer 2:** 1024
Image Decoder	Convolutional Neural Network	**Deconv layer 1:** 20 channels, 3x3, stride 2 **Deconv layer 2:** 3 channels, 5x5, stride 2

6.2 Environment

The robotic prototype used for the evaluation of the approach is an actuated cartesian robot that has been designed to be compatible with existing mushroom farms. The robot encompasses three actuators, M1, M2 and M3 enabling it o move along a mushroom shelf (x direction) using wheels (M1) and traverse the shelf from side to side (y direction) via a linear slide (M2). An additional linear slide (M3) ensures the ability to lower the robot's end-effector towards the mushroom bed (z-direction). The overall system is shown in Figure 6.2. A detailed presentation of the robot's configuration and actuation principles is provided in [111].

Figure 6.2 The actuated cartesian robotic system used for mushroom picking trials. The gantry-like robot moves along the x-axis with actuated wheels (M1), and along the y-axis and the z-axis with linear slides (M2 and M3 respectively).

The end-effector of the robot is a soft-gripper that has been designed and implemented to replicate the grasping forces recorded during expert demonstration trials. The gripper can deliver a grasping force of max. 2.5N when 1 bar of pneumatic pressure is applied to inflate the fingers. The gripper's design and operational characteristics is detailed in [112]. The design also allows for integrating an in-hand camera in a straightforward fashion. Figure 6.3 illustrates the gripper.

The gripper features three actuators, M4, M5 and M6. The first two are used to configure the gripper's angle towards the mushroom bed while the last one is used to deliver the twisting motion. Of the six actuators of the cartesian robot and the gripper, in our experiments, only M1, M2, M3 and M6 is used during the agent trajectory; M4 and M5 angles stay the same throughout the picking experiments.

For the purposes of visual servoing, within the scope of our work, the gripper motion is controlled with a velocity controller as described in Eq. 6.4. Motion planning is left entirely to the learned agent which calculates the next target position based on visual input from the current image frame.

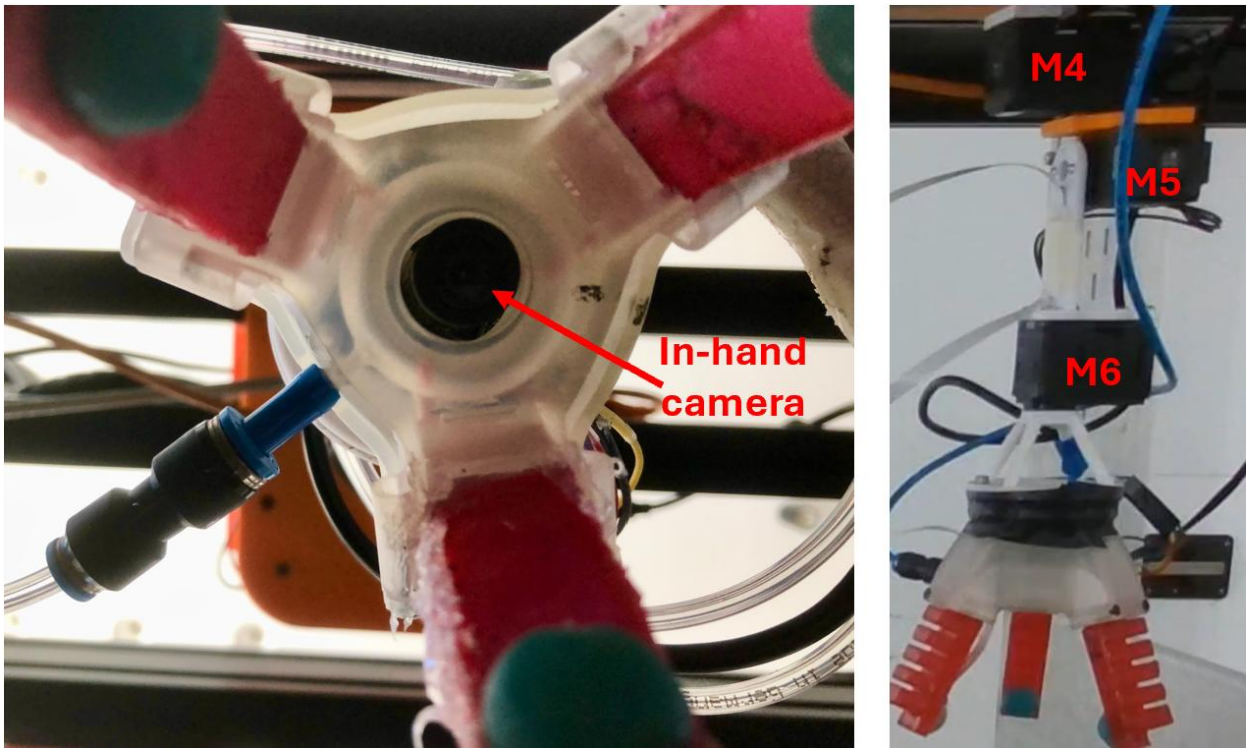

Figure 6.3 The in-hand camera position as well as the motors (M1, M2 and M3) of controlling the motion of the gripper. Only M3 which controls the twisting motion was active during our experiments.

6.3 Experiments

We follow a simple yet practical approach to collecting the necessary data for learning the visual servoing controller. The procedure can be described as follows:

1. The target mushroom is randomly placed on the soil and a number of other mushrooms are randomly placed around it. All mushrooms are 30mm-50mm in cap diameter.
2. The gripper is manually moved in a position that allows for firm grasping, i.e. with the fingers around the target mushroom.
3. The gripper is then moved upwards in a conical spiral with its position and the corresponding image from the in-hand camera recorded at regular intervals. Each observation – relative target position (o_t, p_t) is stored. The radius and the slope of the conical spiral are randomized in each data collection.

We collected a total of 17 such trajectories, each with a different mushroom configuration. The total data collection duration was ~20 min and the dataset comprised 16680 observation-target position pairs. Figure 6.4 and Figure 6.5 illustrate this process, showing a series of screenshots from the eye-in-hand camera during a data collection session and the trajectory of the gripper respectively. We also used extensive Data Augmentation on the Images. This involved several key transformations applied to the input images. Firstly, we adjusted the brightness and contrast levels, which aids in simulating varying lighting conditions and improving model generalization. Additionally, we introduce Gaussian noise to emulate real-world sensor noise and enhance the model's ability to handle noisy input.

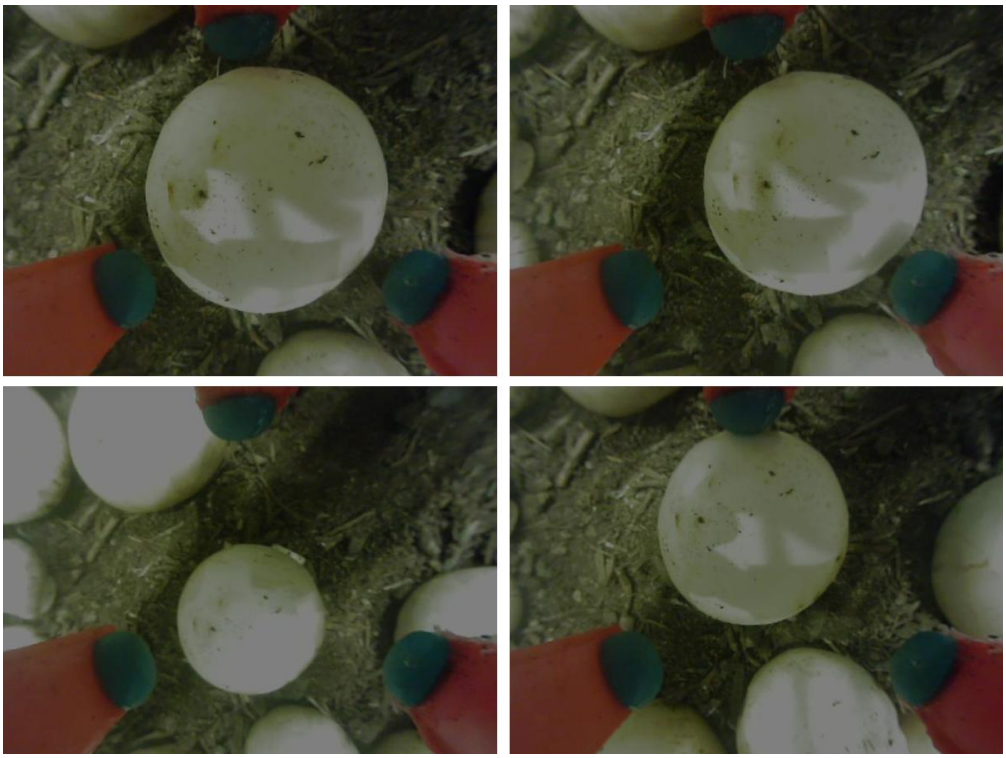

Figure 6.4 Four screenshots of the eye-in-hand camera during a spiral trajectory, Right: 3D plot of the trajectory of the gripper in the 3D space.

Our approach for which we use the identifier, *vq-rec,* is evaluated against the following three benchmarks:

1. A convolutional model following the coarse-to-fine approach of [59] from which our own approach draws inspiration. We refer to this approach as *cnn-c2f*.

2. A simpler variant of our approach where the Image Decoder and the respective loss has been removed to establish the merit of using the Vector Quantization module. This approach is termed *vq-norec*
3. A non-IL based approach where visual servoing is accomplished leveraging YOLOv5 [113], a well-trusted object detector to detect the mushrooms on the scene and a controller is programmed to move the gripper to minimize the error between the centre of the image and the center of the bounding box of the mushroom closer to the centre of the image. This approach is detailed in [111] and we refer to it as *yolo-vs.*

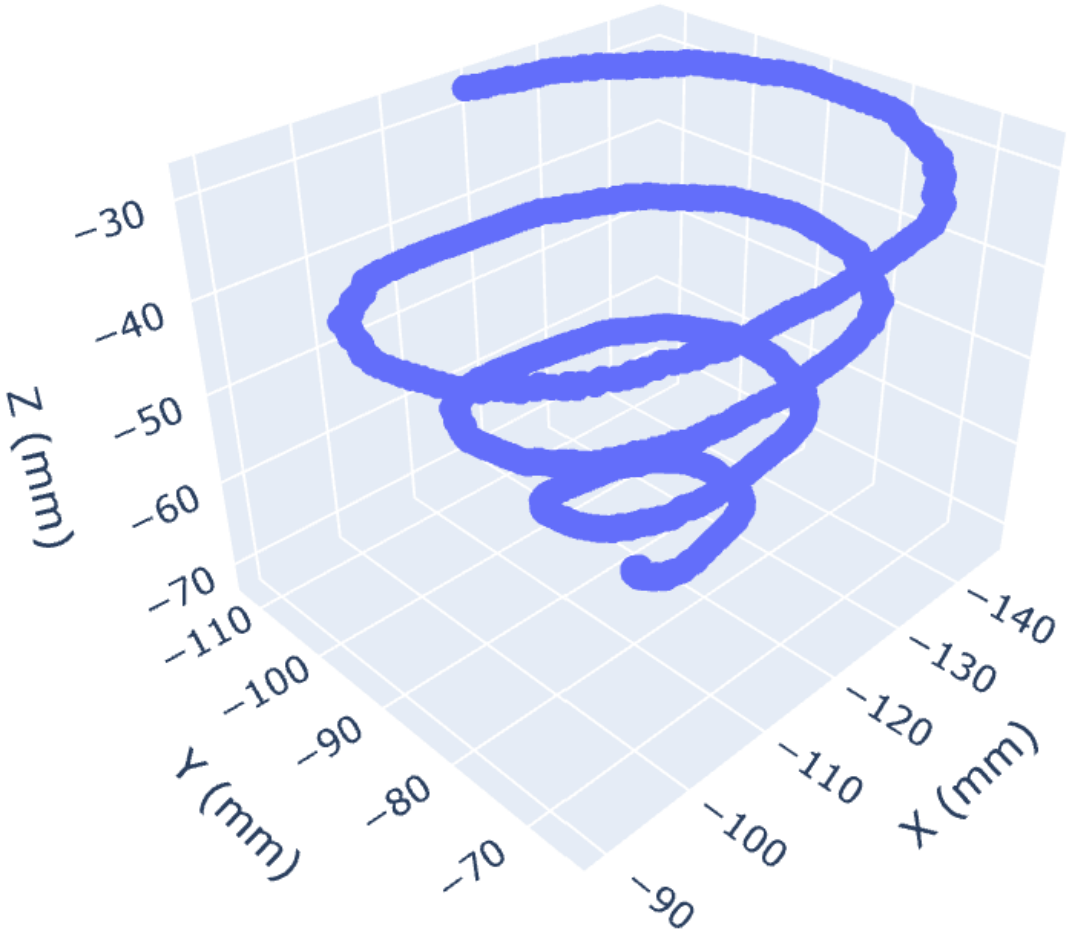

Figure 6.5 3D plot of the trajectory of the gripper in the 3D space

The IL-based approaches, namely *cnn-c2f, vq-norec, vq-rec* are all trained based on the same dataset, collected as described above and they learn a visual servoing controller that moves the gripper to the appropriate position for the grasping, twisting and lifting motion combination to take place. The latter is replicated directly from a single expert demonstration, similarly to [59]. Separating the visual servoing and the outrooting motion sequence can be accomplished by

simply cutting the trajectory at the point where the yaw angle of the gripper starts changing. This is similar to extracting keyframes like in [114]. The expert demonstration is accomplished by a controller with full knowledge of the optimal position for grasping the mushroom.

The *yolo-vs* controller requires learning just for mushroom detection. This is accomplished by collecting and annotating 200 images of various mushroom configurations.

6.4 Results

The resulting models were evaluated over 50 trials, each with a cluster comprising between two and five mushrooms at different positions. The learning-based techniques, namely *vq-rec*, *vq-norec* and *cnn-c2f* were all trained on the same dataset and the transition from the visual-servoing to the scripted twist and lift policy was performed when the prediction of the target position was less than 1mm, i.e. we choose δ=1 in Eq. 6.4. Success or failure was determined in a straightforward manner; whenever the gripper still held the mushroom after the lift motion, the episode was deemed successful. The success rate of each the approaches described above is summarized in Table 6.2.

Table 6.2 Success rates in mushroom grasping an lifting of different approaches.

Approach	Vector Quantization	Image Reconstruction	Success rate
yolo-vs [111]	N/A	N/A	78%
cnn-c2f [59]	No	No	84%
vq-norec	Yes	No	90%
vq-rec	**Yes**	**Yes**	**100%**

As seen in the table above, the proposed approach achieves a remarkable 100% success rate, i.e. it successfully grasped the target mushroom in all instances of the task. Figure 6.6 and Figure 6.7 illustrate a sequence of frames of a successful episode from the front and the eye-in-hand camera respectively. For the rest of the approaches, almost all of the missed cases were mainly due to poor positioning in the z-axis. Indeed, due to the compliant nature of the gripper, a suboptimal alignment in the x-y plane is usually less of a problem as small discrepancies are passively corrected. However, even a few millimetres of deviation from the proper z-position can lead to poor grasping leading to the mushroom slipping off the fingers. The *yolo-vs* approach was shown to be significantly prone to this error mode. We attribute this to the fact that the mushroom scale

is not taken into account in such an approach; a larger mushroom seen from a certain height might look very similar to a smaller mushroom observed by a lower height. The IL-based approaches, however, are inherently taking scale into account through the LSTM-based decoder that operates on a sequence of 5 frames along with the respective z-position of the gripper.

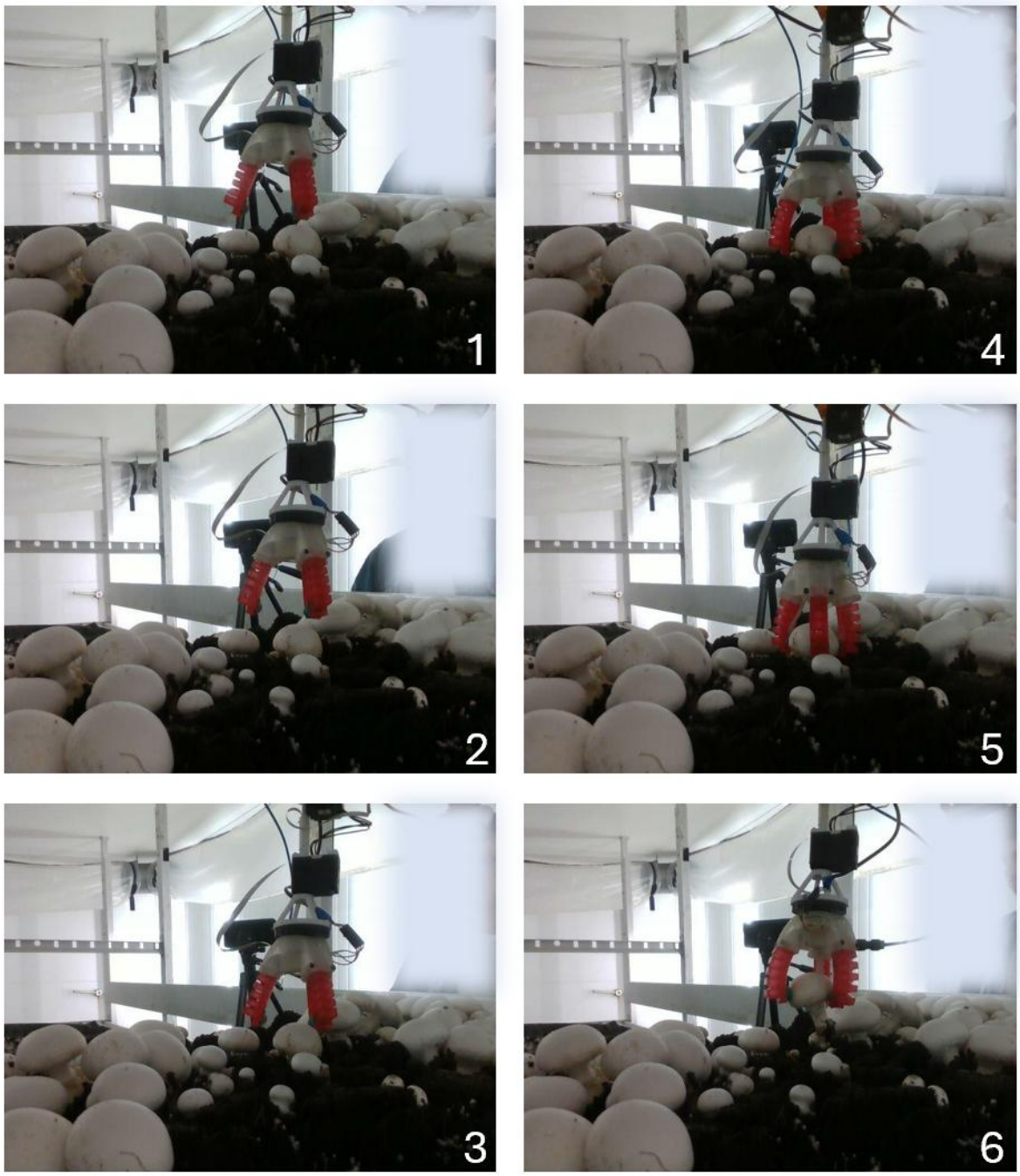

Figure 6.6 Front view and view of a sequence of a successful episode rollout by the trained agent on real mushrooms. (1) the gripper starts at a random position; (2) moves above the mushroom; (3) reaches down; (4) grasps; (5) twists; and (6) pulls the mushroom upwards.

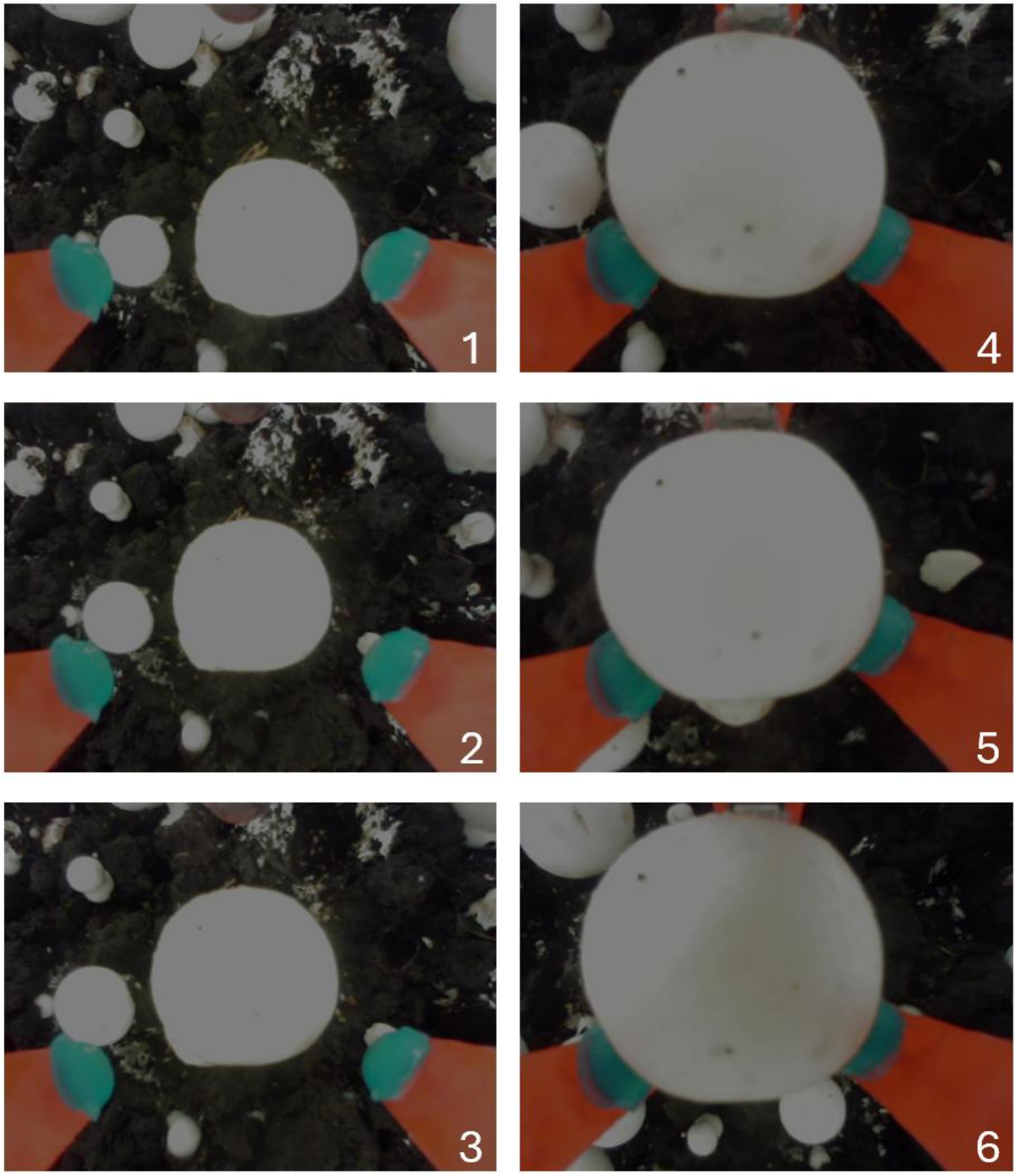

Figure 6.7 Eye-in-hand view of a sequence of a successful episode rollout by the trained agent on real mushrooms. (1) the gripper starts at a random position; (2) moves above the mushroom; (3) reaches down; (4) grasps; (5) twists; and (6) pulls the mushroom upwards.

It is worth noting that the introduction of the Vector Quantization module leads to an increase in performance compared to the Convolutional Neural Network model proposed by *cnn-c2f.* Combining Vector Quantization with Image Reconstruction is shown to significantly improve the

accuracy of the predictions. Figure 6.8 illustrates the predictions produced by the different IL-based agents, based on the observations collected by the expert agent, i.e. a totally scripted agent that has full access to the location of the mushroom to be picked.

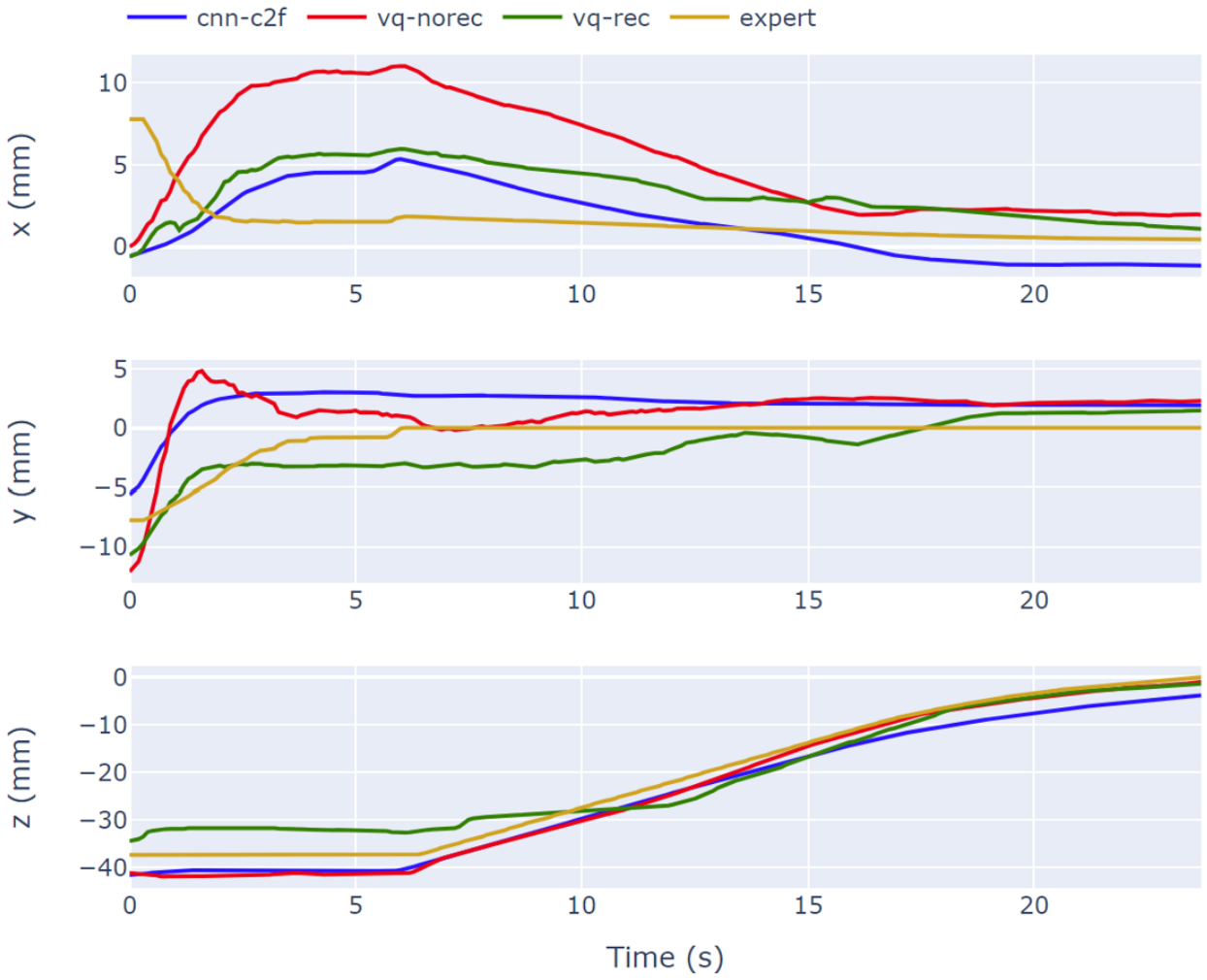

Figure 6.8 Predictions of different IL pipelines on the expert demonstration. Predictions are produced offline, based on the observations obtained during the expert trajectory.

As seen in Figure 6.8, the proposed approach combining Vector Quantization and Image reconstruction predicts the target position closer to the actual one compared to the rest of the benchmarks, particularly for the time steps closer to the end of the episode. Figure 6.9 illustrates the trajectories followed by the expert agent and the agent trained with the proposed approach. In contrast to Figure 6.8, where predictions are produced offline, based on the observations collected by the expert agent, Figure 6.9 illustrates a full rollout by the learned agent on the same conditions with those of the expert episode, i.e. the same mushroom cluster configuration and

the same starting position of the gripper. The proposed agent is able to reach almost precisely the final position of the expert agent, albeit with a delay of ~4 seconds and with some jittering along the trajectory.

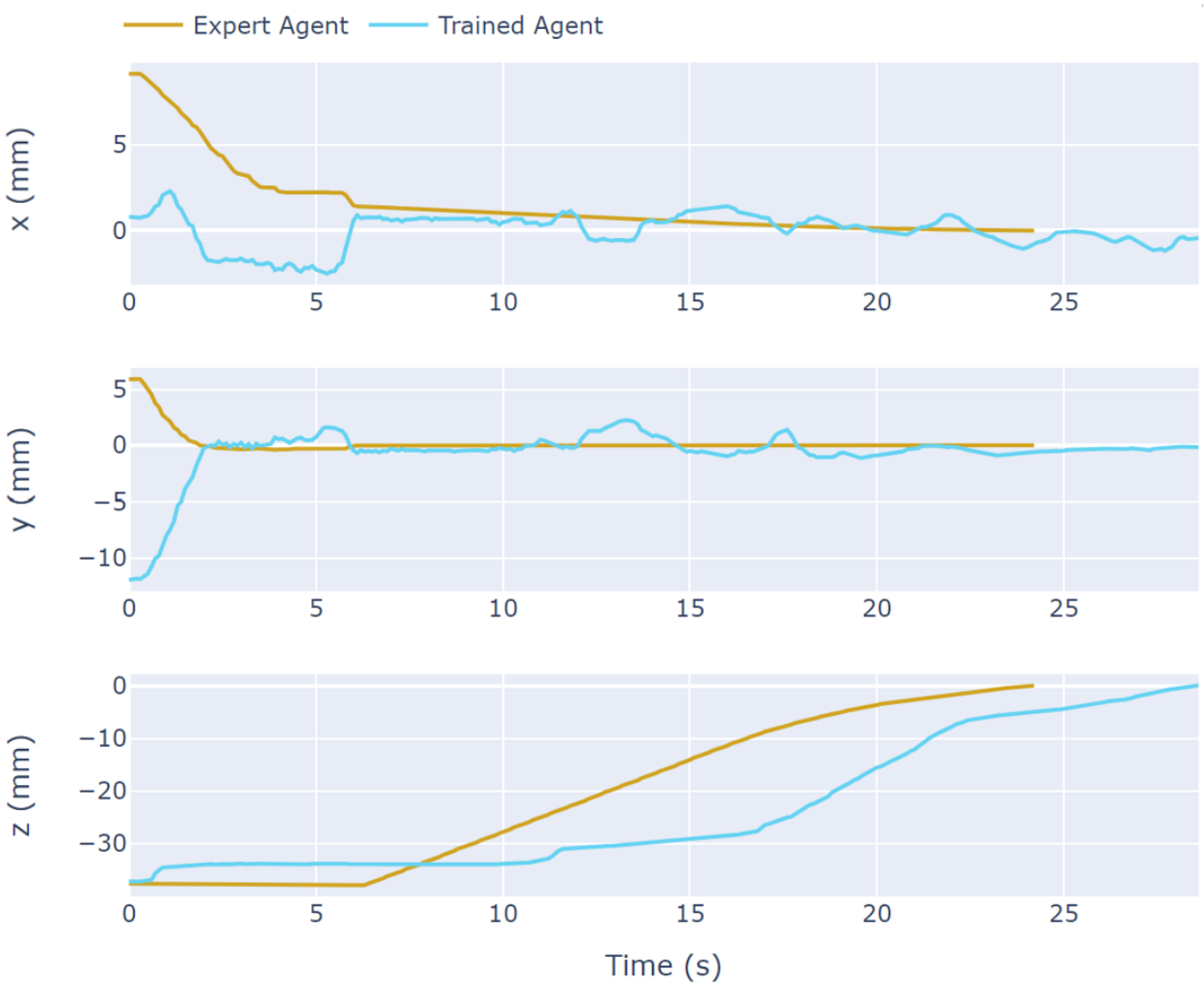

Figure 6.9. Expert and trained agent (vq-rec) trajectories on rollouts with the same environment conditions. Trained agent predictions are produced online by rolling out a new episode.

To further investigate the beneficial effects of Vector Quantization and Image Reconstruction we conduct an analysis of the embeddings passed as inputs to the Action Decoder across the three different IL-based approaches. To visualize the embeddings, we use the well-trusted t-SNE [115] method to map the high-dimensional vectors onto the 2D plane. Figure 6.10 illustrates a visualization of the embeddings of each of the models for a single episode of 305 steps. To accentuate the differences between the approaches we use a relatively low perplexity, value of $p = 5$ in line with the low number of samples.

As seen in Figure 6.10, there are significant differences regarding the smoothness of the embedding manifold; the embeddings produced by the model without Vector Quantization or Image Reconstruction are significantly fragmented, i.e. there are exceptional gaps leading to observations that are close to each other temporally to be mapped far away from each other. The introduction of Vector Quantization significantly improves this mapping, reducing the gaps between episode steps while Image Reconstruction further enhances this aspect. We theorize that this smoothing effect is the main reason behind the improvement of the model performance.

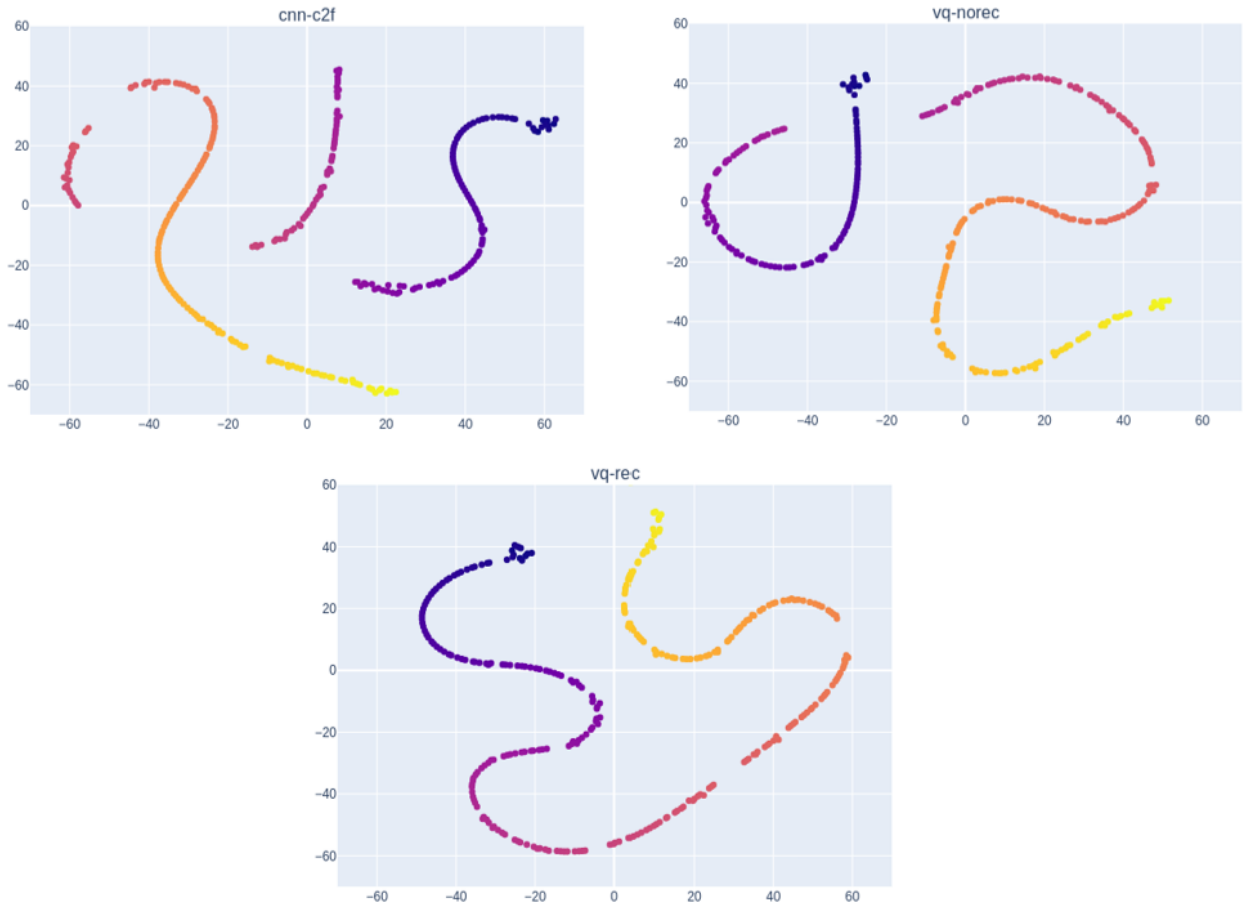

Figure 6.10 t-SNE mapping of embeddings produced by the three IL approaches considered. Colour shade indicates the episode step index.

It is worth noting that even though the actuation of the gripper is not learned by the proposed approach, as the outrooting sequence is just a replication of the expert agent demonstration, accurately positioning the gripper presents challenges that are not normally observed in rigid grippers. An example is illustrated in Figure 6.11 where two screenshots belonging to two different episode rollouts are shown. The rectangular annotations are placed at the exact same

pixel coordinates in both images to highlight that, in contrast to a rigid gripper, whose fingers are expected to show at the exact same location during visual servoing, this is not the case for the soft fingers. The latter are prone to displacements due to hysteresis or because of collisions with other objects. Given the fact that our approach is learning the visual servoing function in an end-to-end fashion, i.e. directly from pixels, such discrepancies present an additional challenge.

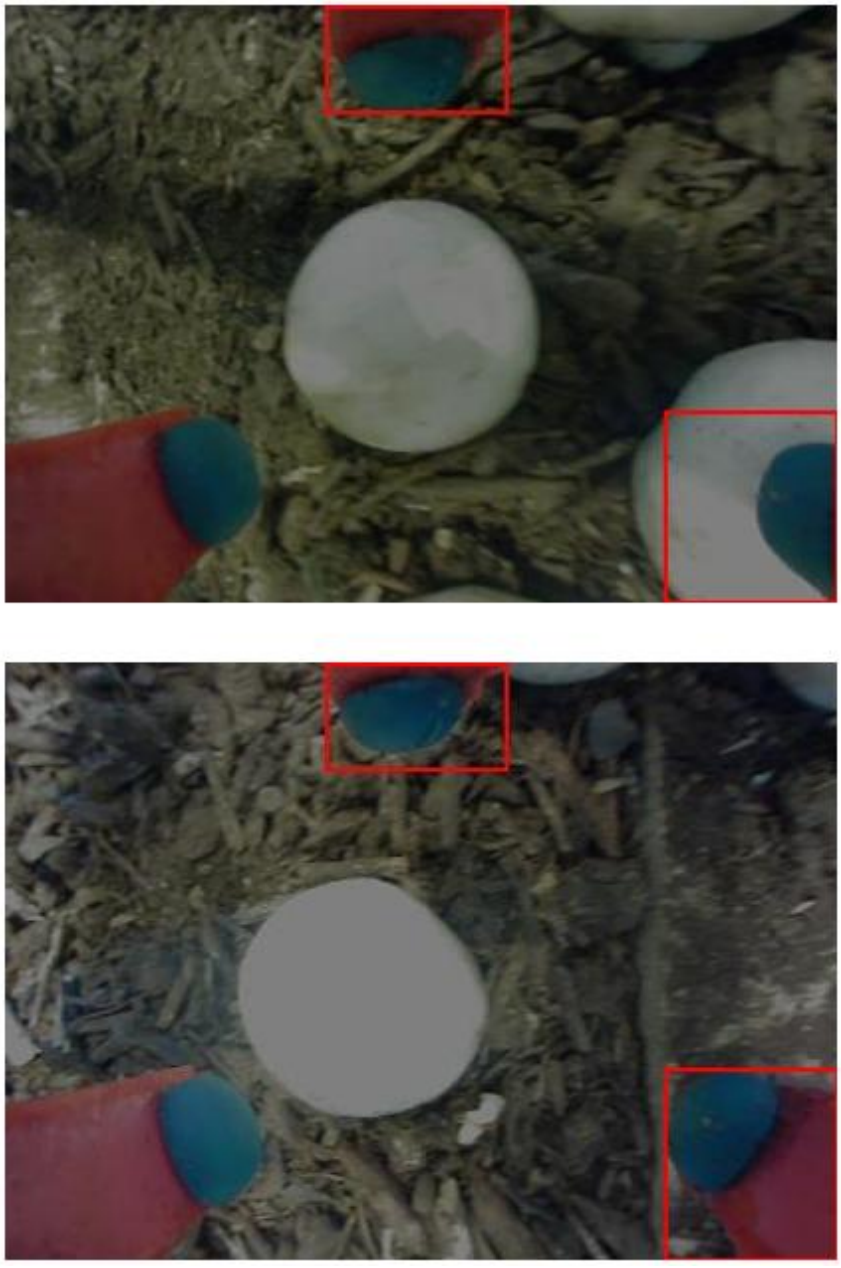

Figure 6.11 Annotated screenshots during trained agent (vq-rec) episode rollouts: (a) the bottom left right finger is significantly displaced due to collision with a nearby mushroom, (b) the top finger is moderately displaced due to hysteresis.

6.5 Discussion

We have presented a pixel-based, one-shot IL approach, leveraging Vector Quantization, for learning to solve the mushroom harvesting task. The approach was tested on a lab-scale mushroom picking environment involving a cartesian robot and a soft, pneumatically actuated gripper. The trials were designed based on observations from real mushroom harvesting experiments. We demonstrated the benefit of using Vector Quantization and showed that the proposed model architecture can learn the desired behaviour with minimal input, namely a simple RGB camera and the z-position encoder measurement of the robot. Our approach operates directly on raw observations with no pre-processing other than downscaling. By casting the problem as a primarily visual servoing task our approach can learn the controller for reaching the optimal position for grasping the mushroom based on less than 20 min of data collection. The approach is shown to solve the task in 100% of the cases even though mushrooms are picked from a cluster of very similar distractors. It is also computationally lean, with the entire pipeline consisting of <800k parameters, allowing for inference of ~20ms on a moderate Laptop GPU (GTX 1650 Ti). The low computational requirements coupled with the low cost of the sensorial streams required for our pipeline make for a practical method and system for IL in an industrial setting.

A crucial aspect of our approach is that it gracefully handles the two most prominent challenges of soft, pneumatically driven grippers:

(i) the lack of proprioception sensing; the shape of the fingers and the location of their tips cannot be directly measured with conventional sensorial systems. In contrast to rigid embodiments where such information can be easily provided using encoders, in soft grippers, determining the shape of the fingers requires sensors such as FBG optical fibres [116], [117] which are costly and difficult to integrate.

(ii) unforeseen displacements due to hysteresis or collisions; the inherent compliance of the fingers renders the prone to changing their default shape under zero pressure as operation cycles strain the material, gradually altering its elasticity properties. At the same time, collisions with distractors can temporarily drastically change the finger configurations. These effects, combined with the lack of proprioception sensing, severely hinder the control of such grippers.

By enabling end-to-end Imitation Learning on soft robotic grippers, our approach lays the groundwork for broader adoption of such systems. This is a crucial design choice for keeping the entire system as low-cost as possible. Silicon-based soft grippers such as the one used in this work can be manufactured in a straightforward manner by silicon casting on 3D printed moulds, and

rely on inexpensive valves rather than motors, thus achieving remarkably low costs of manufacturing [112]. At the same time, thanks to their compliant properties, soft grippers ensure smooth handling of delicate produce by passively conforming to the crop's shape and size and inherently limiting the forces applied based on the choice of the material's elastic properties. In this way, soft gripping eliminates the need for force sensing which drives cost up due to the need of additional hardware modules and integration challenges. This is a crucial aspect from a commercial perspective; in the agricultural sector, and particularly in Europe, where farming is fragmented with almost have of the farm holdings being less than 5ha in size [4] as analysed in Chapter 1, cost is the main factor affecting the rate of adoption of automation technologies.

7 Interpretable Representations for Imitation Learning

The performance of Imitation Learning has so far been primarily judged based on success rate and data- or computation-efficiency secondarily. However, over the past few years, the need for transparent AI approaches has emerged [65], [118]. The robotic agents of the future will not only be required to behave optimally with respect to the task at hand, but also to do so in a way that allows their operators and/or collaborators to understand their inner workings. This is a less explored aspect of Imitation Learning.

Towards this end, we develop a novel Representation Learning (RepL) layer for visual-based IL pipelines that significantly improves the approach's interpretability. For the context of this work, we adopt the term "interpretable", as opposed to "explainable", to describe the ability of a human to directly understand the produced representation rather than obtain post-hoc explanations [118], [119]. Our RepL model is designed to capture an embedding of the image-based observation that directly relates to pixel-based coordinates of certain image patches. In contrast to usual RepL layer outputs that are opaque to human operators, this output is straightforward to visualize, providing insights into what the IL model is focusing on at each step in order to solve the task. The RepL model is fully differentiable and can thus be trained via standard backpropagation enabling end-to-end training of the entire IL pipeline.

We investigate the performance of this approach in a mushroom harvesting context with the added complexity of allowing an expert to specify the target mushroom to be picked instead of the system moving automatically towards the centremost mushroom. This is of particular interest in the mushroom picking domain; selecting the mushroom to be picked is a significant decision as factors like mushroom size and health as well as the profound implications in the surrounding mushrooms play into it.

To further assess the transferability of this approach to more generic robotic manipulation tasks, we investigate its performance in terms of success rates by evaluating it across a range of environments, including a toy task as well as selected tasks of the RoboMimic framework [120]. Like previously, we use two different state-of-the-art Behavioural Cloning models, namely BeT and DP as action decoder and compare the performance of the original implementations vis-à-vis that of using our interpretable RepL layer.

7.1 Representation Learning Architecture

We employ a patch-based approach similar to the ViT architecture [26]. The image is thus broken down in non-overlapping patches. The core of our architecture is designed to predict N sets of per-patch scores, denoted as s_i^n, for each patch in the image, with i indexing over the patch location in the image and n indexing over the N slots. These scores capture the importance of each patch and are used to aggregate the patch embeddings as well as the corresponding normalized coordinates of the centre of each patch yielding the. Different slots are expected to focus, i.e. assign higher scores, to different parts of the image.

The score prediction module builds on the ConvMixer architecture [27], starting with the same initial layers that break the image into 64-dimensional patch embeddings. We extend this with a conditional variant of the ConvMixer layer, incorporating a FiLM [83] conditioning layer that uses the output embeddings of a ResNet18-based encoder [82]. This encoder is also used for observation encoding in all the vanilla policy networks we compare against, making the FiLM layer a straightforward mechanism to leverage the robust output of the ResNet architecture. Apart from this addition, the ConvMixer layer retains its original structure, alternating between depthwise (spatial mixing) and pointwise (channel mixing) convolutional layers.

After passing through the ConvMixer, the patch embeddings are processed by a final pointwise convolution layer to generate the N sets of per-patch score outputs. To encourage the module to assign semantically distinct patches to different sets, we integrate a module that Minimises the Entropy of Sinkhorn (MESH) [91], which normalizes and redistributes the scores across the N slots using an optimal transport algorithm. This allows for sharper and more distinct patch assignments, avoiding redundancy across sets while preserving differentiability in the learning process.

The overall architecture is shown in Figure 7.1. First, the initial patch embeddings z_0 are calculated from the input image $\boldsymbol{I}$ as:

$$\boldsymbol{z}_0 = BN\left(\sigma(Conv_{stride=7,kernel\_size=7}\,(\boldsymbol{I})\right) \tag{7.1}$$

This convolution operation breaks the image into non-overlapping 7x7 patches, each with a 64-dimensional embedding. This dimensionality is maintained throughout the ConvMixer Layers, only being reduced to N channels at the end, where the deep embeddings get transformed into per-patch score sets. We follow the same GELU activation function (σ) and BatchNorm (BN) normalization as used in the original ConvMixer architecture.

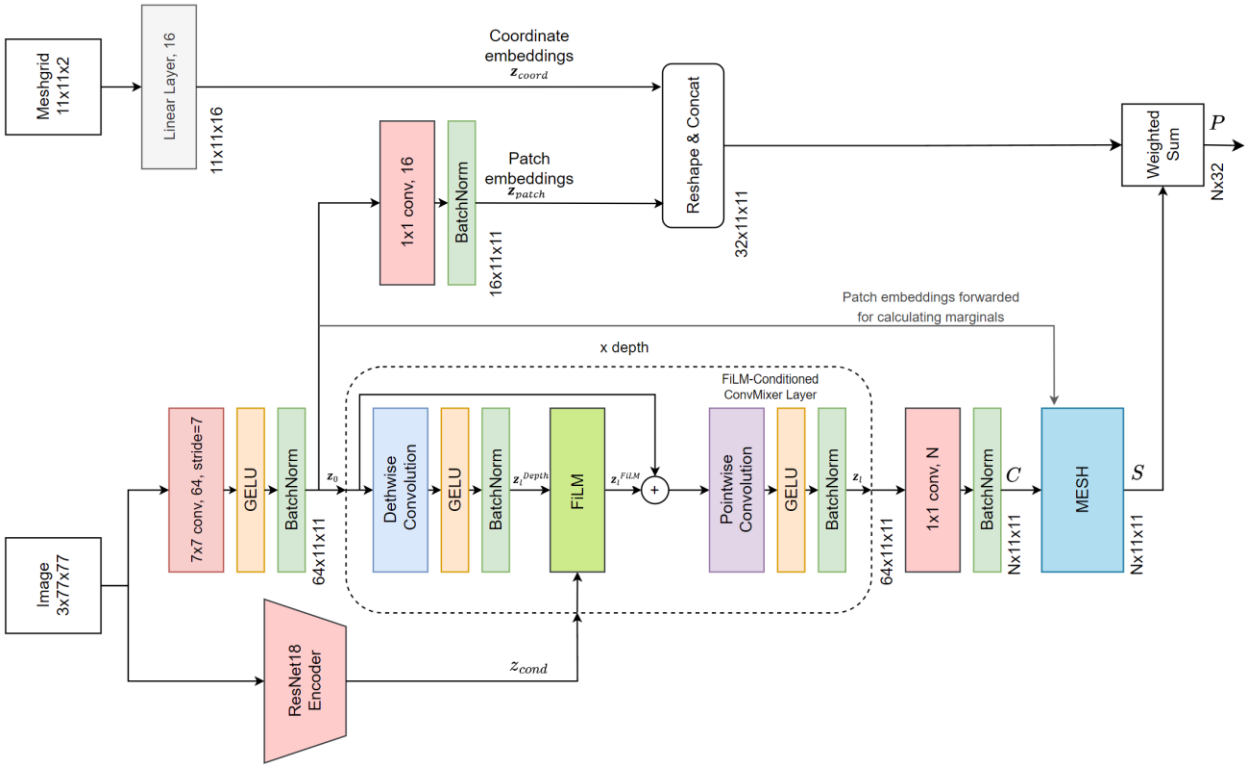

Figure 7.1 Interpretable Representation Learning Module Architecture.

The patch embeddings are processed through a series of L FiLM-conditioned ConvMixer Layers. As in the original ConvMixer, these layers alternate between pointwise (1x1 kernel) and depthwise (per-channel kernel) convolutions. The key difference is the introduction of a FILM layer, which adjusts the embeddings with per-channel scale and bias values. Thus, each ConvMixer block calculates three intermediate embeddings:

$$\boldsymbol{z}_l{}^{Depth} = BN(\sigma\left(Conv_{depthwise,l}(\boldsymbol{z}_{l-1})\right), l = 1, \dots, L \tag{7.2}$$

$$\boldsymbol{z}_l{}^{FiLM} = (\gamma_l(\boldsymbol{z}_{cond}) \odot z_l + \beta_l(\boldsymbol{z}_{cond})) + z_{l-1}, l = 1, \dots, L \tag{7.3}$$

$$\boldsymbol{z}_{l-1} = BN(\sigma\left(Conv_{pointwise,l}\left(\boldsymbol{z}_l{}^{FiLM}\right)\right), l = 1, \dots, L \tag{7.4}$$

where $\odot$ is used to denote element-wise operation as described in [83].

Finally, a pointwise Convolution layer transforms the deep patch embeddings into N per-patch output scores C:

$$\boldsymbol{C} = BN\left(Conv_{pointwise}(\boldsymbol{z}_L)\right) \tag{7.5}$$

The MESH technique, threats C as a cost matrix to compute the transport map S for assigning patches to slots. This is accomplished using a variant of the Sinkhorn algorithm [121] which solves the entropy-regularized optimal transport problem:

$$\boldsymbol{S} = \underset{S \in \mathbb{R}^{N \times M}}{\operatorname{argmin}} \sum_{i,n} s_i^n c_i^n + \lambda H(\boldsymbol{S})$$

$$st. \sum_n s_i^n = \boldsymbol{a}_i, \sum_i s_i^n = \boldsymbol{b}_n,\ i = 1, \dots, M, n = 1, \dots, N \tag{7.6}$$

Where $\boldsymbol{a}_i$ and $\boldsymbol{b}_n$ represent the target patch and slot distributions respectively and H is the entropy function. In our case, those distributions are treated as learned marginals, where the patch distribution is derived from the initial patch embeddings via a fully connected layer, while the slot distribution is derived from learned slot embeddings. These distributions act as weights to reduce the impact of less relevant patches or slots, such as background regions. The Sinkhorn algorithm iterates between row-wise and column-wise normalizations, converging on a matrix that satisfies the distribution constraints. However, by itself, this algorithm tends to produce high-entropy transport maps, leading to patches being distributed across multiple slots, which is undesirable for our purposes. To address this, the MESH module refines the cost matrix $\boldsymbol{C}$ iteratively, applying a gradient descent step to minimize the entropy of the transport map produced by the Sinkhorn algorithm, while also introducing a small perturbation via noise. Thus, the optimization problem shown in (7.6) is solved iteratively with modifying $\boldsymbol{C}$ as follows:

$$\boldsymbol{C}_{t+1} = \boldsymbol{C}_t - \mu \frac{\nabla_{\mathbf{C}_t} H(\boldsymbol{S})}{\left\| \nabla_{\mathbf{C}_t} H(\boldsymbol{S}) \right\|}$$

$$\boldsymbol{C}_0 = \boldsymbol{C} + \varepsilon, \varepsilon \sim \mathcal{N}(0, 10^{-6} \cdot \mathbf{I}) \tag{7.7}$$

This refinement ensures a lower-entropy final transport map, encouraging clearer patch-to-slot assignments. A final execution of the Sinkhorn algorithm yields a transport map with stronger tie-breaking properties, producing more distinct and interpretable assignments. This method preserves differentiability, enabling efficient end-to-end training and offering the potential for extracting interpretable attention maps from the final transport map.

The resulting scores s_i^n are used to produce an image embedding for each slot through a weighted sum of the corresponding patch and position embedding, $\boldsymbol{z}_{patch}, \boldsymbol{z}_{coord}$ as follows:

$$\boldsymbol{p}^n = \sum_{i,n} s_i^n \left(\boldsymbol{z}_{patch} \oplus \boldsymbol{z}_{coord} \right) \tag{7.8}$$

where $\oplus$ denotes the concatenation operation.

7.2 Environments

We test our approach in two types of environments, three simulated generic robotic manipulation tasks and a real environment similar to that of Chapter 5 where a robotic arm attempts to pick 3D printed mushroom models.

7.2.1 Generic Robotic Manipulation Tasks

We use the *Lift, Can* and *Square* tasks from the RoboMimic framework [120] to benchmark our RepL module across generic manipulation tasks. *Lift* requires the robotic arm to simply reach, grasp and lift a cube, *Can* encompasses picking a can and placing it in an appropriate position while *Square* entails picking a square nut and fitting it in a peg. Figure 7.2 shows indicative screenshots for each of these tasks

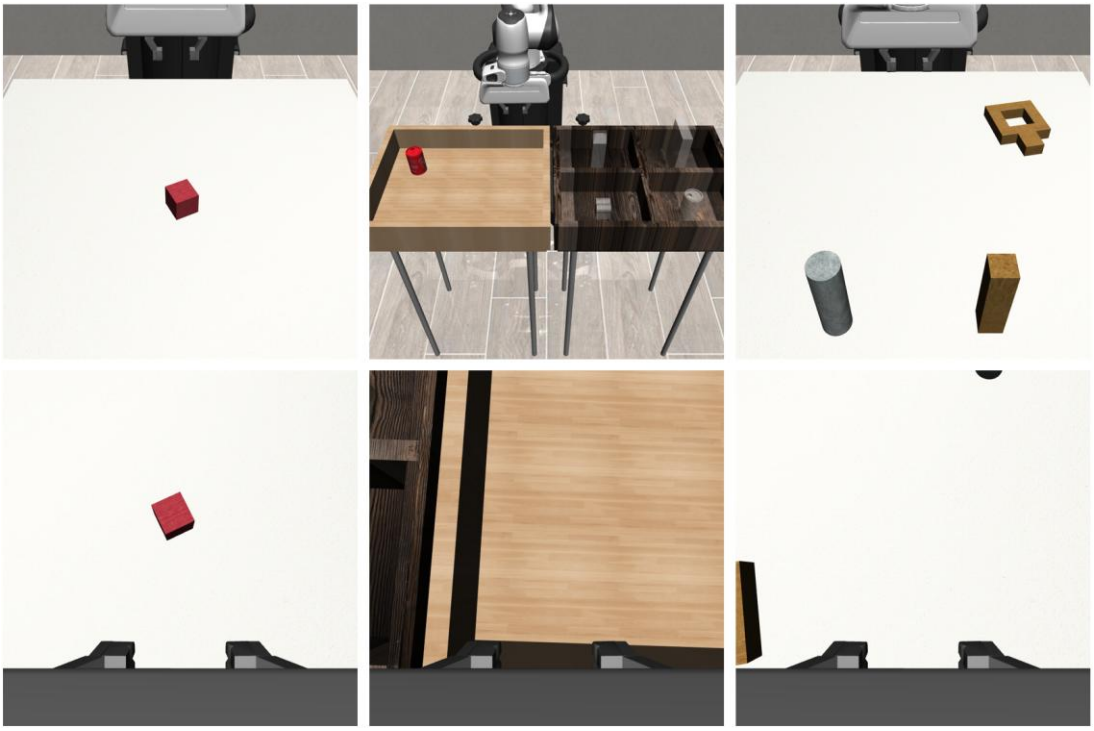

Figure 7.2 Screenshots from the three simulated tasks, Top: Front camera view, Bottom: Eye-in-Hand camera view.

At each environment step, the observations comprise two image frames, one from a camera mounted on the robot's gripper and the other fixed on the opposite side of the table, aimed at the robot and the task scene, as well as the proprioception sensor measurements obtained from the robot's joints and the gripper. No state-based information is provided to the agent, such as the location of the target object or the desired position for placement. The action space comprises the 6 joints of the robotic arm as well as the gripper, for a total of 7 dimensions.

7.2.2 Mushroom Picking Task

We implement a simulation and a real-world task similar to that of Chapter 5 with the additional complexity of multiple mushrooms being in the scene, with the target mushroom being specified at the start of the episode by providing the image coordinates of a point close to its centre. The specified point is then tracked across frames using a standard tracking algorithm, namely the Boosting Tracker [122]. The focus in this investigation is picking a mushroom based on external instructions and thus the simulation does not include a model the root dynamics. In the real-world task these are emulated exactly as described in Chapter 5. The simulation was conducted using the Gazebo engine [123]. The real-world environment also includes soil to enhance realism. Indicative screenshots for both environments are provided in Figure 7.3 and Figure 7.4 respectively.

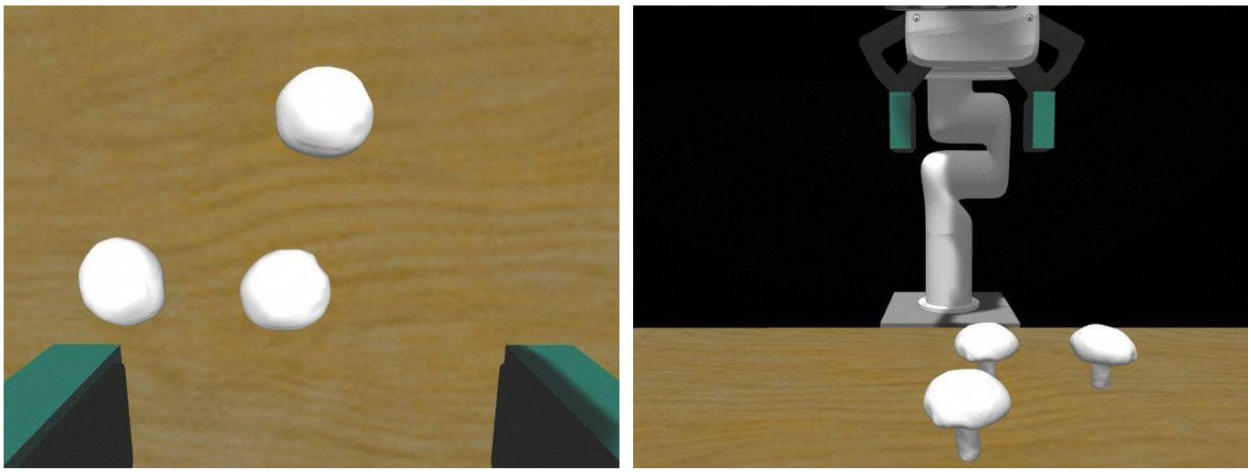

Figure 7.3 Screenshot from simulated mushroom picking task, Left: Eye-in-hand camera view, Right: Front camera view.

Figure 7.4 Screenshot from real-world mushroom picking task, Left: Eye-in-hand camera view, Right: Front camera view.

The observation space is identical to that of Chapter 5; just the image frame from the eye-in-hand camera and the z and yaw measurements from the proprioceptive sensors. Actions are defined in the task space, with the three position coordinates, the yaw angle and the gripper for a total of 5 dimensions.

7.3 Experiments

We conduct extensive experiments to assess the behaviour of the interpretable representations as well as the effect on the accuracy of the overall pipeline. These are described in the next subsections.

7.3.1 Lift, Can and Square manipulation tasks

We evaluate the effectiveness of our RepL module by using two state-of-the-art Behavioural Cloning models, namely Behaviour Transformer (BeT) [40] and Diffusion Policy (DP) [46]. For the latter we train a version of the DiffusionPolicy-C model with fewer neurons per layer that can still achieve near perfect success rates in *Lift* and *Can*. Both models use a simple ResNet-18 as a RepL module to extract feature vectors from the image input in their original implementations. We first produce benchmarks using training these original implementations and producing episode rollouts for each environment using random seeds. We then test the modified versions where the original RepL module is substituted with our own which produces the interpretable representations. Thus, we obtain four IL pipelines, *BeT, DP* and *ModifiedBet*, *ModifiedDP* respectively. We use four slots for the score maps, i.e. we set $N = 4$. The demonstration data were sourced from the RoboMimic framework's webpage which included 200 demonstrations

per task. All IL pipelines process two consecutive frames. The BeT-based pipelines output the next action while the DP-based ones output the next 16 actions, 8 of which are rolled out before the next prediction as in [46]

7.3.2 Mushroom picking task

In this task we explore our approach's capability to operate in a goal-conditioned version of the mushroom picking task explored so far. In both the simulated and real-world version, we collect 200 demonstrations using an expert policy which has access to the 3D location of the cap of the mushroom mock-up to be picked, recorded during the mushroom placement. The training data comprise an additional piece of information, namely the pixel coordinates of a point close to the target mushroom in the image frame of the eye-in-hand camera. This is provided at the start of the episode and the point is tracked across successive frames using a standard implementation of the Boosting Tracker [122]. Since there is no way to specifically determine the centre, and the tracking process is bound to have some drift, the measurement of the tracked pixel is noisy, adding to the challenging nature of the task.

We then consider three methods for providing the tracked pixel coordinates to the IL models. For the simulated environment the first method is to paint the tracked pixel on the image using a red cross and the second is to append the coordinates of the pixel to the embedding produced by the RepL module. For the real-world scenario, we observed that painting the pixel seemed insufficient to achieve high success rates, due to the high color variation induced by noise and lighting variations that are always a challenge in real-world settings. We thus introduced a third variant that provides richer information by passing an modulated inverse distance map as a fourth channel to the input image. This map is produced as follows:

$$\boldsymbol{I}_{map}(i,j) = \exp(-\alpha\sqrt{(i-i_o)^2 + (j-j_o)^2}), i = 1 \dots H, j = 1 \dots W \tag{7.9}$$

where a is a coefficient modulating the breadth of the region carrying distance information, (i_o, j_o) are the coordinates of the target pixel and H, W are the dimensions of the image. This formulation assumes a square image which is the case in our experiments.

We benchmark our approach using the DP model for policy generation. We evaluate all models across 40 simulated and 50 real mushroom mock-up picking trials.

Figure 7.5 Indicative example of determining a target mushroom. Right: Target pixel is painted on the image. Left: Extra image channel carrying an inverse modulated distance map as described in eq. (7.9).

7.4 Results

7.4.1 Lift, Can and Square manipulation tasks

Table 7.1 below summarises the success rates for the original implementations of the BC benchmarks alongside those where the original RepL module has been changed to ours. As seen, the difference in performance is in the order of 2-6%.

Table 7.1 Success rates of original and modified implementation of the BC benchmarks.

IL Pipeline	Lift	Can	Square
BeT [40]	100%	88%	62%
ModifiedBeT	**94%**	**82%**	**58%**
DP [46]	100%	98%	90%
ModifiedDP	**98%**	**98%**	**86%**

Figure 7.6, Figure 7.7 and Figure 7.8 illustrate key screenshots of the important patches identified by our RepL module for the *Lift*, *Can* and *Square* environments respectively. The patches highlighted are the ones with the three highest scores.

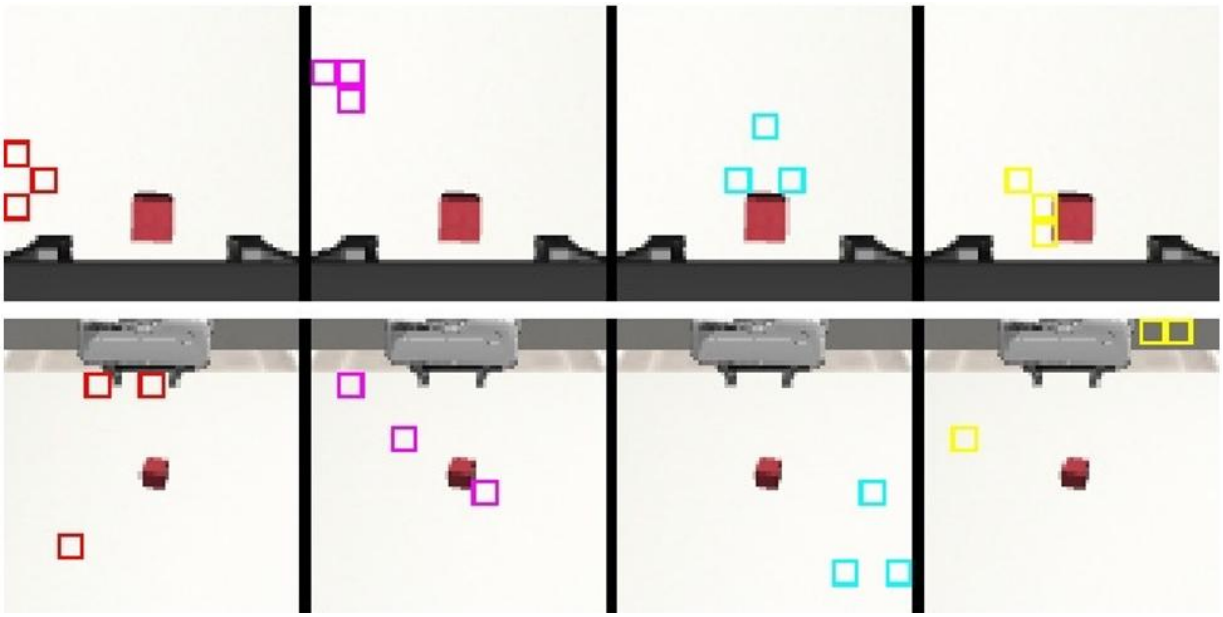

Figure 7.6 Screenshots of the three highest scores for each of the four slots for a single timestep of Lift environment.

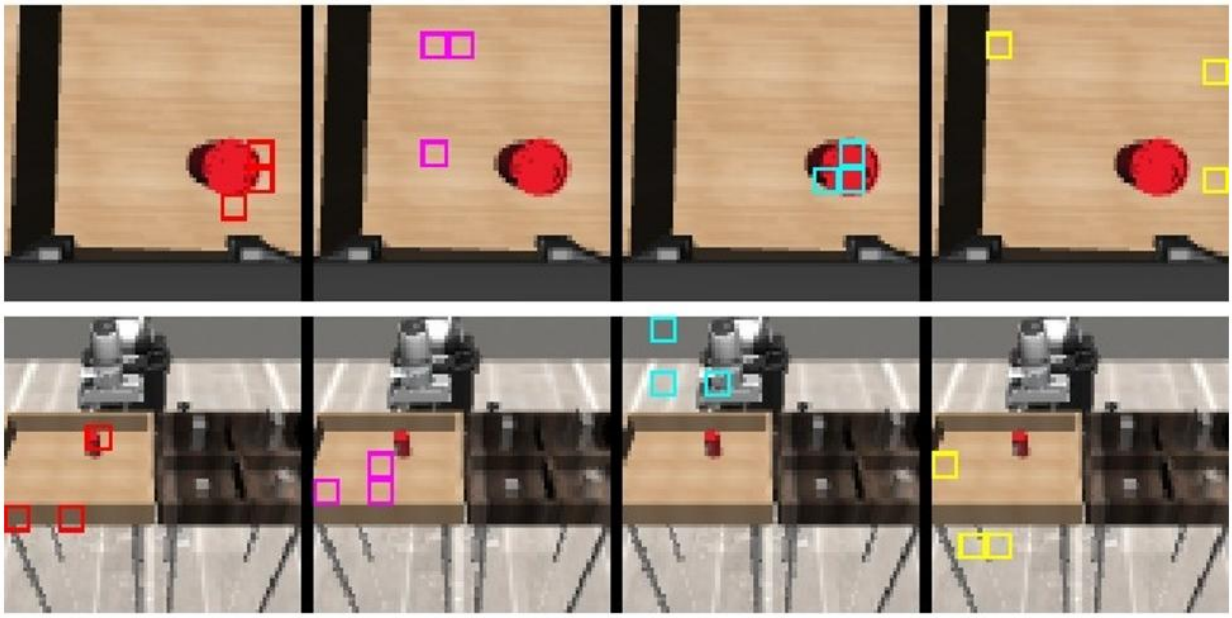

Figure 7.7 Screenshots of highest scores for each of the four slots for a single timestep of the Can environment.

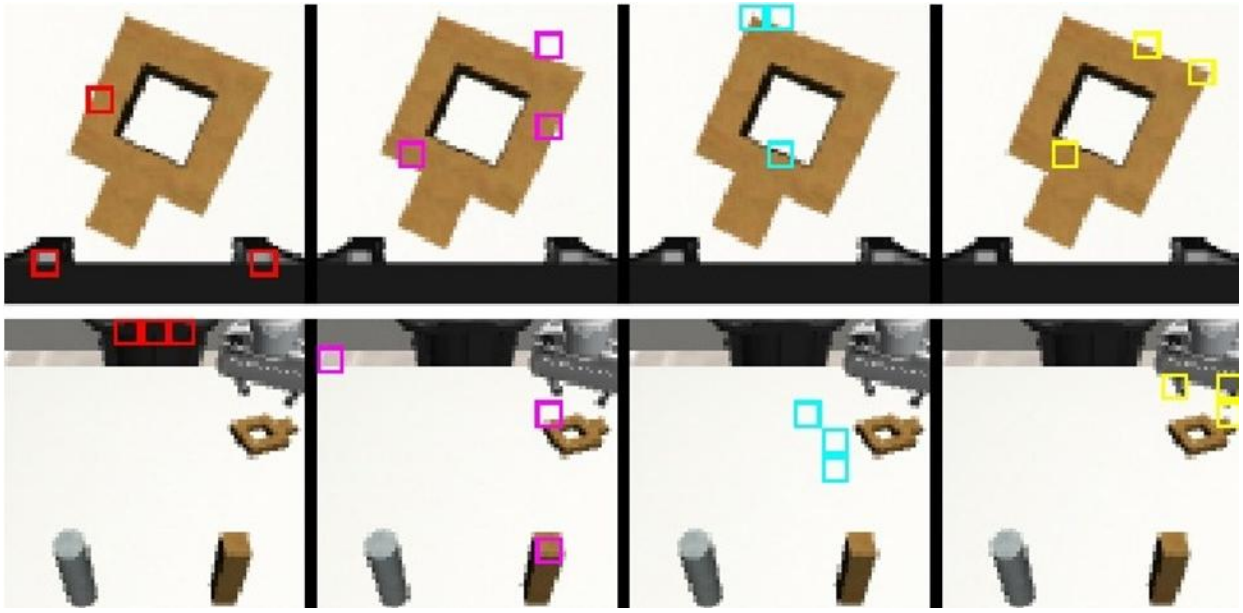

Figure 7.8 Screenshots of highest scores for each of the four slots for a single timestep of the Square environment.

As seen in the figures above, the important patches tend to gravitate towards semantically meaningful regions of the images. This tendency is stronger as the task difficulty increases. In the case of the simpler task, i.e. *Lift* some slots seem to not attend to reasonable regions. This seems to be the case for slots 1,2 for the eye-in-hand-camera and for slots 1,3 for the front view camera. We theorise that this is because, since this task only requires the agent to identify a very simple shape, allocating just two of the slots towards this is enough to solve the task. Indeed, we observe that more of the slots are focusing on meaningful regions in *Can*. This becomes even more evident in the case of *Square*, which is the most complex of the tasks considered as the robot needs to grasp a complex-shaped nut and place it around a peg. In that task, for this particular step, all of the eye-in-hand camera slots attend to parts of the nut.

To explore our RepL module's behaviour across the entire episode, Figure 7.9 illustrates a sequence of screenshots of the eye-in-hand camera and the three most high-scored patches from the second slot. The vast majority of these again attend to semantically important regions of the image, such as edges and corners of the nut or the gripper's fingers. Although at certain timesteps, the highest score patches of a slot may switch to areas of the background instead, this happens when another slot takes over focus to the foreground. Thus, our RepL module can indeed produce representations that capture important regions of the image in an interpretable way with small deterioration of the overall success rates.

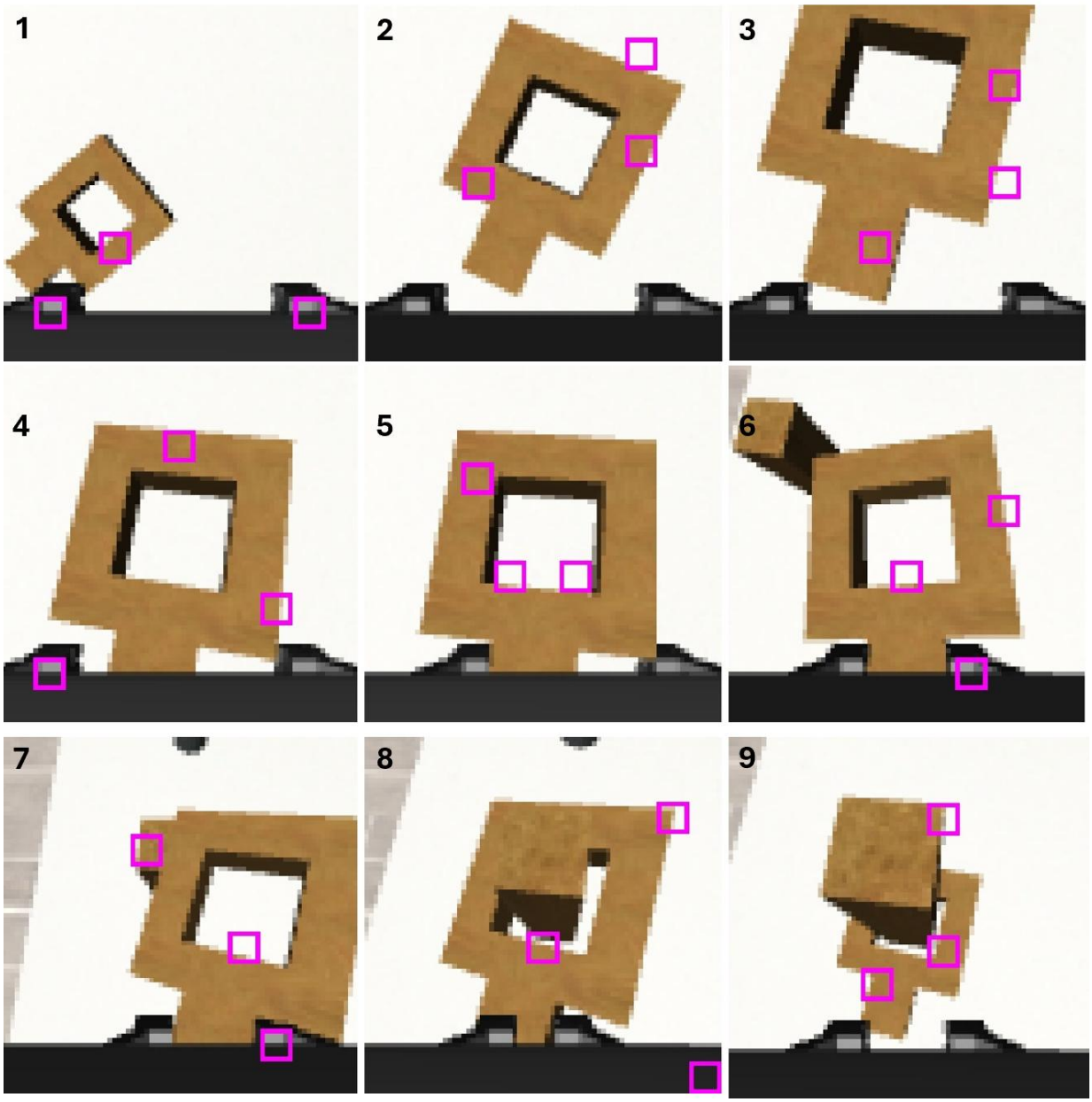

Figure 7.9 Screenshots of a single slot across multiple episode steps of the Square environment for the eye-in-hand camera.

7.4.2 Mushroom picking task

Table 7.2 Success rates for different IL pipelines with different target pixel input modes in simulated mushroom picking task. Table 7.2 and Table 7.3 below summarise the success rates of the original and the modified DP implementations similarly to subsection 7.4.1. An additional experiment is included in Table 7.2 to show that the target pixel coordinates by themselves are not sufficient for reliably performing the picking. In this experiment (*DP without Frames*), we

provide a binary image as an input to the original DP pipeline with a white cross painted against black background so that no information from the actual camera is taken into account.

Table 7.2 Success rates for different IL pipelines with different target pixel input modes in simulated mushroom picking task.

IL Pipeline	Pixel Drawn	Pixel Coordinates
DP without frames	55%	22.5%
DP	97.5%	12.5%
ModifiedDP	**92.5%**	**87.5%**

Table 7.3 Success rates for different IL pipelines with different target pixel input modes in real-world mushroom picking task.

IL Pipeline	Distance Map	Pixel Drawn	Pixel Coordinates
DP	92%	60%	20%
ModifiedDP	**84%**	**58%**	**88%**

As seen in the tables above, the modified DP with our RepL module again performs only slightly worse than the original one in cases where information of the target pixel is provided within the image. However, it is worth noting that our approach performs significantly better when the target pixel coordinates are provided ***after*** the RepL module, as a concatenated vector to the embedding.

Figure 7.10 and Figure 7.11 below provide indicative illustrations of the results of our RepL's representation. Several interesting remarks can be extracted. First, it is evident that most of the important patches are again assigned on or very near semantically important regions, such as mushroom edges or gripper fingers. Second, it is worth noting that that in the case where the target pixel location is not known to the RepL module, the important patches attend to multiple mushrooms whereas in the case of drawn pixel, they tend to cluster around the target mushroom, but not directly on the tracked pixel but rather on the edges of the mushroom. This is crucial to ensure successful grasping as the tracked pixel can be drifting significantly from the centre. Finally, we observe that the important patches attention tends to wane towards irrelevant regions as the episode progresses, particularly after the mushroom is grasped. This is

again sensible since after the mushroom is grasped, the agent can purely rely on the proprioceptive measurements to complete the task, i.e. to twist and lift the mushroom.

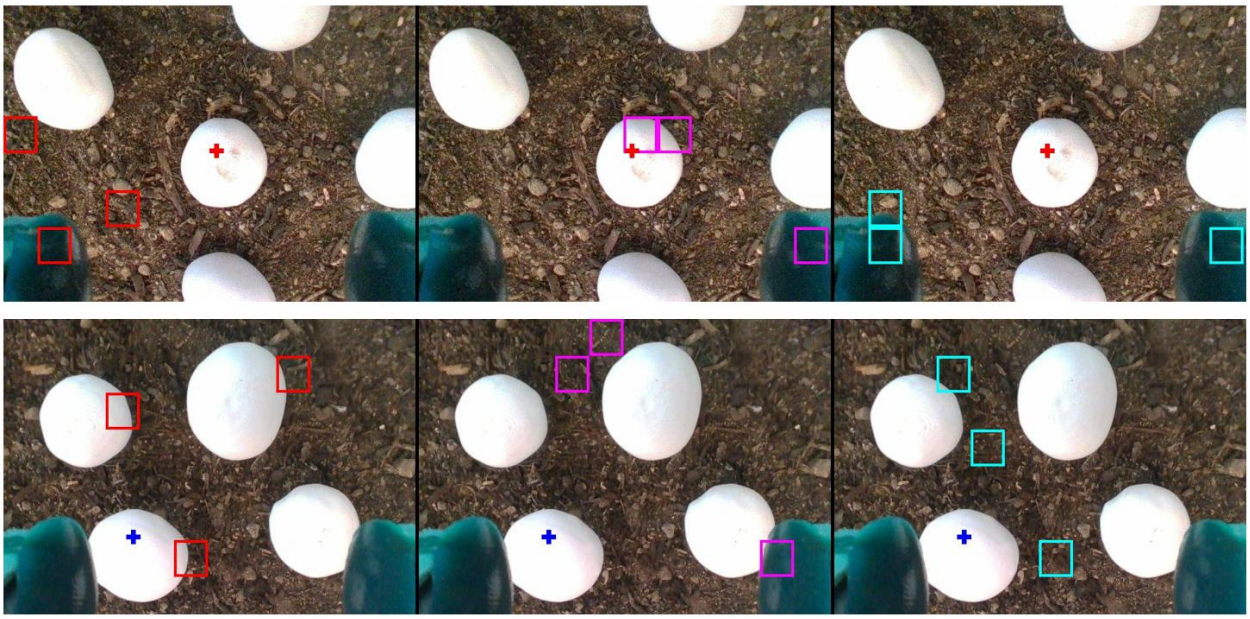

Figure 7.10 Screenshots of highest scores for each of the three slots for a single timestep of the real-world mushroom picking environment. Top: Implementation with target pixel drawn on the image with a red cross. Bottom: Implementation with target pixel concatenated to the embedding. For visualisation purposes, we draw the target pixel in blue colour to distinguish it from the previous case.

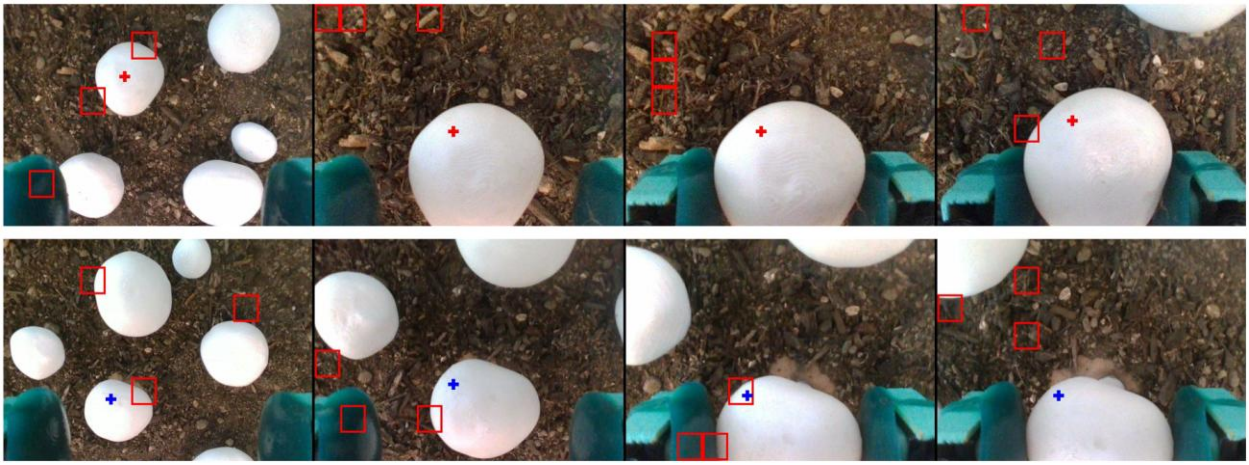

Figure 7.11 Screenshots of a single slot across multiple episode steps of the real world mushroom picking task. Top: Implementation with target pixel drawn on the image with a red cross. Bottom: Implementation with target pixel concatenated to the embedding. For visualisation purposes, we draw the target pixel in blue colour to distinguish it from the previous case.

7.5 Discussion

The Representation Learning Module presented in this chapter, produces interpretable representations that allow for transparency with respect to image regions that are important for solving the task at hand, capturing embeddings directly in terms of pixel coordinates which means that there are readily interpretable by human users. This representation is also shown to be amenable to providing post-RepL goal conditioning in a picking task by means of simple pixel-coordinate tracking. This could enable decoupling of the RepL and the BC module allowing the latter to be retrained separately for different downstream tasks.

A crucial aspect of the designed RepL module is that it operates directly on the image-based observation space, making no assumptions about the spatio-temporal structure or the state composition of the task being carried out. Thus, it can address virtually any visual-based robotic manipulation task. At the same time, the proposed module is fully differentiable allowing for straightforward training via standard gradient descent and back propagation. This means that it can be dropped in practically any Reinforcement or Imitation Learning pipeline, simply by substituting the image encoder, i.e. the stage at which the observation is translated into an embedding with lower dimension before it is being process by the policy model.

To validate the versatility of the proposed approach, we tested it on a variety of different robotic manipulation tasks in both virtual and real settings with diverse challenges. The simulated environments modelled increasingly more difficult tasks ranging from simple reaching, grasping and lifting to picking and placing a complex-shaped nut around a peg with precision. The real task required picking a certain object, in this case a mushroom mock-up, specified by goal-conditioning through pixel-level annotation, among similarly looking distractors and performing a twisting motion to break the adhesive holding it to the table. These tests were performed combination with two significantly different state-of-the-art Behavioural Cloning pipelines, one relying on Transformer and the other on Diffusion models. The fact that experiments across all these environments and BC pipelines showed that the patches ranked as highly important by our RepL module generally captured semantically meaningful regions, shows that our approach can gracefully generalise with respect to different environments.

Naturally, there is significant room for improvement of our approach. It would be important to ensure that different slots focus on different features consistently across timesteps. Designing an inductive bias mechanism to facilitate this would significantly boost transparency and, possibly, eliminate the slight decrease in performance observed compared to non-interpretable RepL

modules. A combination with Hierarchical IL approaches could further increase transparency allowing for interpretability in terms of policy generation besides embeddings. Finally, as computational capabilities continue to increase while visual sensor cost decreases, it would be straightforward to repurpose the module to operate in 3D space, for example, in voxelised representations of the task space, reconstructed by depth sensing to further boost performance in robotic manipulation tasks which greatly rely on 3D perception.

8 Conclusions and Future Work

8.1 General Discussion

Looking back to the results of our research, a taxonomy of desired features of Imitation Learning approaches emerges, which can be considered horizontal with respect to industrial applications. We illustrate it in a simple schematic in Figure 8.1 and explain each term below.

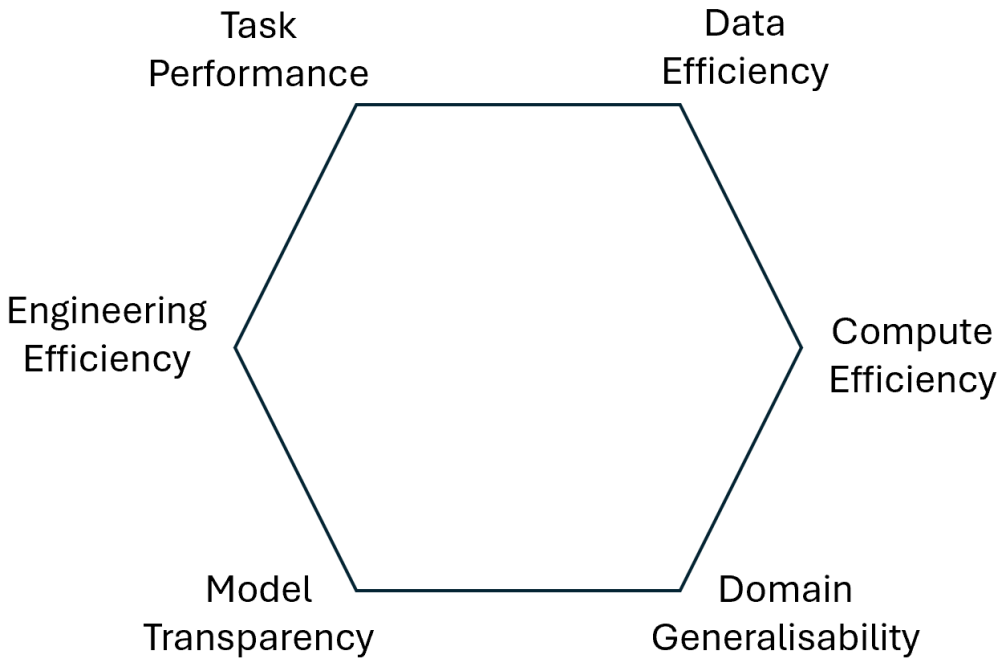

Figure 8.1 Taxonomy of desired features when it comes to Imitation Learning approaches

Task Performance encompasses aspects like accuracy, reliability and robustness to condition variations, i.e. factors that affect the success rate of the algorithm with respect to the task at hand. This is naturally the primary metric by which any approach is judged.

Data Efficiency relates to the size of the data volumes required to train the underlying models, taking into account the effort and cost incurred to gather this data. This is a crucial aspect as it greatly affects the practicality of the approach and therefore its potential for deployment in real environments.

Compute Efficiency considers the computational resources required to run the models in inference mode. This is directly linked to the cost incurred to operate the system as well as energy efficiency and affects real-time capabilities.

Domain Generalisability pertains to the convenience of transferring the Imitation Learning approach to tangential or radically different domains. This is mainly about the strength of assumptions being made about the structure of the tasks which can limit the approach to a certain class of problems.

Model Transparency entails characteristics such as explainability and is a major factor with respect to user trust. It is currently being raised as a crucial aspect for the deployment of AI models in general and new regulations are continuously being pushed with regards to it.

Engineering Efficiency relates to the amount of engineering effort required to implement and fine-tune the Imitation Learning pipeline. End-to-end approaches stand on one end of this spectrum, with engineering effort being minimal while highly tailored pipelines with custom pre-processing steps stand on the other.

These features most often than not, clash with each other requiring careful trade-offs. In the next section we analyse our work with respect to the taxonomy above.

8.2 Retrospective Analysis of our Work

This thesis explored Imitation Learning within the scope of robotic harvesting, an application of crucial importance for the agricultural sector. Automating this process would directly mitigate the challenges caused by labour shortages, caused by demographic shifts, seasonal workforce limitations, and the physically demanding nature of the job. Enabling robots to accurately and reliably carry out harvesting tasks, also contributes to ensuring consistent, high-quality harvests, thanks to timely picking, while increasing productivity laying the groundwork for more sustainable and resilient agricultural practices.

Our work was motivated by the mushroom picking case which on top of the complexity of the harvesting motions presents some additional unique challenges. The common cultivation practice of mushrooms places severe physical limitations; mushroom growing beds are stacked vertically to significant heights and with very little space between them making it extremely difficult for a robotic solution to cover multiple growing beds. This means that multiple robotic platforms are required which places a significant pressure on cost effectiveness. Additionally, unlike other crops where each piece to be harvested is determined solely based on its health and ripeness, picking a mushroom significantly affects the growth of the mushrooms around it. This requires the Imitation Learning algorithms to ideally allow for goal-conditioning so that the right mushrooms are picked in the right order.

Our research was structured in phases, experimenting with different Representation and Supervised Learning approaches with low cost, in terms of sensing modalities and computing resources, considered a hard requirement. We also focused on visual-based, end-to-end approaches to minimise the need for custom pre-processing approaches that would minimise transferability to other crops.

First, we implemented and tested relatively simple Imitation Learning pipelines in a simulated environment, built according to extensive data collection sessions involving human expert pickers. In these sessions, we were able to record visual streams and measure the force interactions during picking, obtaining crucial insights regarding the particularities of this task. We then adjusted our initial approach to ensure transfer to a real-world setting with a real robotic arm and physical mushroom mock-ups in arrangements that qualitatively emulated the dynamics of mushroom picking. We showed that implementing Vector Quantisation as an intermediate step in the Representation Learning module significantly boosted the overall performance, allowing lean policy generation models to reach high success rates. These implementations where described in Sections 4 and 5.

Next, we drew the lessons learned from using VQ and applied them to a different approach which took advantage of the nature of the mushroom picking task as consisting of a reaching-grasping and a manipulation phase with the latter being practically the same across all harvesting instances. This allowed us to implement a pipeline that required a single expert demonstration and a small dataset of auxiliar data, enabling huge gains in terms of Data Efficiency and reaching very high success rates even as it was being tested on a soft, bio-mimetic gripper which added new challenges due to hysteresis and lack of proprioceptive sensing. This was described in detail in Section 6.

Finally, we developed a Representation Module that produced interpretable representations that not only shed light into where the models focus on with respect to the image but also allowed for goal-conditioning of the tasks. The latter feature is crucial as it allows external instructions with respect to which mushroom should be gathered next, during runtime. However, this gain in Model Transparency came at the expense of a slight drop in task performance. Nevertheless, it is our firm belief that this approach has significant room for improvement, paving the way for a new class of interpretable Imitation Learning pipelines for robotic manipulation. To assess this, we also tested in generic robotic manipulation tasks in well-trusted simulated benchmarks. This was analysed in Section 7.

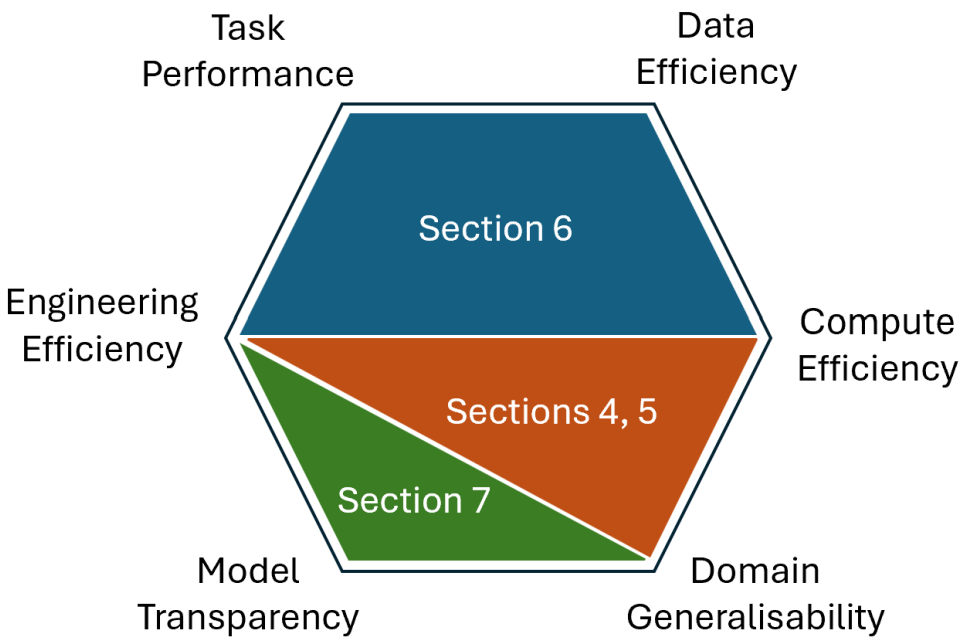

Figure 8.2 Mapping of our implementations with respect to the aforementioned taxonomy

In Figure 8.2 we map the approaches summarised above on to the taxonomy laid out in the previous section. Our initial attempts made no assumptions about the structure of the task at hand and where thus easily generalisable while keeping computational cost at a minimum. However, once we introduced strong assumptions about the harvesting process and the nature of its phases, we were able to achieve very high data efficiency, and task performance at the expense of transferability to other domains. None of these approaches took model transparency into account. The last implementation focused on this aspect while eliminating strong assumptions with respect to the task structure.

8.3 Directions for Future Work

In broader terms, the advent of Imitation Learning, the range of robot capabilities are mainly bound by the power of the AI models underpinning the learning of the expert behaviour. As this power grows, the engineering load associated with designing and implementing complicated rule-based control schemes is gradually diminished, while the range of tasks becoming feasible to solve by autonomous robots greatly expands. It is our firm belief that Imitation Learning holds the key to unlocking the next generation of automation not only in robotic harvesting but across virtually every industrial sector.

In the short-to-medium term we believe that the natural next step would be to combine the lessons learned from the approaches developed within the scope of this thesis towards better

reconciling data efficiency with transparency. This could be achieved by investigating the use of the interpretable representation module tackling the visual servoing phase of the harvesting motion. The patch-based approach is also amenable to more powerful representation structures such as graphs which have recently gained traction in the field of Imitation Learning thanks to their capacity of capturing richer information such as spatiotemporal relationships between regions. Indicative works which could be leveraged towards this end are [124], [125]. To broaden the transferability of such an approach, investigating automated task sequence segmentation could be explored. Further tapping into Object Centric learning approaches could also boost explainability.

In the long term, the rapid advancement of Generative AI should be incorporated more deeply into Imitation Learning pipelines. There is now a considerable body of work investigating the use of Large Language Models and capabilities such as Chain of Thought [126] or Tree of Thought [127] to drive a systematic breakdown of long and complex tasks into shorter and simple constituents. Indeed, several attempts have already been made towards approaches with multi-modal Vision-Language-Action models [128], however these are still only accessible to organisations and institutions with vast computational resources. However there seems to be a clear trend towards reducing the computational cost of fine-tuning such models via techniques such as Low-Rank Adaptation. Eventually, it will be possible to harness the power of this AI model behemoths towards niche Imitation Learning applications. In addition, language can also be leveraged as a tool to provide explanations in an intuitive manner. Interpreting intermediate representations such as ours could become much simpler via a natural language interface. This could build on approaches that have shown great promise in other domains such as recommendation systems [129].

We stand on the brink of a future where robotic systems become capable of accomplishing tasks that seemed impossible mere years ago. It is our duty to ensure that the process of bringing about these new levels of automation comprehensively takes into account the socioeconomic implications, never losing sight of the noble causes that should be driving this revolution; an increasingly sustainable and inclusive future.

References

[1] Publications Office of the European Union, “EU Agricultural Outlook 2018,” 2018.

[2] Publications Office of the European Union, “EU Agricultural Outlook 2023-25,” 2023.

[3] U. D. of A. Economic Research Service, “Farm labor,” 2022.

[4] Eurostat, “Farms and farmland in the European Union: Statistics,” 2022.

[5] United States Department of Agriculture, “Farms and Land in Farms - 2021 Summary,” 2022.

[6] Donal Gernon, “Irish Mushroom Production - Factsheet,” 2020.

[7] The Indepedent (Ireland), “Mushroom Growers Face ''massive crisis",” 2018.

[8] D. J. Royse, J. Baars, and Q. Tan, “Current Overview of Mushroom Production in the World,” *Edible and Medicinal Mushrooms: Technology and Applications*, pp. 5–13, Aug. 2017, doi: 10.1002/9781119149446.CH2.

[9] Teagasc Mushroom Stakeholder Consultative Group, “Mushroom Sector Development Plan,” 2013.

[10] M. Huang, L. He, D. Choi, J. Pecchia, and Y. Li, “Picking Dynamic Analysis for Robotic Harvesting of Agaricus 2 Bisporus Mushrooms 3,” 2021.

[11] J. Carrasco, D. C. Zied, J. E. Pardo, G. M. Preston, and A. Pardo-Giménez, “Supplementation in mushroom crops and its impact on yield and quality,” *AMB Express*, vol. 8, no. 1, pp. 1–9, Dec. 2018, doi: 10.1186/S13568-018-0678-0/FIGURES/2.

[12] S. Yang, J. Ji, H. Cai, and H. Chen, “Modeling and Force Analysis of a Harvesting Robot for Button Mushrooms,” *IEEE Access*, vol. 10, pp. 78519–78526, 2022, doi: 10.1109/ACCESS.2022.3191802.

[13] M. G. M. M. G. Mohanan and A. S. A. Salgaonkar, “Robotic Mushroom Harvesting by Employing Probabilistic Road Map and Inverse Kinematics,” *BOHR International Journal of Internet of things, Artificial Intelligence and Machine Learning*, vol. 1, no. 1, pp. 1–10–1–10, Dec. 2022, doi: 10.54646/bijiam.001.

[14] SureHarvest, “The Mushroom Sustainability Story: Water, Energy and Climate Environmental Metrics,” 2017.

[15] Hannah Ritchie, “The carbon footprint of foods: are differences explained by the impacts of methane?,” *Our World in Data*.

[16] J. M. Hess, Q. Wang, C. Kraft, and J. L. Slavin, “Impact of Agaricus bisporus mushroom consumption on satiety and food intake,” *Appetite*, vol. 117, pp. 179–185, Oct. 2017, doi: 10.1016/J.APPET.2017.06.021.

[17] E. P. AGRI Committee, “The EU farming employment: current challenges and future prospects,” 2019.

[18] Michael Ryan, “Labour and skills shortages in the agro-food sector,” *OECD Food, Agriculture and Fisheries Papers*, no. 189, 2023.

[19] Y. Duan, X. Chen, C. X. B. Edu, J. Schulman, P. Abbeel, and P. B. Edu, “Benchmarking Deep Reinforcement Learning for Continuous Control,” Jun. 11, 2016, *PMLR*. Accessed: Sep. 14, 2023. [Online]. Available: https://proceedings.mlr.press/v48/duan16.html

[20] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, “Recent Advances in Robot Learning from Demonstration,” *https://doi.org/10.1146/annurev-control-100819-063206*, vol. 3, pp. 297–330, May 2020, doi: 10.1146/ANNUREV-CONTROL-100819-063206.

[21] F. Rosenblatt, “Principles of neurodynamics. perceptrons and the theory of brain mechanisms.,” Buffalo NY, 1961.

[22] K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks,” *Int J Res Appl Sci Eng Technol*, vol. 10, no. 12, pp. 943–947, Nov. 2015, doi: 10.22214/ijraset.2022.47789.

[23] A. Van Den Oord, O. Vinyals, and K. Kavukcuoglu, “Neural discrete representation learning,” *Adv Neural Inf Process Syst*, vol. 2017-Decem, no. Nips, pp. 6307–6316, 2017.

[24] A. Ramesh *et al.*, “Zero-Shot Text-to-Image Generation,” *Proc Mach Learn Res*, vol. 139, pp. 8821–8831, Feb. 2021, Accessed: Aug. 03, 2024. [Online]. Available: https://arxiv.org/abs/2102.12092v2

[25] I. Tolstikhin *et al.*, “MLP-Mixer: An all-MLP Architecture for Vision,” *Adv Neural Inf Process Syst*, vol. 29, pp. 24261–24272, May 2021, Accessed: Aug. 04, 2024. [Online]. Available: https://arxiv.org/abs/2105.01601v4

[26] A. Dosovitskiy *et al.*, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” *ICLR 2021 - 9th International Conference on Learning Representations*, Oct. 2020, Accessed: Aug. 04, 2024. [Online]. Available: https://arxiv.org/abs/2010.11929v2

[27] A. Trockman and J. Z. Kolter, “Patches Are All You Need?,” *Transactions on Machine Learning Research*, Accessed: Aug. 04, 2024. [Online]. Available: https://github.com/locuslab/convmixer.

[28] A. Y. Ng and S. J. Russel, “Algorithms for Inverse Reinforcement Learning,” in *ICML ’00: Prcoeedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 663–670.

[29] C. Finn, S. Levine, and P. Abbeel, “Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization,” Jun. 11, 2016, *PMLR*.

[30] N. Das, S. Bechtle, T. Davchev, D. Jayaraman, A. Rai, and F. Meier, “Model-Based Inverse Reinforcement Learning from Visual Demonstrations,” Oct. 04, 2021, *PMLR*.

[31] S. Haldar, V. Mathur, D. Yarats, and L. Pinto, “Watch and Match: Supercharging Imitation with Regularized Optimal Transport,” Jun. 2022, doi: 10.48550/arxiv.2206.15469.

[32] P. Sermanet *et al.*, “Time-Contrastive Networks: Self-Supervised Learning from Video,” in *Proceedings - IEEE International Conference on Robotics and Automation*, Institute of Electrical and Electronics Engineers Inc., Sep. 2018, pp. 1134–1141. doi: 10.1109/ICRA.2018.8462891.

[33] J. H. Openai and S. Ermon, “Generative Adversarial Imitation Learning,” *Adv Neural Inf Process Syst*, vol. 29, 2016.

[34] I. Goodfellow *et al.*, “Generative Adversarial Networks,” *Commun ACM*, vol. 63, no. 11, pp. 139–144, Jun. 2014, doi: 10.48550/arxiv.1406.2661.

[35] R. Dadashi, L. Hussenot, M. Geist, O. Pietquin, and O. P. P. Wasserstein, “Primal Wasserstein Imitation Learning,” in *ICLR 2021 - Ninth International Conference on Learning Representations*, May 2021.

[36] D. A. Pomerleau, “ALVINN: An Autonomous Land Vehicle in a Neural Network,” *Adv Neural Inf Process Syst*, vol. 1, 1988.

[37] R. Rahmatizadeh, P. Abolghasemi, L. Boloni, and S. Levine, “Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration,” *Proc IEEE Int Conf Robot Autom*, pp. 3758–3765, Sep. 2018, doi: 10.1109/ICRA.2018.8461076.

[38] P. Florence *et al.*, “Implicit Behavioral Cloning,” in *Proceedings of the 5th Conference on Robot Learning*, PMLR, Jan. 2022, pp. 158–168.

[39] M. Janner, Q. Li, and S. Levine, “Offline Reinforcement Learning as One Big Sequence Modeling Problem,” *Adv Neural Inf Process Syst*, vol. 2, pp. 1273–1286, Jun. 2021, doi: 10.48550/arxiv.2106.02039.

[40] L. Shafiullah, N. M., Cui, Z., Altanzaya, A. A., & Pinto, “Behavior Transformers: Cloning $ k $ modes with one stone,” in *Advances in neural information processing systems*, May 2022, pp. 22955–22968.

[41] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving Language Understanding by Generative Pre-Training.” Accessed: Aug. 04, 2024. [Online]. Available: https://gluebenchmark.com/leaderboard

[42] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, “Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware,” in *Proceedings of Robotics: Science and Systems*, Apr. 2023.

[43] M. Shridhar, L. Manuelli, and D. Fox, “Perceiver-Actor: A Multi-Task Transformer for Robotic Manipulation,” in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, Sep. 2022. doi: 10.48550/arxiv.2209.05451.

[44] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Adv Neural Inf Process Syst*, vol. 2020-Decem, no. NeurIPS 2020, pp. 1–25, 2020.

[45] T. Pearce *et al.*, “Imitating Human Behaviour with Diffusion Models,” pp. 1–24, 2023.

[46] C. Chi *et al.*, “Diffusion Policy: Visuomotor Policy Learning via Action Diffusion,” 2023, doi: 10.15607/rss.2023.xix.026.

[47] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, “FiLM: Visual Reasoning with a General Conditioning Layer,” *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 3942–3951, Sep. 2017, doi: 10.1609/aaai.v32i1.11671.

[48] A. G. Kupcsik, M. P. Deisenroth, J. Peters, and G. Neumann, “Data-Efficient Generalization of Robot Skills with Contextual Policy Search,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 27, no. 1, pp. 1401–1407, Jun. 2013, doi: 10.1609/AAAI.V27I1.8546.

[49] M. P. Deisenroth, P. Englert, J. Peters, and D. Fox, “Multi-task policy search for robotics,” *Proc IEEE Int Conf Robot Autom*, pp. 3876–3881, Sep. 2014, doi: 10.1109/ICRA.2014.6907421.

[50] T. Schaul, D. Horgan, K. Gregor, and D. Silver, “Universal Value Function Approximators,” Jun. 01, 2015, *PMLR*. Accessed: Apr. 17, 2024. [Online]. Available: https://proceedings.mlr.press/v37/schaul15.html

[51] A. Kupcsik, M. P. Deisenroth, J. Peters, A. P. Loh, P. Vadakkepat, and G. Neumann, “Model-based contextual policy search for data-efficient generalization of robot skills,” *Artif Intell*, vol. 247, pp. 415–439, Jun. 2017, doi: 10.1016/J.ARTINT.2014.11.005.

[52] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, “One-Shot Visual Imitation Learning via Meta-Learning,” in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 78:357-368.

[53] C. Finn, P. Abbeel, and S. Levine, “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks,” in *Proceedings of the 34th International Conference on Machine Learning*, PMLR, Jul. 2017, pp. 70:1126-1135. Accessed: Apr. 16, 2024. [Online]. Available: https://proceedings.mlr.press/v70/finn17a.html

[54] S. James, M. Bloesch, and A. J. Davison, “Task-Embedded Control Networks for Few-Shot Imitation Learning,” in *Proceedings of The 2nd Conference on Robot Learning*, PMLR, Oct. 2018, pp. 87:783-795. Accessed: Apr. 16, 2024. [Online]. Available: https://proceedings.mlr.press/v87/james18a.html

[55] E. Jang *et al.*, “BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning,” in *Proceedings of the 5th Conference on Robot Learning*, PMLR, Jan. 2022, pp. 164:991-1002. Accessed: Apr. 16, 2024. [Online]. Available: https://proceedings.mlr.press/v164/jang22a.html

[56] Z. Mandi, F. Liu, K. Lee, and P. Abbeel, “Towards More Generalizable One-shot Visual Imitation Learning,” *Proc IEEE Int Conf Robot Autom*, pp. 2434–2444, 2022, doi: 10.1109/ICRA46639.2022.9812450.

[57] D. Pathak *et al.*, “Zero-shot visual imitation,” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2018-June, pp. 2131–2134, Dec. 2018, doi: 10.1109/CVPRW.2018.00278.

[58] S. Bahl, A. Gupta, and D. Pathak, “Human-to-Robot Imitation in the Wild,” *Robotics: Science and Systems*, Jul. 2022, doi: 10.15607/RSS.2022.XVIII.026.

[59] E. Johns, “Coarse-to-Fine Imitation Learning: Robot Manipulation from a Single Demonstration,” *Proc IEEE Int Conf Robot Autom*, vol. 2021-May, pp. 4613–4619, May 2021, doi: 10.1109/ICRA48506.2021.9560942.

[60] P. Vitiello, K. Dreczkowski, and E. Johns, “One-Shot Imitation Learning: A Pose Estimation Perspective,” *Proc Mach Learn Res*, vol. 229, Oct. 2023, Accessed: Apr. 24, 2024. [Online]. Available: https://arxiv.org/abs/2310.12077v1

[61] E. Valassakis, G. Papagiannis, N. Di Palo, and E. Johns, “Demonstrate Once, Imitate Immediately (DOME): Learning Visual Servoing for One-Shot Imitation Learning,” *IEEE International Conference on Intelligent Robots and Systems*, vol. 2022-October, pp. 8614–8621, Apr. 2022, doi: 10.1109/IROS47612.2022.9981982.

[62] Davide Castelvecchi, “Can we open the black box of AI? : Nature News & Comment,” Nature | News Feature. Accessed: Apr. 17, 2024. [Online]. Available: https://www.nature.com/news/can-we-open-the-black-box-of-ai-1.20731

[63] High-Level Expert Group on Artificial Intelligence, “Ethics guidelines for trustworthy AI | Shaping Europe’s digital future,” 2019. Accessed: Apr. 17, 2024. [Online]. Available: https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

[64] T. Speith, “A Review of Taxonomies of Explainable Artificial Intelligence (XAI) Methods,” *ACM International Conference Proceeding Series*, pp. 2239–2250, Jun. 2022, doi: 10.1145/3531146.3534639.

[65] A. Barredo Arrieta *et al.*, “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI,” *Information Fusion*, vol. 58, pp. 82–115, Jun. 2020, doi: 10.1016/J.INFFUS.2019.12.012.

[66] S. M. Lundberg and S. I. Lee, “A Unified Approach to Interpreting Model Predictions,” *Adv Neural Inf Process Syst*, vol. 2017-December, pp. 4766–4775, May 2017, Accessed: Apr. 17, 2024. [Online]. Available: https://arxiv.org/abs/1705.07874v2

[67] M. T. Ribeiro, S. Singh, and C. Guestrin, “‘Why Should I Trust You?’: Explaining the Predictions of Any Classifier,” *NAACL-HLT 2016 - 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Demonstrations Session*, pp. 97–101, Feb. 2016, doi: 10.18653/v1/n16-3020.

[68] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K. R. Müller, “Layer-Wise Relevance Propagation: An Overview,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11700 LNCS, pp. 193–209, 2019, doi: 10.1007/978-3-030-28954-6_10/COVER.

[69] E. Puiutta and E. M. S. P. Veith, “Explainable Reinforcement Learning: A Survey,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12279 LNCS, pp. 77–95, 2020, doi: 10.1007/978-3-030-57321-8_5/FIGURES/7.

[70] D. Hein, A. Hentschel, T. Runkler, and S. Udluft, “Particle swarm optimization for generating interpretable fuzzy reinforcement learning policies,” *Eng Appl Artif Intell*, vol. 65, pp. 87–98, Oct. 2017, doi: 10.1016/J.ENGAPPAI.2017.07.005.

[71] D. Hein, S. Udluft, and T. A. Runkler, “Interpretable policies for reinforcement learning by genetic programming,” *Eng Appl Artif Intell*, vol. 76, pp. 158–169, Nov. 2018, doi: 10.1016/J.ENGAPPAI.2018.09.007.

[72] Y. Coppens, K. Efthymiadis, T. Lenaerts, and A. Nowé, “Distilling Deep Reinforcement Learning Policies in Soft Decision Trees,” in *Proceedings of the IJCAI 2019 Workshop on Explainable Artificial Intelligence*, 2019, pp. 1–6. Accessed: Nov. 25, 2021. [Online]. Available: https://drive.google.com/file/d/1drMQs5XPTEPovKdBtCOfnJ2eentEoyt4/view%0A https://cris.vub.be/files/46718934/IJCAI_2019_XAI_WS_paper.pdf

[73] V. G. Costa, J. Pérez-Aracil, S. Salcedo-Sanz, and C. E. Pedreira, “Evolving interpretable decision trees for reinforcement learning,” *Artif Intell*, vol. 327, p. 104057, Feb. 2024, doi: 10.1016/J.ARTINT.2023.104057.

[74] N. Bougie, T. Onishi, and Y. Tsuruoka, “Interpretable Imitation Learning with Symbolic Rewards,” *ACM Trans Intell Syst Technol*, vol. 15, no. 1, Dec. 2023, doi: 10.1145/3627822.

[75] W. Liu, D. Li, E. Aasi, R. Tron, and C. Belta, “Interpretable Generative Adversarial Imitation Learning,” *Proc Mach Learn Res*, vol. vvv, pp. 1–13, Feb. 2024, Accessed: Apr. 17, 2024. [Online]. Available: https://arxiv.org/abs/2402.10310v1

[76] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, “Explainable Reinforcement Learning through a Causal Lens,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 03, pp. 2493–2500, Apr. 2020, doi: 10.1609/AAAI.V34I03.5631.

[77] Z. Yu, J. Ruan, and D. Xing, “Explainable Reinforcement Learning via a Causal World Model,” *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2023-August, pp. 4540–4548, 2023, doi: 10.24963/IJCAI.2023/505.

[78] T. Zhao *et al.*, “Interpretable Imitation Learning with Dynamic Causal Relations,” in *WSDM ’24: Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, Association for Computing Machinery (ACM), Mar. 2024, pp. 967–975. doi: 10.1145/3616855.3635827.

[79] Y. Li, J. Song, and S. Ermon, “InfoGAIL: Interpretable Imitation Learning from Visual Demonstrations,” *Adv Neural Inf Process Syst*, vol. 30, 2017, Accessed: Apr. 17, 2024. [Online]. Available: https://github.com/ermongroup/InfoGAIL.

[80] D. Zhang, Q. Li, Y. Zheng, L. Wei, D. Zhang, and Z. Zhang, “Explainable Hierarchical Imitation Learning for Robotic Drink Pouring,” *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 3871–3887, Oct. 2022, doi: 10.1109/TASE.2021.3138280.

[81] Y. Tang, D. Nguyen, and D. Ha, “Neuroevolution of Self-Interpretable Agents,” *GECCO 2020 - Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp. 414–424, Mar. 2020, doi: 10.1145/3377930.3389847.

[82] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[83] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, “FiLM: Visual Reasoning with a General Conditioning Layer,” *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 3942–3951, Sep. 2017, doi: 10.1609/aaai.v32i1.11671.

[84] X. Chen *et al.*, “An Empirical Investigation of Representation Learning for Imitation,” Nov. 11, 2021.

[85] J. Park *et al.*, “Object-Aware Regularization for Addressing Causal Confusion in Imitation Learning,” *Adv Neural Inf Process Syst*, vol. 34, pp. 3029–3042, Dec. 2021.

[86] K. Kujanpää, J. Pajarinen, and A. Ilin, “Hierarchical Imitation Learning with Vector Quantized Models,” in *International Conference on Machine Learning*, Jan. 2023.

[87] C. Devin, P. Abbeel, T. Darrell, and S. Levine, “Deep Object-Centric Representations for Generalizable Robot Learning,” *Proc IEEE Int Conf Robot Autom*, pp. 7111–7118, Aug. 2017, doi: 10.1109/ICRA.2018.8461196.

[88] M. Sieb, Z. Xian, A. Huang, O. Kroemer, and K. Fragkiadaki, “Graph-Structured Visual Imitation,” *Proc Mach Learn Res*, vol. 100, pp. 979–989, Jul. 2019, Accessed: Sep. 16, 2024. [Online]. Available: https://arxiv.org/abs/1907.05518v2

[89] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu, “VIOLA: Imitation Learning for Vision-Based Manipulation with Object Proposal Priors,” *Proc Mach Learn Res*, vol. 205, pp. 1199–1210, Oct. 2022, Accessed: Sep. 16, 2024. [Online]. Available: https://arxiv.org/abs/2210.11339v2

[90] F. Locatello *et al.*, “Object-Centric Learning with Slot Attention,” *Adv Neural Inf Process Syst*, vol. 33, pp. 11525–11538, 2020, Accessed: Sep. 16, 2024. [Online]. Available: https://github.com/google-research/

[91] Y. Zhang, D. W. Zhang, S. Lacoste-Julien, G. J. Burghouts, and C. G. M. Snoek, “Unlocking Slot Attention by Changing Optimal Transport Costs,” *Proc Mach Learn Res*, vol. 202, pp. 41931–41951, Jan. 2023, Accessed: Sep. 16, 2024. [Online]. Available: https://arxiv.org/abs/2301.13197v2

[92] A. Porichis, K. Vasios, M. Iglezou, V. Mohan, and P. Chatzakos, “Visual Imitation Learning for robotic fresh mushroom harvesting,” *2023 31st Mediterranean Conference on Control and Automation, MED 2023*, pp. 535–540, 2023, doi: 10.1109/MED59994.2023.10185745.

[93] G. E. Uhlenbeck and L. S. Ornstein, “On the Theory of the Brownian Motion,” *Physical Review*, vol. 36, no. 5, p. 823, Sep. 1930, doi: 10.1103/PhysRev.36.823.

[94] T. P. Lillicrap *et al.*, “Continuous control with deep reinforcement learning,” *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, Sep. 2015.

[95] D. P. Kingma and M. Welling, “Auto-Encoding Variational Bayes”.

[96] X. Chen *et al.*, “An Empirical Investigation of Representation Learning for Imitation,” Nov. 11, 2021.

[97] E. Coumans and Bai Yunfei, “Pybullet, a python module for physics simulation for games, robotics and machine learning.” Accessed: Feb. 15, 2023. [Online]. Available: https://pybullet.org/wordpress/

[98] M. Breyer, E. Zürich, J. J. Chung, L. Ott, R. Siegwart, and J. Nieto, “Volumetric Grasping Network: Real-time 6 DOF Grasp Detection in Clutter,” in *Proceedings of the 2020 Conference on Robot Learning*, PMLR, Oct. 2021, pp. 1602–1611.

[99] J. Bohdziewicz, G. Czachor, and P. Grzemski, “Anisotropy of mechanical properties of mushrooms (Agaricus bisporus (J.E. Lange) Imbach),” *Inżynieria Rolnicza*, vol. R. 17, nr, no. 148, pp. 15–23, 2013.

[100] J. Hua, L. Zeng, G. Li, and Z. Ju, “Learning for a Robot: Deep Reinforcement Learning, Imitation Learning, Transfer Learning,” *Sensors 2021, Vol. 21, Page 1278*, vol. 21, no. 4, p. 1278, Feb. 2021, doi: 10.3390/S21041278.

[101] A. Kumar, J. Hong, A. Singh, and S. Levine, “When Should We Prefer Offline Reinforcement Learning Over Behavioral Cloning?,” *ICLR 2022 - 10th International Conference on Learning Representations*, Apr. 2022, Accessed: Jan. 19, 2025. [Online]. Available: https://arxiv.org/abs/2204.05618v1

[102] A. Porichis, K. Vasios, M. Iglezou, V. Mohan, and P. Chatzakos, “Visual Imitation Learning for robotic fresh mushroom harvesting,” *2023 31st Mediterranean Conference on Control and Automation, MED 2023*, pp. 535–540, 2023, doi: 10.1109/MED59994.2023.10185745.

[103] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A Next-generation Hyperparameter Optimization Framework,” *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2623–2631, Jul. 2019, doi: 10.1145/3292500.3330701.

[104] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[105] S. K. Saksena, B. Navaneethkrishnan, S. Hegde, P. Raja, and R. M. Vishwanath, “Towards Behavioural Cloning for Autonomous Driving,” *Proceedings - 3rd IEEE International Conference on Robotic Computing, IRC 2019*, pp. 560–567, Mar. 2019, doi: 10.1109/IRC.2019.00115.

[106] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *J Big Data*, vol. 6, no. 1, pp. 1–48, Dec. 2019, doi: 10.1186/S40537-019-0197-0/FIGURES/33.

[107] R. Alghonaim and E. Johns, “Benchmarking Domain Randomisation for Visual Sim-To-Real Transfer,” *Proc IEEE Int Conf Robot Autom*, vol. 2021-May, pp. 12802–12808, 2021, doi: 10.1109/ICRA48506.2021.9561134.

[108] S. A. Mehta, Y. U. Ciftci, B. Ramachandran, S. Bansal, and D. P. Losey, “Stable-BC: Controlling Covariate Shift with Stable Behavior Cloning,” Aug. 2024, Accessed: Jan. 27, 2025. [Online]. Available: https://arxiv.org/abs/2408.06246v1

[109] A. Porichis, M. Inglezou, N. Kegkeroglou, V. Mohan, and P. Chatzakos, “Imitation Learning from a Single Demonstration Leveraging Vector Quantization for Robotic Harvesting,” *Robotics*, vol. 13, no. 7, p. 98, Jul. 2024, doi: 10.3390/ROBOTICS13070098/S1.

[110] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/NECO.1997.9.8.1735.

[111] P. Mavridis, N. Mavrikis, A. Mastrogeorgiou, and P. Chatzakos, “Low-cost, accurate robotic harvesting system for existing mushroom farms,” *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, vol. 2023-June, pp. 144–149, 2023, doi: 10.1109/AIM46323.2023.10196219.

[112] N. Pagliarani, G. Picardi, R. Pathan, A. Uccello, H. Grogan, and M. Cianchetti, “Towards a Bioinspired Soft Robotic Gripper for Gentle Manipulation of Mushrooms,” *2023 IEEE International Workshop on Metrology for Agriculture and Forestry, MetroAgriFor 2023 - Proceedings*, pp. 170–175, 2023, doi: 10.1109/METROAGRIFOR58484.2023.10424253.

[113] G. Jocher *et al.*, “ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation,” Nov. 2022, *Zenodo*. doi: 10.5281/zenodo.7347926.

[114] M. Shridhar, L. Manuelli, and D. Fox, “Perceiver-Actor: A Multi-Task Transformer for Robotic Manipulation,” in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, Sep. 2022. doi: 10.48550/arxiv.2209.05451.

[115] L. van der Maaten and G. Hinton, “Visualizing Data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008, Accessed: Apr. 30, 2024. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html

[116] B. Jamil, G. Yoo, Y. Choi, and H. Rodrigue, “Proprioceptive Soft Pneumatic Gripper for Extreme Environments Using Hybrid Optical Fibers,” *IEEE Robot Autom Lett*, vol. 6, no. 4, pp. 8694–8701, Oct. 2021, doi: 10.1109/LRA.2021.3111038.

[117] H. Godaba, I. Vitanov, F. Aljaber, A. Ataka, and K. Althoefer, “A bending sensor insensitive to pressure: Soft proprioception based on abraded optical fibres,” *2020 3rd IEEE International Conference on Soft Robotics, RoboSoft 2020*, pp. 104–109, May 2020, doi: 10.1109/ROBOSOFT48309.2020.9115984.

[118] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, and G. Z. Yang, “XAI—Explainable artificial intelligence,” *Sci Robot*, vol. 4, no. 37, Dec. 2019, doi: 10.1126/SCIROBOTICS.AAY7120.

[119] H. J. Escalante *et al.*, Eds., “Explainable and Interpretable Models in Computer Vision and Machine Learning,” 2018, doi: 10.1007/978-3-319-98131-4.

[120] A. Mandlekar *et al.*, “What Matters in Learning from Offline Human Demonstrations for Robot Manipulation,” *Proc Mach Learn Res*, vol. 164, pp. 1678–1690, Aug. 2021, Accessed: Oct. 04, 2024. [Online]. Available: https://arxiv.org/abs/2108.03298v2

[121] M. Cuturi, “Sinkhorn Distances: Lightspeed Computation of Optimal Transport,” *Adv Neural Inf Process Syst*, vol. 26, 2013.

[122] H. Grabner, M. Grabner, and H. Bischof, “Real-time tracking via on-line boosting,” *BMVC 2006 - Proceedings of the British Machine Vision Conference 2006*, pp. 47–56, 2006, doi: 10.5244/C.20.6.

[123] N. Koenig and A. Howard, “Design and use paradigms for Gazebo, an open-source multi-robot simulator,” *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, pp. 2149–2154, 2004, doi: 10.1109/IROS.2004.1389727.

[124] M. Sieb, Z. Xian, A. Huang, O. Kroemer, and K. Fragkiadaki, “Graph-Structured Visual Imitation,” in *Proceedings of the Conference on Robot Learning*, PMLR, May 2020, pp. 979–989. Accessed: Oct. 06, 2024. [Online]. Available: https://proceedings.mlr.press/v100/sieb20a.html

[125] F. Di Felice, S. D’Avella, A. Remus, P. Tripicchio, and C. A. Avizzano, “One-Shot Imitation Learning with Graph Neural Networks for Pick-and-Place Manipulation Tasks,” *IEEE Robot Autom Lett*, vol. 8, no. 9, pp. 5926–5933, Sep. 2023, doi: 10.1109/LRA.2023.3301234.

[126] J. Wei *et al.*, “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models,” *Adv Neural Inf Process Syst*, vol. 35, Jan. 2022, Accessed: Oct. 06, 2024. [Online]. Available: https://arxiv.org/abs/2201.11903v6

[127] S. Yao *et al.*, “Tree of Thoughts: Deliberate Problem Solving with Large Language Models,” *Adv Neural Inf Process Syst*, vol. 36, May 2023, Accessed: Oct. 06, 2024. [Online]. Available: https://arxiv.org/abs/2305.10601v2

[128] A. Brohan *et al.*, “RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control,” in *Proceedings of The 7th Conference on Robot Learning*, PMLR, Dec. 2023, pp. 2165–2183. Accessed: Oct. 07, 2024. [Online]. Available: https://proceedings.mlr.press/v229/zitkovich23a.html

[129] S. Lubos, T. N. T. Tran, A. Felfernig, S. Polat Erdeniz, and V. M. Le, “LLM-generated Explanations for Recommender Systems,” *UMAP 2024 - Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization*, pp. 276–285, Jun. 2024, doi: 10.1145/3631700.3665185.