POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Strawberries picking scheduling: challenges in robotic harvesting

Tesi di Laurea Magistrale in
Automation and Control Engineering - Ingegneria dell'Automazione

Author: **Chiara Bigi**

Student ID: 968327
Advisor: Prof. Paolo Rocco
Co-advisor: Prof. Amir Ghalamzan Esfahani
Academic Year: 2021-22

Abstract

Agriculture is experiencing a crisis due to economical, social, political, and climatic factors, causing a dropping in manpower in farming. Accordingly, research in robotics has been expanded to improve productivity, specialization, and environmental sustainability in the sector.
Collaborative robots are presenting exciting solutions for the collection of fruits. In berry cultivation, labour represents the largest cost and a vast operational uncertainty for farmers. Therefore, automation is desirable even though picking small soft crops is challenging for manipulators since it requires high accuracy and robustness while working in unstructured environments.

This thesis work aims to address some issues in the field of autonomous strawberry harvesting. It tackles the problem of perception using modern object detectors such as DETR and Detectron2 to identify and classify fruits by ripeness and occlusion properties. Regarding the reach-to-pick task, it exposes some improvements on the Deep-Probabilistic-Movement-Primitives model for trajectory generation. The main novelty stands in the introduction of a deep model based on Graph Attention Networks for the prediction of picking scheduling from visual input. This allows human-like reasoning to reduce failures in autonomous harvesting.
A trajectory dataset and annotated strawberry images dataset are collected and processed for this work. The overall proposed procedure for the autonomous collection of a specific berry target was tested with a Franka Emika robotic arm.

Keywords: Robotics, Reach-to-Pick, Learning from Demonstration, Object Detection, Scheduling, Graph Attention Network, Strawberry.

Abstract in lingua italiana

L'agricoltura sta attraversando una crisi dovuta a fattori economici, sociali, politici e climatici, che stanno causando una diminuzione della manodopera in questo campo. Di conseguenza, parte della ricerca in robotica si è sviluppata con lo scopo di migliorare la produttività, la specializzazione e la sostenibilità ambientale nel settore.
I robot collaborativi stanno introducendo soluzioni interessanti nel campo della raccolta della frutta. Nella coltivazione dei frutti di bosco, la manodopera rappresenta il costo maggiore, nonché una incertezza operativa per gli agricoltori. Pertanto l'automazione in questo settore sarebbe vantaggiosa, nonostante maneggiare piccoli frutti sia impegnativo per i manipolatori poiché richiede elevata precisione e robustezza lavorando in un ambiente non uniforme.

Questo lavoro di tesi affronta alcune problematiche relative alla raccolta autonoma delle fragole. Propone l'utilizzo di moderni strumenti di object detection come DETR e Detectron2, per identificare e classificare i frutti in base alle proprietà di maturazione e occlusione. Per quanto riguarda il problema di reach-to-pick, espone alcuni miglioramenti al modello Deep-Probabilistic-Movement-Primitives per la generazione di traiettorie. Ma la principale novità sta nell'introduzione di un deep model basato su Graph Attention Networks per la predizione di uno scheduling di raccolta partendo da input visivi. Ciò permette una percezione simile a quella umana, con lo scopo di diminuire le imprecisioni nella raccolta autonoma.
Per questo lavoro di tesi sono stati raccolti ed elaborati un dataset di traiettorie e un dataset di immagini di fragole provvisto di annotazioni. La procedura proposta per la raccolta autonoma di una specifica fragola è stata testata su un braccio robotico Franka Emika.

Parole chiave: Robotica, Reach-to-Pick, Learning from Demonstration, Object Detection, Scheduling, Graph Attention Network, Fragola

Contents

Abstract **i**

Abstract in lingua italiana **iii**

Contents **v**

1 Introduction **1**
1.1 Automation in the farming industry 1
1.1.1 Arising issues in the agricultural sector 1
1.1.2 Robots in agriculture 2
1.2 Case study: strawberries 3
1.2.1 Robots challenges in strawberry farms 4
1.2.2 Picking scheduling decision problem 5
1.3 Thesis research objectives and achievements 6
1.4 Thesis structure 7

2 State of the art **9**
2.1 Deep Learning in Motion Planning for Reach-to-pick task 9
2.2 Object Detection for strawberries identification and classification 11
2.3 Graph Neural Networks for scheduling fruit harvesting 14

3 Deep Probabilistic Movement Primitives for Motion Planning **19**
3.0.1 Fundamentals of Deep-ProMPs 19
3.1 Improved Deep-ProMP 22

4 Scheduling decision **25**
4.1 Strawberries Object Detection 25
4.2 Heuristic scheduling computations 28
4.3 GAT model 32

5 Experimental Setup **37**
5.1 Motion Planning 37
5.1.1 Trajectories dataset collection 37
5.1.2 Data processing 40
5.2 Scheduling decision 42
5.2.1 Dataset description 42
5.2.2 Data as Graphs 43
5.3 Robot test of the models' integration 44

6 Results and Discussion **47**
6.1 Object Detection 47
6.2 Graph Attention Network 48

7 Conclusions and future developments **55**

Bibliography **57**

List of Figures **67**

List of Tables **71**

Acknowledgements **73**

1 | Introduction

1.1. Automation in the farming industry

The field of Agricultural Robotics has been increasing in the past years. The demand for new technologies has been imposed by different social, political, and economic factors [1]; this is an indication of how relying on human labour is not safe for the agricultural chain.

1.1.1. Arising issues in the agricultural sector

Since the global population is increasing, with a projection of 9.7 billion people by 2050, agricultural production will need to increase by at least 70% from current levels to serve nutritional trends. This is bringing our planet's health under even more stress [2]: it is well known that climate change is becoming a growing threat to agricultural systems as they operate under progressively hostile conditions like erratic rainfalls, drought spells, and floods [3].

Global food security mainly depends on the price of fossil fuels to manufacture fertilizers and pesticides, run tractors, heat greenhouses, etc. Russia and Ukraine account for 73% of the sunflower oil trade (mainly from Ukraine). Russia is the world's largest fertilizer exporter, the world's 2nd largest oil exporter, and the world's 1st natural gas exporter. The country accounts for 10% of global nitrogen fertilizer exports, 10% of global phosphate fertilizer exports and 17% of global potassium fertilizer exports [4]. They also account for 12% of all calorie exports traded internationally. These two countries represent 23% of global wheat exports as well as 16% of global corn exports, which corresponds respectively to the 7% and 3% of global consumption [4]. Today, the war in Ukraine and its consequences prove once again the fragility of our globalized agricultural and food systems.

The recent COVID-19 pandemic introduced a new problem by imposing restrictions on travelling. Let us consider that more than a quarter of all Italian food is produced by the over 370,000 seasonal workers who travel to the country each year [5]. But this is not to be considered circumscribed to the pandemic problem: according to the European

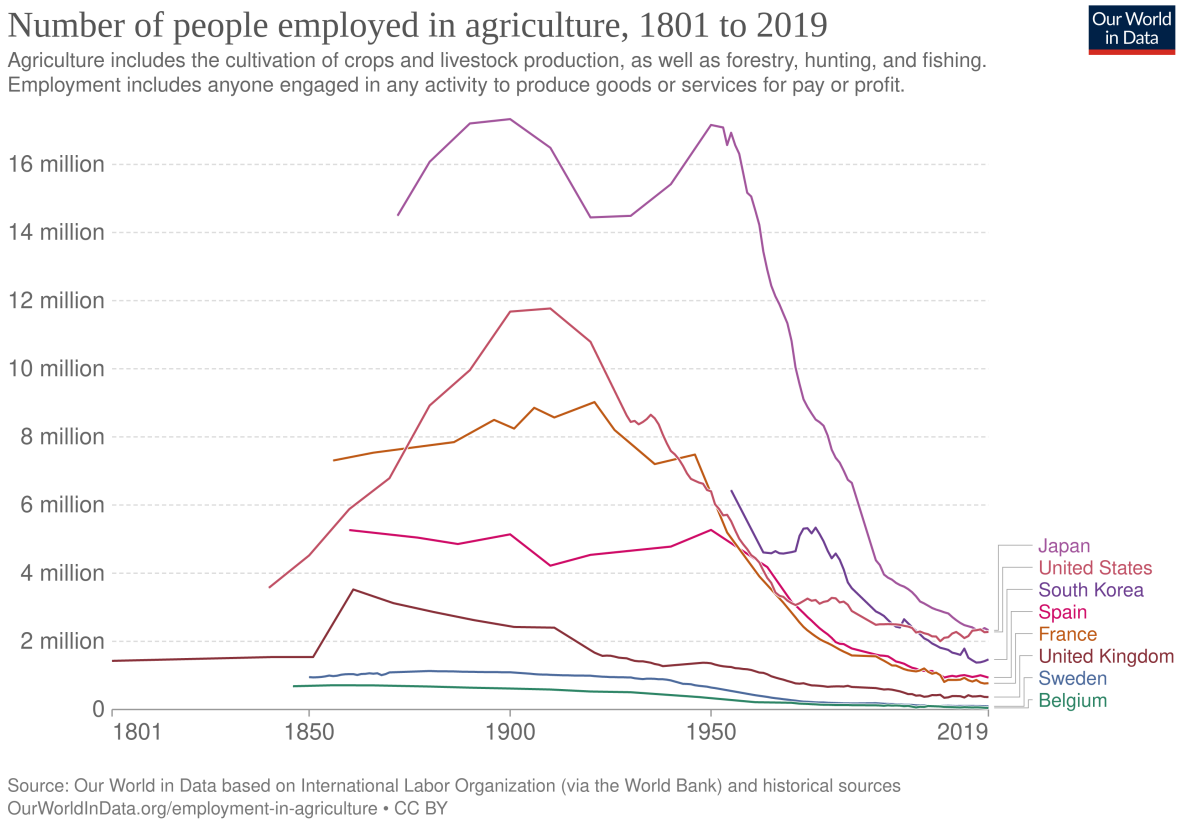


Figure 1.1: Number of people employed in agriculture [8]

Commission president, Europe and the rest of the world must prepare all sectors to cope with an "era of pandemics" [6].

Being an agricultural worker comes therefore with many more risks. The labour shortage was already a growing problem for the past decade: a study from the International Labor Organization (ILO) shows that the manpower of agricultural workers dropped from 81.0% to 48.2% in developing countries and from 35.0% to 4.2% in developed ones since 2014 [7]. A decline of 12.8% is observed in the European agriculture sector alone.

1.1.2. Robots in agriculture

The objective of agricultural robotics is to help the sector in its efficiency and in the profitability of the processes. In other words, mobile robotics (and manipulator robotics) works in the agricultural sector to improve productivity, specialization, and environmental sustainability. The incorporation of robotics in agriculture improves both productivity and working conditions. Intelligent systems are becoming the ideal solution to drive

precision agriculture [9]. Agricultural robots are increasing production yields for farmers in various ways, automating slow, repetitive, and dull tasks [10].

The quality of the agri-food chain is one of the excellence of Made in Italy, but to maintain this primacy, especially in the current historical period, traditional strategies in agriculture must be accompanied by those of Agriculture 4.0.
Sigma Consulting describes Agriculture 4.0 as a synergy of the use of various innovative digital technologies such as Internet of Things (IoT), 5G Radio, Big Data Analytics, Artificial Intelligence, and Robotics, which offer the possibility to expand the data valorization approach to multiple company functions (e.g. logistics, planning, management control) and, to the entire agri-food chain, aimed at improving the yield and sustainability of agricultural activity, production and processing quality, social conditions and the environmental impact of the entire agri-food chain [11].

Therefore, global spending on "smart" agriculture, including AI and machine learning, is projected to triple to $15.3 billion by 2025 [12]. The market size of AI in agriculture should expect a compound annual growth rate (CAGR) of 20%, reaching $2.5 billion by 2026 [13].

1.2. Case study: strawberries

Collaborative robots are now commonly used in fruit harvesting or insect grafting and cultivation, where Artificial Intelligence provides predictive data to optimize farms and plantations [9].

In the early 2010s, while robots were already incorporated in many areas of agricultural work, they were still largely missing in the harvesting of various crops. This started to change as companies began to develop robots that complete more specific tasks on the farm. The major concern over robots picking fruit comes from harvesting soft crops such as strawberries which can easily be damaged or missed entirely [14, 15].

Strawberries are popular fruits consumed in almost all parts of the globe, and this expansion is a result of the fruit adaption capability that makes cultivation possible in multiple climates. The efforts of producers and scientists were also aimed to expand strawberry production and commercialization, generating adaptive systems that allow the cultivation of fruit to specific conditions of each region. Furthermore, the production and commercialization of strawberries are present in 76 countries, reaching a global production of 7.7 million tons in 2013. This consumption has increased in the 21st century and has been accompanied by technological innovations that allow the availability of strawberries all

Figure 1.2: Dyson strawberries poly-tunnel farm [20]

year long, thus feeding a continuous market demand [16].
In the European sphere, Italy is the fourth country for strawberry production (around 140,000 tons), on a surface of 3,700 ha, 80% of which takes place inside covered spaces. Greenhouses are located in the northeast of the Emilia Romagna Region, one of the most important horticultural areas in Italy, both for the variety and quality of the products [17].

Labour represents the largest cost and also a vast operational uncertainty for strawberry farmers [18]: labour costs for the domestic strawberry industry approach about $1 billion annually, and equipment costs are competitive compared to human harvest ones [15]. Therefore, automated fruit picking is desirable. However, despite extensive research over the past four decades, there are no mechanical harvesters for the fresh market commercially available, for strawberries and many other crops [19].

1.2.1. Robots challenges in strawberry farms

Robotic picking at a commercial level needs to overcome problems that are mended in hand-picking due to human expertise. Selective harvesting is a well-known and studied topic. Parsa et al. [21] already exposed a work reasoning on these scientific research questions in the specific strawberry case. First of all, berries are easily damaged and bruised. Secondly, this kind of harvesting requires highly selective procedures, since these fruits tend to ripe unevenly; as a result, at a given time, berries exhibit large variations

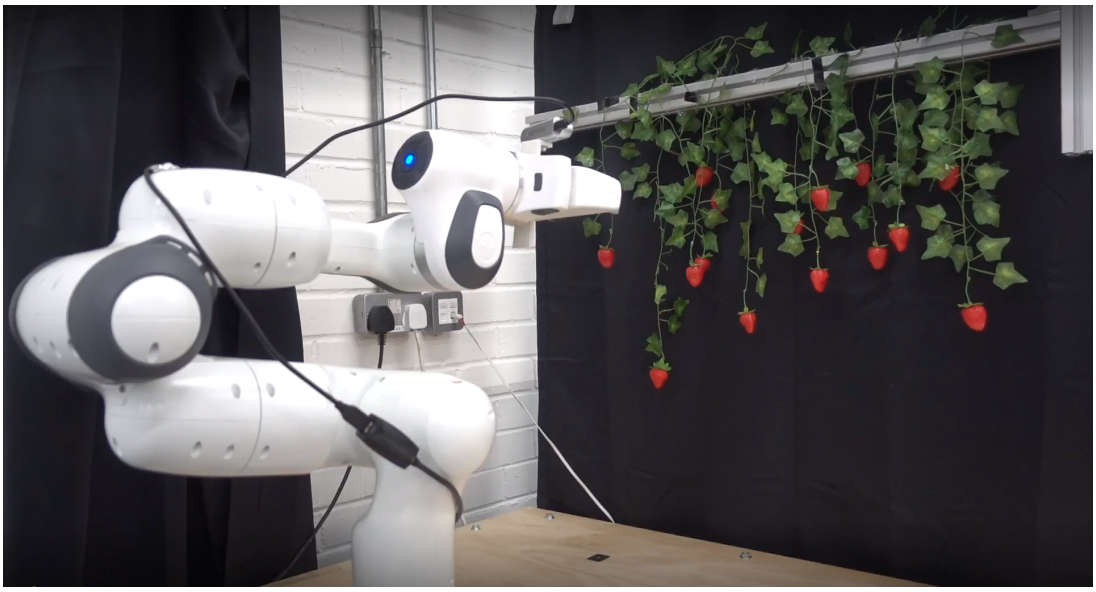


Figure 1.3: Franka configuration, in the robotics laboratory of the Univesity of Lincoln

in colour and size. Finally, strawberries tend to grow in clusters, which makes it hard to identify and pick them individually [22].

So a harvesting robot needs to be a tightly integrated system, incorporating advanced features and functionalities from numerous fields, including navigation, perception, motion planning, and manipulation. These robots are also required to operate at high speed, with high accuracy and robustness, and at a low cost, all features that are especially challenging in unstructured environments, such as strawberry farms [23].

1.2.2. Picking scheduling decision problem

Scheduling of the order of picking is needed to collect a set of objects.
Many Computer Vision (CV) tools allow a machine to identify strawberries in an image. Nowadays, AI techniques can even classify berries as ripe or unripe [24].

Choosing a random target from the objects identified as pluckable might lead to a failure in the picking. It would be simpler to raise an isolated strawberry, but they grow in clusters. Accordingly, they are often occluded with respect to the robot view by leaves or other berries; hence these fruits do not share reachability easiness.
The human eye is able to identify which element is better to take first. It can also plan the harvesting of a whole cluster, taking into consideration how plucking one strawberry could ease the visibility of another one, and so on.

Notwithstanding how teaching a machine to output an optimized order of the raise would

complete the automation of fruit harvesting, the state of the art still lacks research on AI techniques for scheduling fruit picking.

1.3. Thesis research objectives and achievements

This thesis work was carried out at the Intelligent Manipulator Lab of the University of Lincoln. The problems addressed in the research are:

- identification of strawberries from an image;
- classification of ripe and unripe strawberries;
- classification of occlusion properties of ripe strawberries;
- picking scheduling decision;
- motion planning for reach-to-pick.

The novelty of this work stands in the proposal of a deep model for outputting the scheduling order of strawberry picking starting from information about strawberries in an image.

A Franka Emika Panda Manipulator with an eye-in-hand RealSense camera was used. To formulate an initial simple problem, these constraints were followed: the robot returns to its home position after arriving at each berry; the images collected from the camera are exploited only before any trajectory generation, so there is the sole possibility of frontal view perception.

To obtain an optimal harvesting order, a dataset of human-chosen scheduling was used, so that the decision of an individual could be mimicked. Two heuristic scheduling computations are also proposed, estimated using Euclidean distances between strawberries, and a weighted score to represent the occlusion properties.

During the testing of the in-development work, it was noted that after the picking of a single berry, the configuration of the cluster might change, since the manipulator might collide with leaves or other fruits. Although being able to output directly the whole sequence of order of picking is an interesting challenge, it could be fairly important to evaluate just which is the easiest element to be picked.

1.4. Thesis structure

Chapter 2 - State of the art: Overview of the state-of-the-art technologies used for motion planning in a reach-to-pick task, and strawberry visual perception.

Chapter 3 - Deep Probabilistic Movement Primitives for Motion Planning: Explanation of the theory that builds the Deep Probabilistic Movement Primitives model and proposed improvements to this approach.

Chapter 4 - Scheduling decision: Report on the object detection and Graph Attention Network models for the identification of the strawberries and the scheduling prediction.

Chapter 5 - Experimental Setup: Description of the trajectories and annotated images datasets and their preprocessing. Illustration of the integration of the models for the testing of this research in a laboratory.

Chapter 6 - Results and Discussions: Presentation of the results obtained for the various models of strawberry detection and scheduling prediction and comments on the trending of these.

Chapter 7 - Conclusions and future developements: Summary of the main topics of this thesis work and proposals on possible next steps.

2 | State of the art

This chapter reports a detailed overview of the state of the art of technologies used in this specific work: trajectory planning for a reach-to-pick task, strawberries perception and classification, and deep learning models to handle graph data.

2.1. Deep Learning in Motion Planning for Reach-to-pick task

Arisen as a subfield of Artificial Intelligence (AI), Machine Learning (ML) is the *field of study that gives computers the ability to learn without being explicitly programmed.* [25]; in a few words, it uses data to gain models. In turn, Deep Learning (DL) is a class of machine learning algorithms that learns data representation directly from data, by progressively extracting higher-level features from the raw input. ML and DL processes make use of Neural Networks (NN), a very complex mechanism made of simple neurons organized in layers. An example of NN is in Figure 2.1
As will be described below, research on motion planning started way before the introduction of AI.

Learning from demonstrations (LfD) is a well-established technique for teaching robots how to perform useful tasks. The basic idea is that the robot learns behaviour from one or several demonstrations performed by a teacher, most often human [26]. In the task of trajectory generation, this has been exhaustively exploited.
For example, **Movement Primitives (MPs)** are commonly used for representing and learning basic movements in robotics. MP formulations are compact parameterizations of the robot's control policy. Modulating their parameters permits imitation and reinforcement learning as well as adapting to different scenarios [27].
The control policies of a fundamentally different approach to motion representation, based on nonlinear dynamic systems as policy primitives, were termed **Dynamic Movement Primitives (DMPs)**. DMPs are based on methods of second-order differential equations

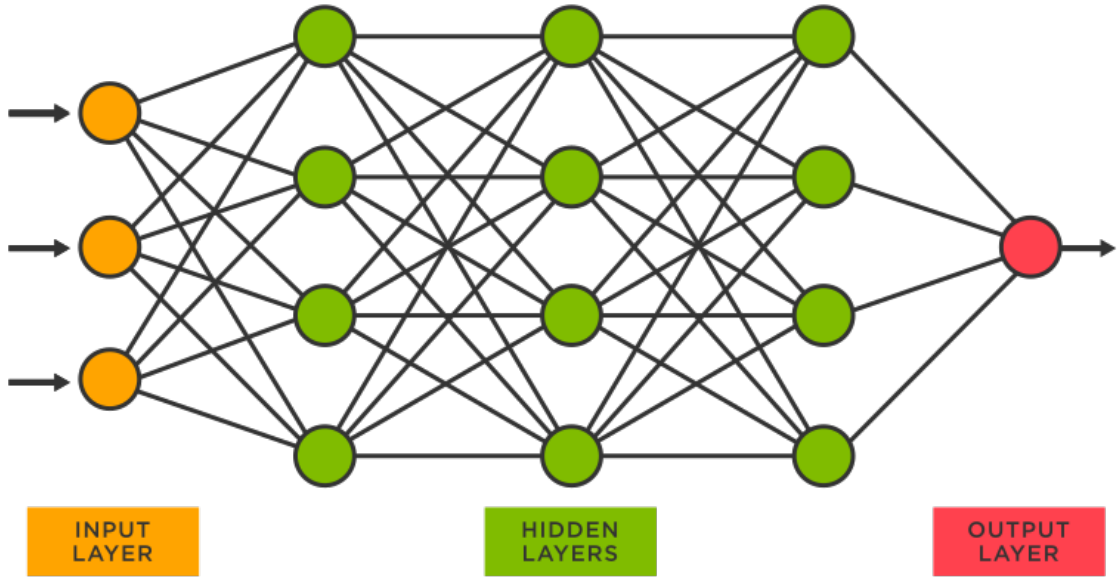


Figure 2.1: Fully-connected Neural Network

to encode the properties of the desired motion. Equations have been developed for periodic and discrete movements. One of the most critical advantages of DMPs is the ability to take into account perturbations and include feedback terms, which can be added to change the timing and/or avoid some areas of the workspace [28].
Probabilistic movement primitives (ProMPs) was introduced as a general probabilistic framework for representing and learning MPs; it is a distribution over trajectories. Since a trajectory distribution can also encode the variance of the movement, a ProMP can often directly encode optimal behaviour in stochastic systems. Moreover, a probabilistic framework allows the modelling of the covariance between trajectories of different degrees of freedom, which can be used to couple the joints of the robot [27].

Recently, some LfD approaches were proposed directly mapping the visual perception into the learned trajectory. **Deep Movement Primitives (deep-MP)** captures the correlation between the visual information and the corresponding manipulation trajectories, for complex trajectory/path planning tasks [29]. This work was extended with **Deep Probabilistic Motion Planning (d-PMP)**, an approach that maps the visual sensory information into a distribution of robot trajectories, generating distributions of trajectories by the corresponding mean and variance [30]. This probabilistic approach to Deep LfD outperforms the state-of-the-art method in a series of real robot tasks of mock strawberry picking.

2.2. Object Detection for strawberries identification and classification

Object detection is a key field in artificial intelligence, allowing computer systems to "see" their environments by detecting objects in visual images or videos. It is one of the fundamental problems of computer vision, forming the basis of many other downstream computer vision tasks such as instance and image segmentation, image captioning, object tracking, and more [31]. The goal of object detection is to predict a set of bounding boxes and category labels for each object (or in this case fruit) of interest.

Fruit detection and classification remain challenging due to the form, colour, and texture of different fruit species. Classification of fruit is a relatively complex problem due to the vast number of varieties. In species and varieties, significant variations in appearance occur including irregular forms, colours, and textures. Moreover, images are easily blurred by natural lighting, and a small change in camera view can have a great impact on depth perception and occlusion. This weakness led to the absence of real-life implementations of multi-class automated fruit classification systems [32]. In this work, the classification task is limited to one fruit - the strawberry - with multiple classes to distinguish different properties of occlusion of the berries.

Object detection has evolved over the past 20 years. It can be performed using either traditional image processing techniques or modern deep learning networks. The pioneering work that started the development of traditional object detection methods is **Viola-Jones Detector** (2001) [33]; other milestones are from **HOG Detector** (2006) [34], a popular feature descriptor for object detection in computer vision and image processing, and **DPM** (2008) [35] with the first introduction of bounding box regression. Image processing techniques generally do not require annotated images (where humans labelled data manually for supervised training) for training and are unsupervised in nature. These techniques are restricted to multiple factors, such as complex scenarios without unicolour background, occlusion, illumination and shadows, and clutter effect [31].

Deep Learning methods depend on supervised or unsupervised learning, meaning they can infer a function both from a finite set of labelled or unlabelled data. Specifically designed networks to process pixel data are the Convolutional Neural Networks (CNN); convolution is a mathematical operation $*$ on two functions (f and g) that produces a third function that expresses how the shape of one is modified by the other. Note that the performance of these DL models is limited by the computation power of GPUs.
In general, DL-based object detectors extract features from the input image or video

frame. An object detector solves two subsequent tasks: find an arbitrary number of objects (possibly even zero); classify every single object, and estimate its size with a bounding box. To simplify the process, those tasks can be separated into two stages. Other methods combine both tasks into one step to achieve higher performance at the cost of accuracy. The two-stage architecture involves object region proposal with conventional Computer Vision methods or deep networks, followed by object classification based on features extracted from the proposed region with bounding-box regression. These methods achieve the highest detection accuracy but are typically slower, because of the many inference steps per image. Various two-stage detectors include region convolutional neural network (**RCNN**) [36], with evolutions Faster R-CNN [37] or Mask R-CNN [38]. The latest evolution is the granulated RCNN (G-RCNN) [39]. Chen et al. [40] employed a faster region-based convolutional neural network for fruit detection in orchards. Villacrés, and Auat Cheein [41] went ahead to employ Faster R-CNN in a more unique and modified way for the purpose of cherry detection and characterization.

Two-stage object detectors first find a region of interest and use this cropped region for classification. However, such multi-stage detectors are usually not end-to-end trainable because cropping is a non-differentiable operation. On the other end, one-stage detectors predict bounding boxes over the images without the region proposal step. This process consumes less time and can therefore be used in real-time applications. Such detectors prioritize inference speed but are not as good at recognizing irregularly shaped objects or a group of small objects [31]. The most popular one-stage detectors include the **YOLO** [42], **SSD** [43], and **RetinaNet** [44]. Kirk et al. [45] proposed this model for a rapid and robust outdoor fruit detection system combining bio-inspired features with one-stage DL networks. The latest real-time detectors are YOLOv7 (2022) [46], YOLOR (2021) [47] and YOLOv4-Scaled (2020) [48]. YOLO has earned popularity due to its ability to detect tiny objects of which various researchers utilized different versions for the purpose of fruit detection. For example, a mango-based model was proposed by Jia et al. [49] for a task of real-time fruit detection and orchard fruit load estimation; this is the integration of the YOLO V3 and YOLO V2 (tiny) networks which form the benchmarking of "Mango Yolo" [50].

Deep learning object detection is significantly more robust to occlusion, complex scenes, and challenging illumination. Unfortunately, a huge amount of training data is required; the process of image annotation is labour-intensive and expensive. For example, labelling 500'000 images to train a custom DL object detection algorithm is considered a small dataset. However, many benchmark datasets (MS COCO, Caltech, KITTI, PASCAL VOC, V5) provide the availability of labelled data. For this work, a new labelled dataset of

images of strawberries is provided, described in chapter 5.2.1. Today, deep learning object detection is widely accepted by researchers and adopted by computer vision companies to build commercial products [31].

To establish a fair comparison between different detectors many metrics were defined over the years. IoU metric evaluates the division between the area of overlap and the area of union. In other words, it evaluates the degree of overlap between ground truth and predictions. It ranges from 0 to 1, where 1 would be a perfect overlap between the ground truth and the prediction.
In classification, a correct detection is termed a true positive (TP), an incorrect one false positive (FP), while a false negative (FN) is a Ground-truth not detected. With these values, we can evaluate precision and Recall.

$$precision = \frac{TP}{TP + FP} \tag{2.1}$$

$$recall = \frac{TP}{TP + FN} \tag{2.2}$$

Precision indicates the proportion of correct positive identification. The recall relates to the proportion of actual positives that were correctly identified.
When plotting the precision-recall curve evaluated at an IoU threshold, we get the Average Precision (AP):

$$AP_\alpha = \int_0^1 p(r)dr \tag{2.3}$$

where "α" is the threshold value, p is the precision, and r is the recall value.
When considering a multi-class object detector the mean average precision (mAP) gives the mean AP across all classes.

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{2.4}$$

where "AP" is the average precision of each "q" class and "Q" is the number of classes.
In recent years, object detection performance has improved significantly, with an increase from 30% mean average precision (mAP) to more than 90% in 2018 on the PASCAL VOC benchmark (measured on the PASCAL VOC object detection public dataset). The main driver of these improved results was the use of Deep Learning. With the development of new technologies that allow faster and easier pipelines and the availability of large-scale open data sets, the development of powerful models has become a reality [51].

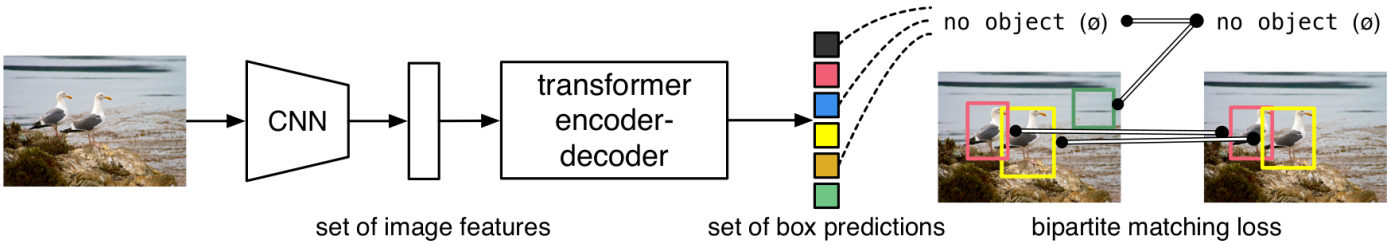


Figure 2.2: DETR Architecture, from the paper "End-to-end object detection with transformers" [53]: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "no object" class prediction.

This thesis work uses a couple of famous architectures:

- **Detectron2** [52]: based on Masked-RCNN, it has become the standard for instance segmentation. It is also able to detect keypoints for human pose estimation. It was fine-tuned for strawberry segmentation and keypoint estimation [24].
- **DETR** [53]: consists of a convolutional backbone followed by an encoder-decoder Transformer which can be trained end-to-end for object detection. It greatly simplifies a lot of the complexity of models like Faster-R-CNN and Mask-R-CNN, which use tools like region proposals, non-maximum suppression procedures, and anchor generation. Moreover, DETR can also be naturally extended to perform panoptic segmentation [54], by simply adding a mask head on top of the decoder outputs.

2.3. Graph Neural Networks for scheduling fruit harvesting

With a model designed for an object detection task, it's difficult to learn a scheduling order of picking - for example, by classifying fruits as first-to-be-picked or not - from a visual input. Instead, every berry should learn its representation in the contest of the cluster, to acquire knowledge on its easiness-of-picking with respect to the other strawberries in the neighbourhood. This is the field of Graph Neural Networks. In this section, there is an overview of the evolution of these networks and their purpose.

Any set of objects and the connections between them is naturally expressed as a graph. In numerous scientific tasks, the data is suitably represented by a graph structure. This ap-

plies to telecommunication networks, but also to 3D meshes, molecules or other biological networks, and so on [55].

Graph Neural Networks were introduced in 2006 by Scarselli and Gori [56] and in 2008 [57], setting the mathematical foundations for the Graph Neural Network. It can be seen as a recursive neural network - meaning a NN that recursively applies the same set of weights on structured inputs- which can directly deal with any cyclic, directed, and undirected graph. A GNN can be used for graph-level, node-level, or edge-level predictions.
Single nodes are defined by a vector with a set dimension, that encodes features (e.g. colour for an image's pixel, an object's pattern, etc). The edges have a representation vector, too, to describe the relationship among nodes (e.g. "distance" between them). The main idea behind GNNs is to propagate nodes' information among neighbours, and aggregate them to create for each node features enriched with other nodes' representations. It's an iterative process, which propagates the node states until a certain threshold is reached [58]. An illustration of the functioning of these networks is in Figure 2.3

DeepWalk [59] came out in 2014 as a potential improvement for GNN. It's the first graph to implement the embedding method and it is based on representation learning and language modelling — SkipGram model [60]. DeepWalk returns latent representations of an input graph based on nodes' social attributes [61]. The graph is described as $G = (V, E)$, with V vertices and E edges.

From 2014 to 2017, a lot of publications were made in the attempt of applying CNNs to graphs [62]. The reason behind this was to highlight the relationship between the spectral analysis of a graph and the convolution operation; this is possible just by adapting the grid convolution operation in a graph domain [63]. **Graph Convolutional Neural Networks** (GCNs) have been wildly used for object detection and classification tasks; for example, Sajitha et al. [64] introduce a GCN for banana's fruit disease detection and categorization, to overcome the main difficulties with phytopathology.

In 2016, Atwood and Towsley introduced **Diffusion-Convolutional Neural Networks** (DCNN) [65, 66]. This network deals with the graph classification problem by learning a summarization of local information through a diffusion process [67].

To handle graphs with evolving features or connectivities - dynamic graphs -, **Temporal Graph Networks** (TGNs) were introduced, represented as sequences of timed events. Using a combination of memory modules and graph-based operators, it's a computationally efficient approach [68]. Other models had already been introduced to deal with dynamic graphs, represented as a sequence of graph snapshots [69], or that support

a continuous-time scenario [70]. It is shown that such models can be cast as specific instances of TGNs. The applications for this framework are many. Vyas et al. [71] proposed a novel GNN-based solution that learns temporal graph structures and forecasts soil moisture in an end-to-end framework, to schedule irrigation and optimize the use of water.

In the contest of node classification of graph-structured data, Veličković et al. [72] proposed an attention-based architecture called **Graph Attention Network** (GAT). It follows a self-attention strategy, attending to each node's neighbours to compute its representation. The attention architecture is parallelizable across node neighbour pairs. It can be applied to graph nodes with different degrees by establishing arbitrary weights for the neighbours. Moreover, the model can handle inductive learning problems, so it can deal with unseen graphs.

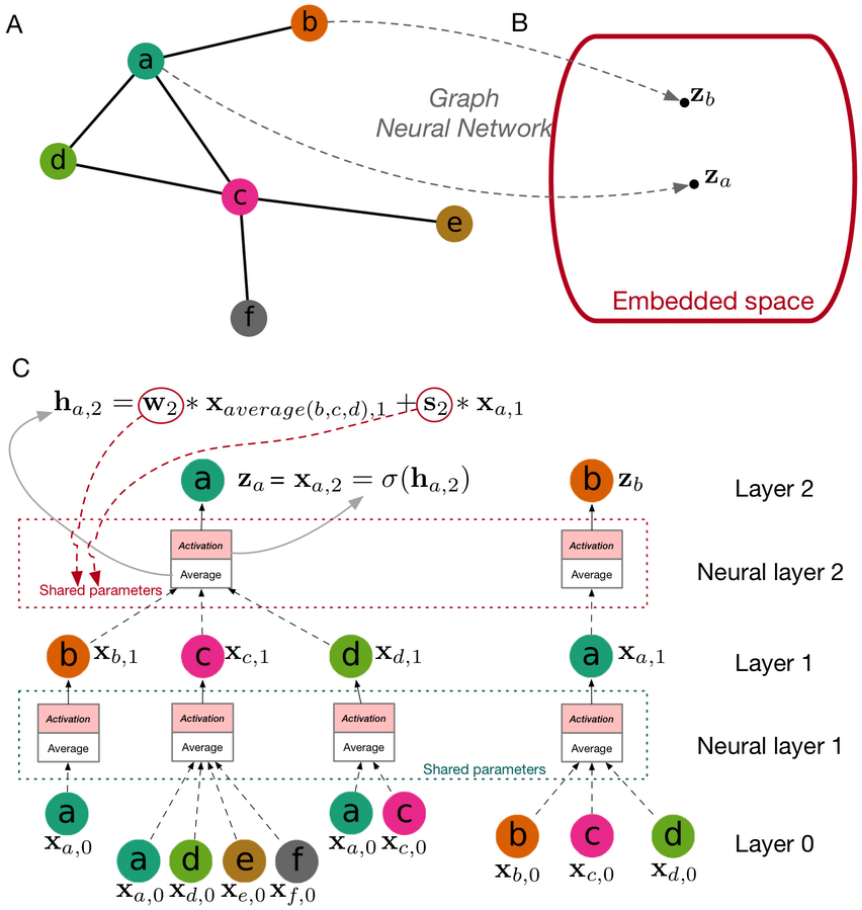


Figure 2.3: Illustration of a graph neural network. (A) A typical example of graph data. (B) The embedding space. In this embedding space, each data point is represented by a vector while the original topological information in (A) is preserved in that vector. (C) The graph neural network for embedding the network in (A). Let's use nodes a and b as examples. The internal properties of each node are considered as the original representations. In each layer, the nodes aggregate information from their neighbours and update the representations with averaging and activation functions. The output of layer 2 is considered as the embedding result in this example. Notice that the parameters within the same layer between different trees are shared so this method can be generalized to the previously unseen graph of the same type. Image and description are taken from Li et. al [73]

3 | Deep Probabilistic Movement Primitives for Motion Planning

Probabilistic Movement Primitives (ProMPs) are a Learning from Demonstration tool for generating a distribution over trajectories. A probabilistic approach can be preferred over a deterministic one in terms of trajectory prediction: starting from a distribution of trajectories, a sample from it can be chosen to meet secondary objectives such as collision avoidance. The basics of Movement Primitives and ProMPs are described in the next section.

Deep Probabilistic Motion Planning has been proposed to generate trajectories in reach-to-pick motion in [30], mapping the visual information into a distribution of effective robot trajectories. A couple of improvements can be easily implemented, which are explained in this chapter.

3.0.1. Fundamentals of Deep-ProMPs

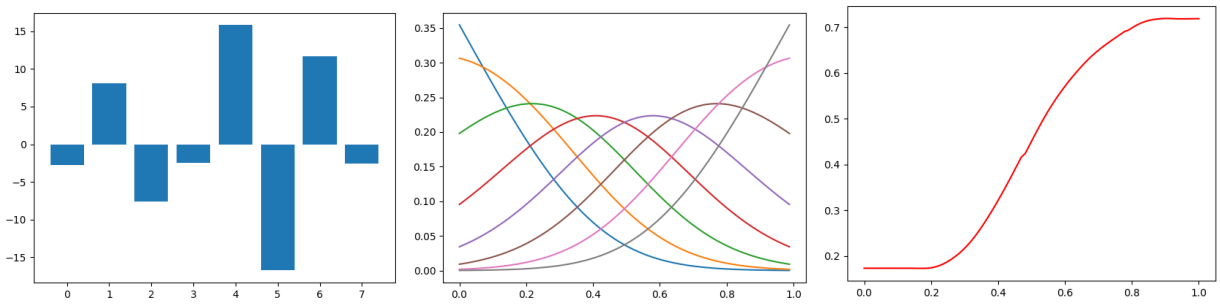
The Movement Primitives model [27] describes a trajectory q as:

$$q = \sum_{i=1}^{N_{bas}} \theta_i \psi i(z(t)) + \epsilon_q \tag{3.1}$$

introducing the trajectory weights $\theta_i \in \mathbb{R}$; the Gaussian basis functions $\psi i(z(t))$; the function for time modulation $z(t)$, $z(t) = \frac{t}{f}$ (with f sampling frequency) if no modulation is needed; the Gaussain noise ϵ_q. In this case, the Gaussian basis functions are chosen for stroke-like movements:

$$\psi_i = \frac{b_i(z(t))}{\sum_{j=1}^{N_{bas}} b_j(z(t))}, \quad b_i(z(t)) := exp(-\frac{(z(t) - c_i)^2}{2h}).$$

Equation 3.1 can be expressed in matrix form $q_t = \Psi_t^T \Theta + \epsilon_q$, defining $\Psi_t := (\psi_1(z(t)), ..., \psi_{N_{bas}}(z(t))) \in \mathbb{R}^{N_{bas} \times 1}$ and $\Theta_t := (\theta_1, ..., \theta_{N_{bas}}) \in \mathbb{R}^{N_{bas} \times 1}$.



(a) (b) (c)

Figure 3.1: Starting from the left: compact weight representation of a trajectory for each basis function; Gaussian basis functions; trajectory of a single DoF

Since this is valid for every joint of a robot, it is then defined also $\Omega := (\Theta_1, ..., \Theta_{N_{joint}}) \in \mathbb{R}^{N_{bas}N_{joint} \times 1}$ and $\Phi := [\Psi_1, ..., \Psi_T]^T \in \mathbb{R}^{TxN_{bas}}$.
Figure 3.1 shows an example of how a trajectory can be retrieved for a single joint from a compact weight representation with specific basis functions.

Probabilistic Movement Primitives model [27] introduces the probability of observing a trajectory q_t as

$$p(q_t|\Theta) = \mathcal{N}(q_t|\Psi_t^T\Theta, \Sigma_q). \tag{3.2}$$

Being Σ_q always the same for every time step and q_t taken from i.i.d. (independent identical distributions), then the probability of observing a trajectory q is expressed as:

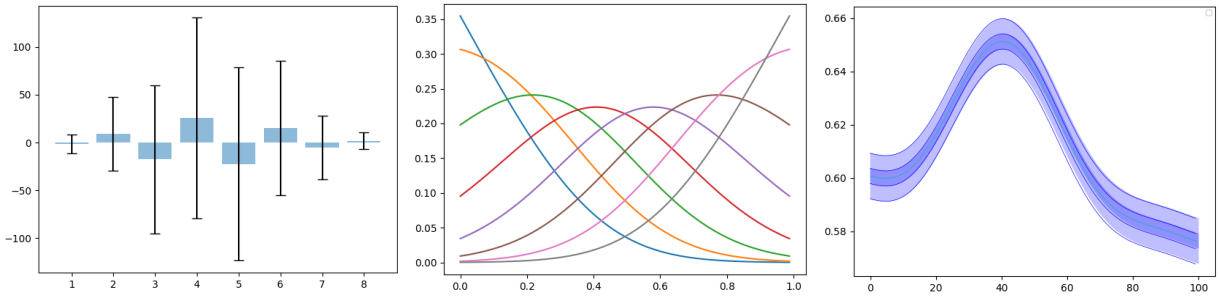
$$p(q|\Theta) = \prod_{t=1}^{T} p(q_t|\Theta).$$

where weights Θ are assumed to be taken from a Gaussian distribution: $\Theta \sim p(\Theta|\rho) = \mathcal{N}(\Theta|\mu_\Theta, \Sigma_\Theta)$, taking $\rho = (\mu_\Theta, \Sigma_\Theta)$. Then, substituting in (3.2) we obtain:

$$p(q_t|\rho) = \int \mathcal{N}(q_t|\Psi_t^T\Theta, \Sigma_q)\mathcal{N}(\Theta|\mu_\Theta, \Sigma_\Theta)d\Theta = \mathcal{N}(q_t|\Psi_t^T\mu_\Theta, \Sigma_q + \Psi_t^T\Sigma_\Theta\Psi_t) \tag{3.3}$$

In Figure 3.2, it's shown an example of how from a weight distribution, with specific basis functions, a trajectory distribution can be retrieved.

A step further is made by Deep Probabilistic Movement Primitives model[30], which learns the relation between the mean vector $\Theta_{mean,j}$ and covariance matrix $\Sigma_{\Theta,j}$ of a trajectory distribution and an input image. This allows the mapping of visual information of a



(a) (b) (c)

Figure 3.2: Starting from the left: weight distribution of a trajectory for each basis function; Gaussian basis functions; trajectory distribution of a single DoF

robot's workspace to the distribution of robot trajectories, for every joint j as:

$$\Theta_{mean,j}, \Sigma_{\Theta,j} = f_j(W_j, I^n, \sigma_j) \tag{3.4}$$

where f_j is a non-linear function of the input image I^n, W_j is the weight parameter and σ_j is the node activation. As it's shown in Equation 3.3, the trajectory distribution can now be retrieved as:

$$q_{mean,j} = \Phi^T \Theta_{mean,j} \quad , \quad \Sigma_{q,j} = \Phi^T \Sigma_{\Theta,j} \Phi \tag{3.5}$$

The model has two parts: the first encodes the high-dimensional input RGB image in a low-dimensional latent space vector using a set of convolutional layers; the second maps the latent embedded vector to the relative ProMPs weights distribution using a Multi-layer Perceptron per each joint. MPL is a neural network connecting multiple layers in a directed graph.

For the encoding part different baselines were tested, such as auto-encoder (AE), Variational AE (VAE), and Conditional VAE (cVAE). AE is an unsupervised artificial neural network composed of an encoder and a decoder. The encoder learns how to efficiently compress and encode data, instead, the decoder learns how to reconstruct the data back from the reduced encoded representation to a representation that is as close to the original input as possible. By design, the model reduces data dimensions by learning how to ignore the noise in the data [74]. A next step is done in VAEs, in which the latent space is represented by the mean and covariance of a distribution so that multiple samples can be generated from it [75, 76]. Further evolution is found in cVAEs, which inserts label information in the latent space to force a deterministic constrained representation of the learned data [77].

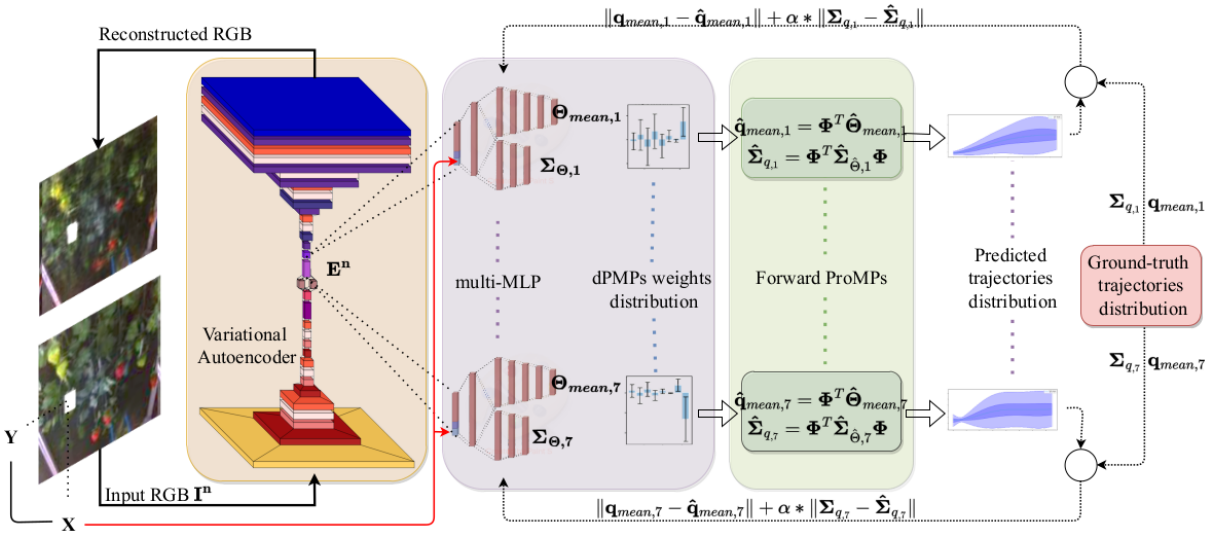


Figure 3.3: d-PMP architecture from [30]

3.1. Improved Deep-ProMP

In the d-PMP paper [30] there are some annotations on possible future works. The solution to a couple of these is explored in this section.

A Root Mean Squared Error loss is used in the training phase of the model to judge the error of the generated trajectory distribution. This should be corrected with a more justified metric such as Kullback-Leibler divergence (KLD), which measures how the predicted probability distribution differs with respect to the ground truth one. Being p and q two normal probabilities distributions defined on the sample space with mean μ and variance Σ, KLD is defined as:

$$D_{KL}(p||q) = \sum_{x \in X} p(x) \log(\frac{p(x)}{q(x)}) = \frac{1}{3}[\log \frac{|\Sigma_q|}{|\Sigma_p|} - k + (\mu_p - \mu_q)^T \Sigma_q^{-1} + tr(\Sigma_q^{-1}\Sigma_p)] \quad (3.6)$$

A visualisation of Kullback-Leibler divergence is in Figure 3.4.

Another improvement is regarding the joint correlation. Indeed there is a trained model for each joint or task space degree of freedom (DoF). As a matter of fact, the model is more accurate if it can output fewer values. To obtain a trajectory for a DoF 36 weights are needed. These are used to reconstruct the mean and covariance of the trajectory distribution. If the model had to output weights for all the DoFs, it would mean 252 values: a number too big for the proposed approach!

However, having an independent ProMP for each degree of freedom does not allow the capturing of potentially important correlations between degrees of freedom, like the cor-

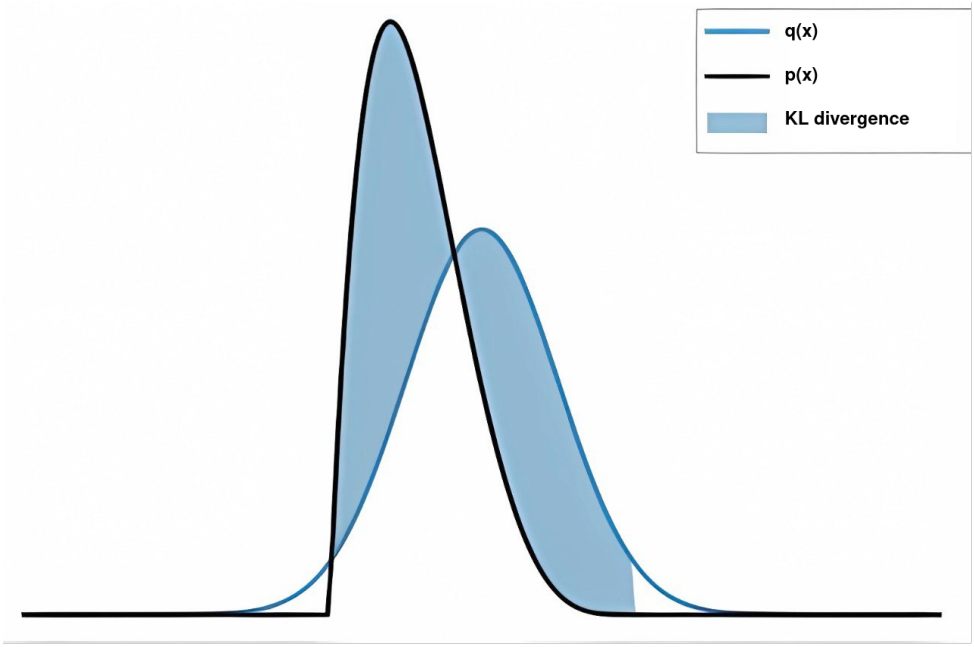


Figure 3.4: Visualisation of KLD between two distributions p and q. Graphic inspired by [78]

relations between joint angles to achieve a desired end-effector pose. In the Probabilistic Movement Primitives paper, the authors propose a way of encoding coupling between joints, to coordinate their movement [27].

$$p(y_t|w) = N\left(\begin{bmatrix} y_{1,t} \\ . \\ . \\ . \\ y_{d,t} \end{bmatrix} \middle| \begin{bmatrix} \phi_t^T & . & . & . & 0 \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ 0 & . & . & . & \phi_t^T \end{bmatrix} w, \sigma_y\right) = N(y_t|\Phi_t w, \sigma_y) \tag{3.7}$$

The problem of the dimension of the model output could be addressed with the right preprocessing of the trajectories' weights. In order to do this, a new dataset is needed. The dataset collection and preprocessing are described in chapter 5.1.
This work was done in collaboration with another MSc student, whose thesis was developed in parallel to the present work. Starting from these improvements, he then proceeded by introducing a novel model to map the visual sensory information into a distribution of robot trajectories.

4 | Scheduling decision

This chapter enters in a detailed explanation of the techniques used to learn harvesting scheduling from perception. First, it introduces the problem of the perception of the fruits. Then, a couple of heuristic scheduling computations are described and confronted with a human-decided harvesting order. Finally, it proposes a deep model for strawberry picking scheduling decision.

4.1. Strawberries Object Detection

In the context of autonomous fruit harvesting, the first goal is to visualize the strawberries. The perception can be done using any vision sensor or camera with an in-eye or in-hand configuration. As shown in Figure 1.3, in this work an in-hand configuration was used.

After the perception, the second step is to identify the berries. Chapter 2.2 shows many techniques that allow the prediction of bounding boxes and category labels. A manually annotated set of images is needed to learn this task. As it is exhaustively described in 5.2.1, the dataset provided contains annotations with information on: bounding boxes, occlusion properties, and human-decided scheduling. This information is on the ripe strawberries only. The occlusion properties are chosen between four options: occluded, occluding, occluded and occluding, or neither. So it can be trained a model that can detect strawberries and classify them by distinguishing occlusion properties or order of picking.

End-to-end Object Detection with Transformer [53] was shown to significantly outperform competitive baselines, viewing object detection as a direct set prediction problem. The innovation stands in the encoder-decoder architecture based on transformers, a deep learning model that adopts the mechanism of self-attention. This self-attention mechanism, a technique meant to mimic cognitive attention first introduced by Vaswani et al. [79], explicitly models all pairwise interactions between elements in a sequence making these architectures particularly suitable for specific constraints of set prediction such as removing duplicate predictions. On the cons, training any transformer-based architecture

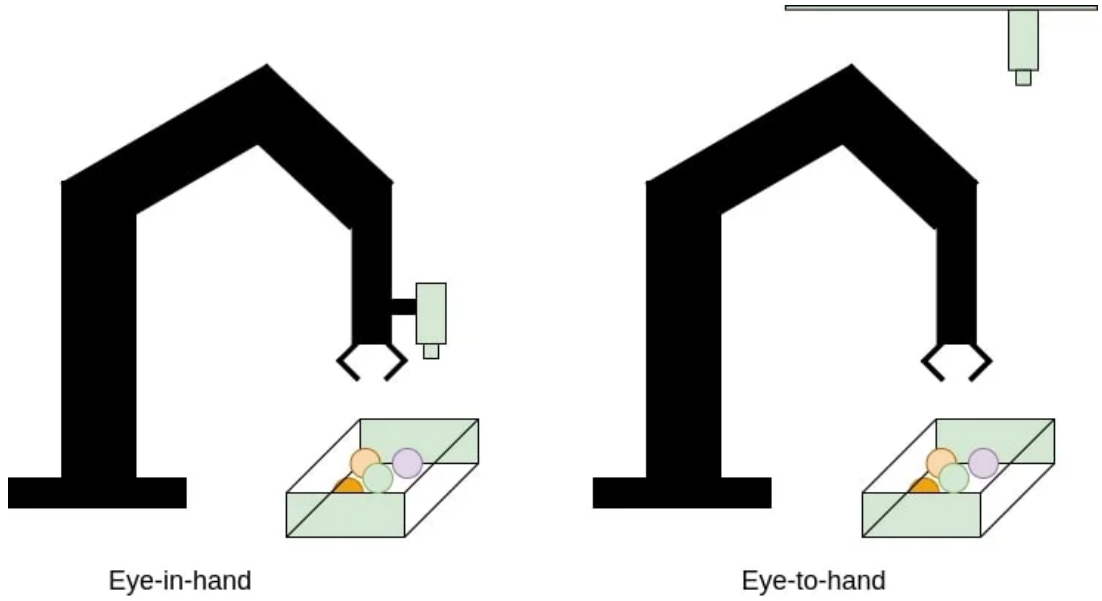


Figure 4.1: Eye-in-hand VS Eye-to-hand configuration [80].

requires a huge amount of data and computational power. To get around this the training was not from scratch, but fine-tuning was done for 300 epochs starting from a DETR-DC5 benchmark model. Giving as input the dataset's images and annotation files, the obtained output is a detection and classification for the strawberries for the 5 classes of the occlusion properties indicated in the data annotation - "occluded", "occluding", "occluded and occluding", or "neither".

This model can now be used for obtaining annotations of bounding boxes and occlusion properties of ripe strawberries starting from "raw" images of strawberry plantations.

Another fine-tuning was executed, this time exploiting the scheduling annotations and treating as classes the scheduling orders. This wasn't a successful attempt: it is difficult to learn the number of order of picking a berry just by confronting it with other images of berries with the same scheduling value. Moreover, as illustrated in Table 5.1, the classes are imbalanced, so it is more different for a model to learn a classification. As shown in 4.4 (b), other issues of this approach were that in the output more strawberries can have the same harvesting number and the scheduling is not guaranteed to start from 1 and have a linearly increasing value. For these reasons, it was decided that it was more rational and interesting to predict scheduling with another technique besides object detection.

The dataset does not have annotations of unripe strawberries present in the clusters. To retrieve such information, a Detectron-2 model [52] was used. Fine-tuned for Strawberry picking point localization ripeness and weight estimation [24], it predicts also bounding boxes and ripeness of berries. The purpose of using this model is to acquire information

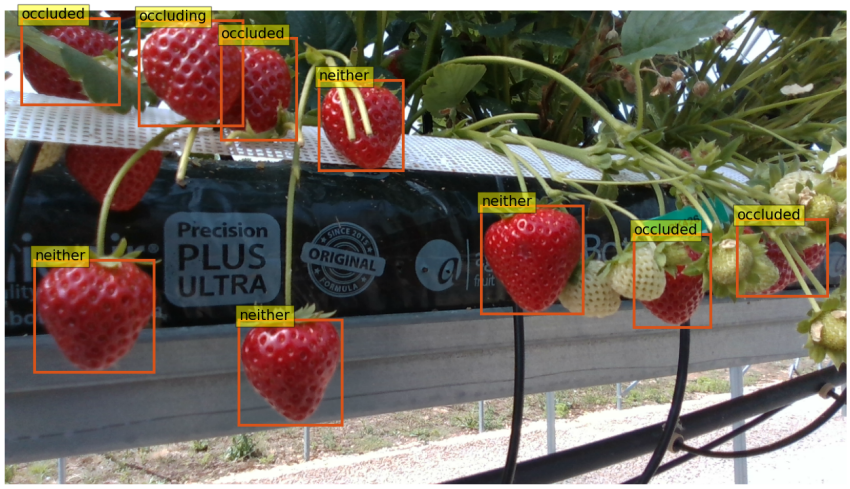


Figure 4.2: Fine-tuned DETR output: strawberries identification and occlusion properties classification.

Figure 4.3: Fine-tuned DETR output: strawberries identification and scheduling identification. The integer numbers represent the predicted order of picking, the other number is the probability of the prediction

about all the fruits in the images so that all factors are taken into account for a more complete prediction of the scheduling order. Note that the unripe berries do not have occlusion information: a new dataset with such annotations would be needed to train a model that detects this information. This is not seen as a problem since the targets for the harvesting are of course just the ripe strawberries.

Figure 4.4: From the paper: Strawberry picking point localization ripeness and weight estimation [24], where ripe strawberries are labelled as "pluckable", and unripe as "unpluckable"

4.2. Heuristic scheduling computations

In the dataset, the scheduling is a human choice. The human makes this decision taking into consideration:

- the level of isolation of the strawberries in the cluster
- the depth of the strawberries in the image
- the occlusion of the strawberries:
  - by a leaf
  - by a stem
  - by another strawberry
- how the picking of a strawberry can ease the visibility and reachability of another one

A con of human annotation is that it's difficult to check the consistency of the dataset. For example, two individuals could make a different choice on the same configuration.

To compute the order of picking starting from bounding boxes and occlusion properties

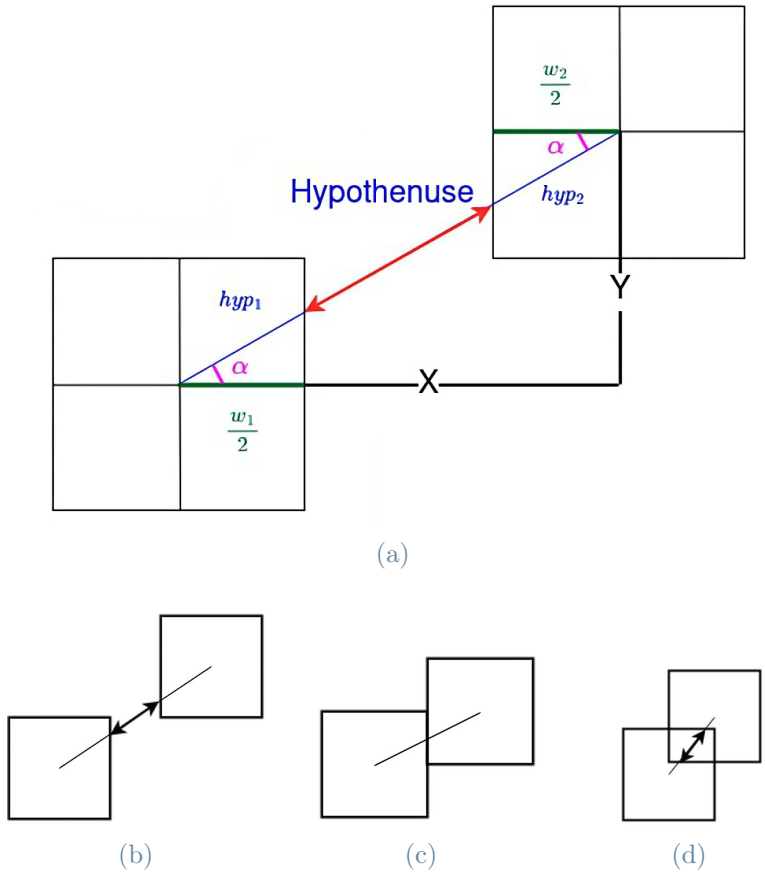


Figure 4.5: Distance between bounding boxes for boxes' distance computation

without the need for a neural network, two heuristic scheduling methods are proposed. They will be used as a comparison with the dataset and with the scheduling GNN model. For both heuristic computations, it is necessary to calculate the distances between the strawberries' bounding boxes. Figure 4.5 (a) shows a visualization of this distance D, represented by the red arrow. Knowing the width w, height h, and left-down corner coordinates (x_{min}, y_{min}) for each box, the distance between the centres is the $Hypothenuse$ indicated in the figure. From the $Hypothenuse$ it can be estimated the $\cos\alpha$.

$$x_c = x_{min} + \frac{w}{2}, \quad y_c = y_{min} + \frac{h}{2}$$
$$Hypothenuse = \sqrt{X^2 + Y^2}, \quad X = |x_{c_1} - x_{c_2}|, \quad Y = |y_{c_1} - y_{c_2}|$$
$$\cos\alpha = \frac{X}{Hypothenuse}, \quad hyp_1 = \frac{w_1/2}{\cos\alpha}, \quad hyp_2 = \frac{w_2/2}{\cos\alpha}$$

The distance between the boxes can then be retrieved as:

$$D = Hypothenuse - hyp_1 - hyp_2$$

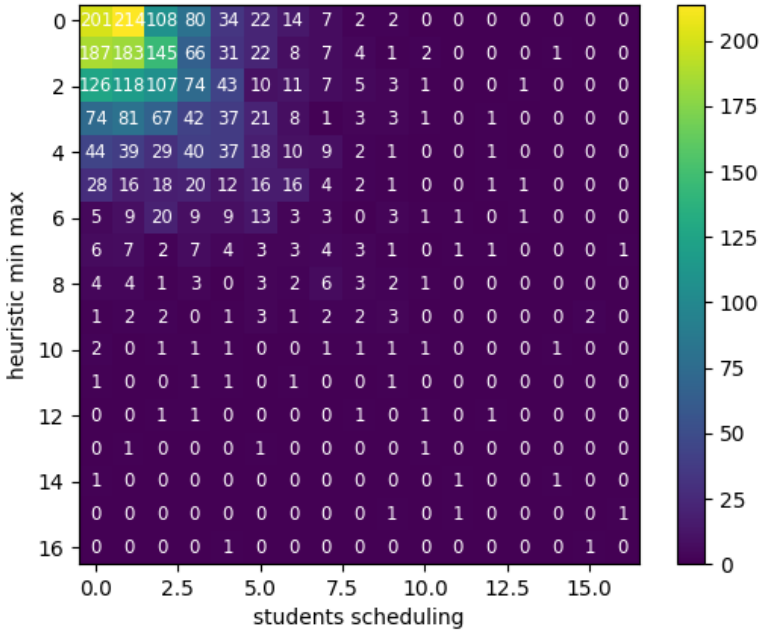
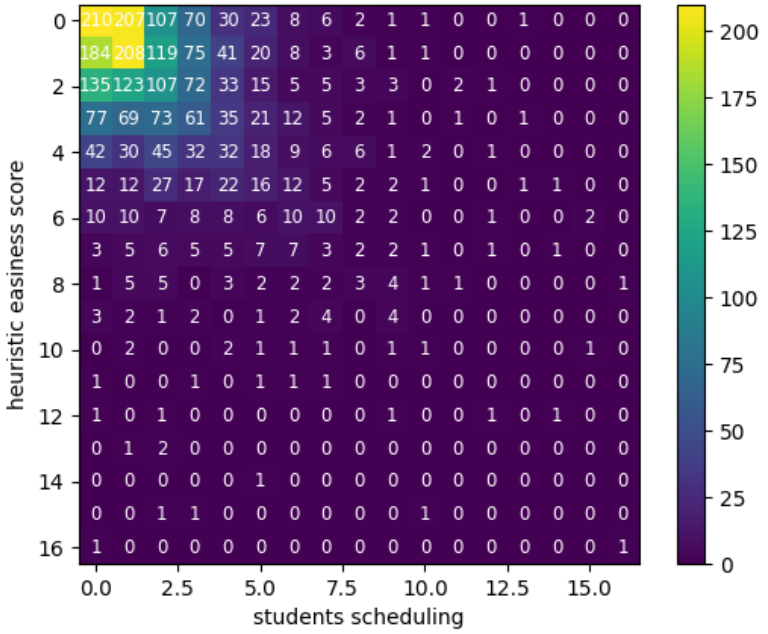


Figure 4.6: Comparison between students scheduling and heuristic min-max computation over the dataset

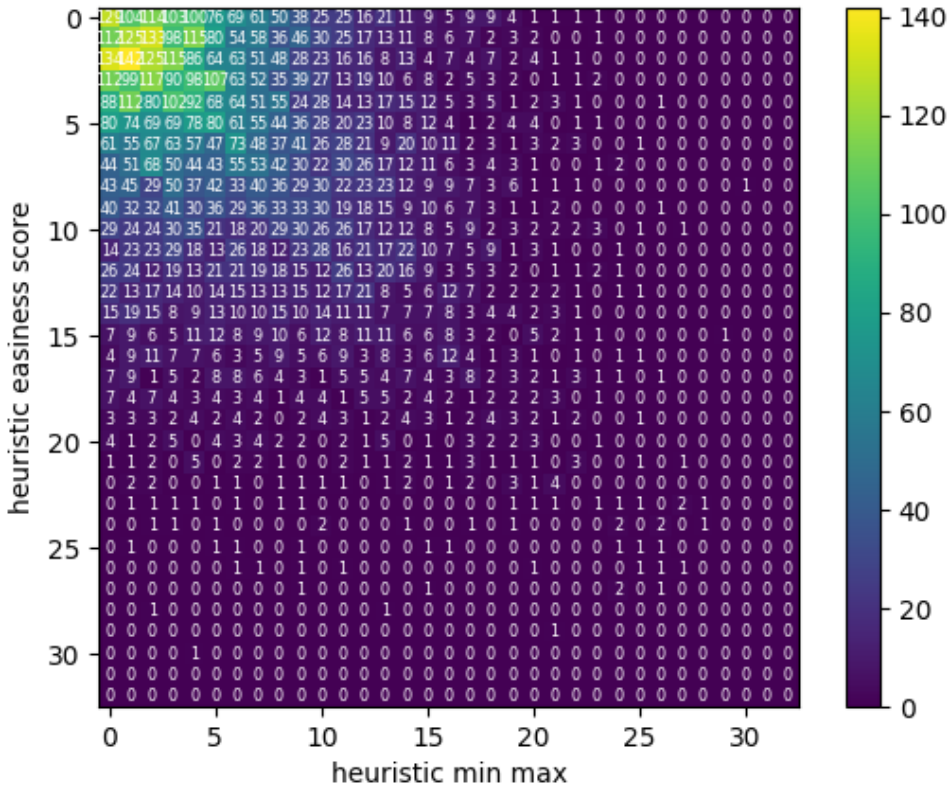
One of the reasons behind taking this distance computation instead of the simple distance between centres is that including the size of the bounding boxes can help with giving an idea of the depth level of the strawberries.

Heuristic min-max

The first-to-be-picked is the most isolated and non-occluded strawberry.
For each fruit, the distance between all the other ripe and unripe berries in the image is computed. The minimum of these distances is added to a vector $min\_distances$. The ripe strawberries are divided into four groups relative to the four occlusion properties: non-occluded, occluding, occluded/occluding, and occluded. Following this order, for each of the occlusion groups, the scheduling is subsequently decided by ordering the strawberries according to the decreasing maximum minimum distance. Then a harvest schedule can also be easily estimated for unripe berries.
In Figure 4.6 we can visualize with a heatmap the correspondence between the scheduling just computed and the human-chosen scheduling of the dataset. Every element of the cell $c_{i,j}$ indicates the number of strawberries with heuristic scheduling label i and human-decided scheduling label j. This means that if the two schedulings coincided, the heatmap would be a diagonal matrix.



(a)



(b)

Figure 4.7: Comparison between (a) students scheduling choice and heuristic easiness score computation, and (b) heuristic min-max computation and heuristic easiness score computation over the dataset

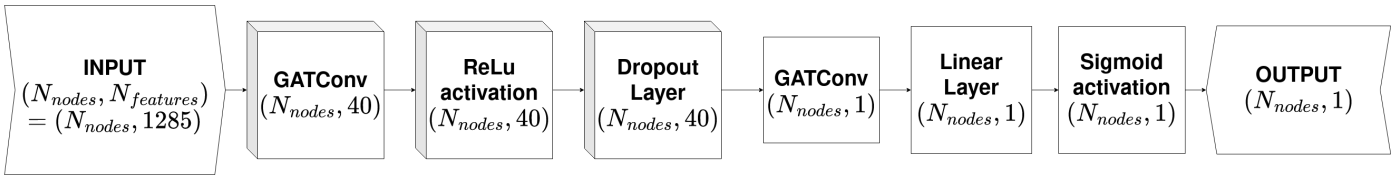


Figure 4.8: Visualization of GAT scheduling classification model

Heuristic easiness score

This second heuristic scheduling computation assigns an easiness score to each strawberry and then retrieves the scheduling, ordering berries from easiest to most difficult to pick. The easiness score is a simple multiplication between the distance and the occlusion scores. The distance score is computed as in 4.2.
The occlusion score is chosen to be 1 if a strawberry is not occluded; 0.7 if it's occluded by a leaf, meaning that it was annotated as occluded but it only has positive distances from the other boxes; 0.9 times the percentage of the free area of the bounding box if the occlusion is by another berry. Figure 4.5 shows an example of two boxes with a positive distance (b), with null distance (c) or with a negative distance (d). In the latter case, the percentage of occlusion is computed as the fraction between the overlapping area and the bounding box area.

Figures 4.7 is similar to Figure 4.6, but 4.7 (b) takes into consideration the unripe strawberries, too.

4.3. GAT model

To predict the picking order, the idea is to make each strawberry learn its representation in the context of the cluster. For this type of learning, Graph Neural Networks are used. In fact, the concept behind these networks is that in every layer each node sends its representation to its neighbours, and aggregates its features with the ones received: so it becomes aware of the presence and the characteristics of other nodes. Moreover, with self-attention, it can be learnt how each neighbour is important for every node.
As explained in 5.2.2, the dataset is inputted in the form of a graph. The nodes are the strawberries represented by the bounding box coordinates, an occlusion weight, a ripeness indicator, and compressed information of the image patch inside the bounding box. So a graph neural network can be trained to obtain a model that can output a scheduling sequence.

Two main attempts were followed. The first four layers are the same for all of the trials:

1. **Graph Convolutional Attention Layer**. It is a convolution, adapted to work on graph structures, that uses the self-attention mechanism to compute how every node attends to each neighbour.
2. **ReLu** activation function. The activation functions decide if a neuron in the neural network needs to be activated or not. They introduce non-linearity for the learning of more complex tasks. The Rectified Linear Unit (ReLu) is expressed as $f(x) = max(0, x)$.
3. **Dropout layer**. It randomly drops out neurons during training allowing for generalization. It is used to help the prevention of overfitting, which is what happens when a model adapts to the observations it was trained on, and it is so unable to perform correctly on unseen data.
4. second **Graph Convolutional Attention Layer**

The reason for avoiding having too many convolutional layers is the risk of oversmoothing. If the message passing and aggregation are done too many times, every node will end up with the same information making it impossible to distinguish and classify them.

In the first approach, the model executes a node-classification task. It's a binary classification problem where the classes are not first-to-be-picked (0) and first-to-be-picked (1). The output of the model is a vector of probabilities as long as the number of nodes. Each probability represents how likeable is for the strawberry (node) to be picked first.
The last layer of this model is a **Sigmoid** activation function $f(x) = \frac{1}{1+e^{-x}}$, which allows all the output values to be between 0 and 1. This is an important constraint since the output values indicate a probability.
The loss, which is a function used to evaluate how similar are the predictions of the model with respect to the ground truth, is the Binary Cross Entropy(BCE) loss:

$$BCE = \frac{1}{N_{nodes}} \sum_{i=1}^{N_{nodes}} y_i \log(p(y_i)) + (1 - y_i) \log(1 - p(y_i)) \tag{4.1}$$

where N_{nodes} is the number of analyzed nodes in the batch, y_i is the label 0 or 1, and $p(y_i)$ is the probability of a node being 1. As a matter of fact, BCE is the standard loss function for binary classification problems. It computes the cross-entropy loss between predicted labels and true labels, where the cross-entropy is a measure of the difference between two probability distributions. Since the dataset is imbalanced (see Table 5.1), the loss can be weighted.

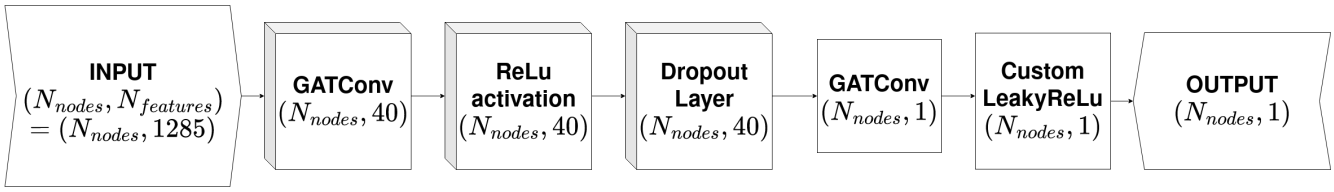


Figure 4.9: Visualization of GAT scheduling score predictor model

In the beginning, the classification targeted was similar to the DETR scheduling attempt (see chapter 4.1). This means the model had to learn how to distinguish between 17 classes, where the classes represented the order of picking. The activation function was just linear and the loss was Cross Entropy. Similar to BCE, the loss is calculated for each class and summed:

$$CE = -\frac{1}{N_{nodes}} \sum_{i=1}^{N_{nodes}} \sum_{c=1}^{N_{class}} y_{i,c} \log(p_{i,c}) \tag{4.2}$$

where N_{nodes} is the number of analyzed nodes in the batch, N_{class} is the number of classes, $y_{i,c}$ is a binary indicator of the correctness of the prediction of observation i to belong to class c, and $p_{i,c}$ is the predicted probability for observation i to belong to class c.
Some trials were also made in the training of a classifier for 5 classes, in which we identified as "fifth to be picked" the strawberries labelled as sixth higher, too. This helped with the imbalance of the dataset; nevertheless, the GAT scheduling model performed better when trained as a binary classifier. It also became more reasonable to concentrate the efforts on the correct prediction of the first strawberry to pick. This is because during the collection of a target, the manipulator might collide with leaves, stems, or other berries, changing the configuration and consequently the best scheduling order.

The other approach consists of the prediction of an "easiness of picking" score for each strawberry. This is a way of generalizing: this score is in fact an absolute measure, not relative to the single images.
The score is not present as an annotation of the dataset. It is retrieved as described in the Heuristic easiness score subsection of this chapter.
Here the last layer of the model is a custom **LeakyReLU**, with a small negative slope α and a positive slope β: $f(x) = \beta max(0, x) + \alpha min(0, x)$. Since the variables α and β are hyperparameters, their values are tuned during the training. There is no need for the model to output values between zero and one, but this activation function helps in the distribution of the scores following the trends noted in the dataset. The score distribution of the dataset can be visualized in Figure 6.3 (a).

The loss function used is Mean Squared Error:

$$MSE = \frac{1}{N_{nodes}} \sum_{n=1}^{N_{nodes}} (y_{pred_n} - y_{true_n})^2 \tag{4.3}$$

where N_{nodes} is the number of analyzed nodes in the batch, y_{pred_n} is the predicted score for node n and y_{true_n} is the true label score of node n. For each node, it measures the squared difference between the predicted and the true score and then computes the mean.

In all the approaches a scheduling order can be retrieved from the model output: the harvesting order is from the most likeable to the least likeable to be picked first, or from the easiest to pick to the least easy to pick.

5 | Experimental Setup

This chapter outlines the collection and preprocessing of the trajectories dataset and the strawberry images dataset. Then it proceeds to describe the testing of the presented work on a Franka Emika manipulator.

5.1. Motion Planning

5.1.1. Trajectories dataset collection

The working environment for the data collection of a dataset of trajectories was comprehensive of:

- a Franka Emika manipulator;
- a RealSense camera;
- Operating system: Ubuntu 16.04;
- ROS Kinetic;
- Python 2.7;
- Libfranka 0.5.0;
- Frankaros 0.6.0.

The trajectories are retrieved for 5 different configurations of strawberry clusters, each one containing 20 strawberries. For every strawberry in every configuration, a picture was taken. In Figure 5.1 you can see an example of one of the configurations.

To collect the trajectories the robot was hand-guided from a fixed home to the reach-to-pick position for every strawberry. For this operation, torque control was applied, meaning the output of the controller is the torque, and there is feedback from the applied force. After reaching a strawberry, the robot is sent back to the home position. Here, the control is switched to position control, where the output is the position of the joints. In Figure 5.2 it is illustrated the robot's control interface. The trajectories were collected at

Figure 5.1: Shot taken from the camera of the robot in home position

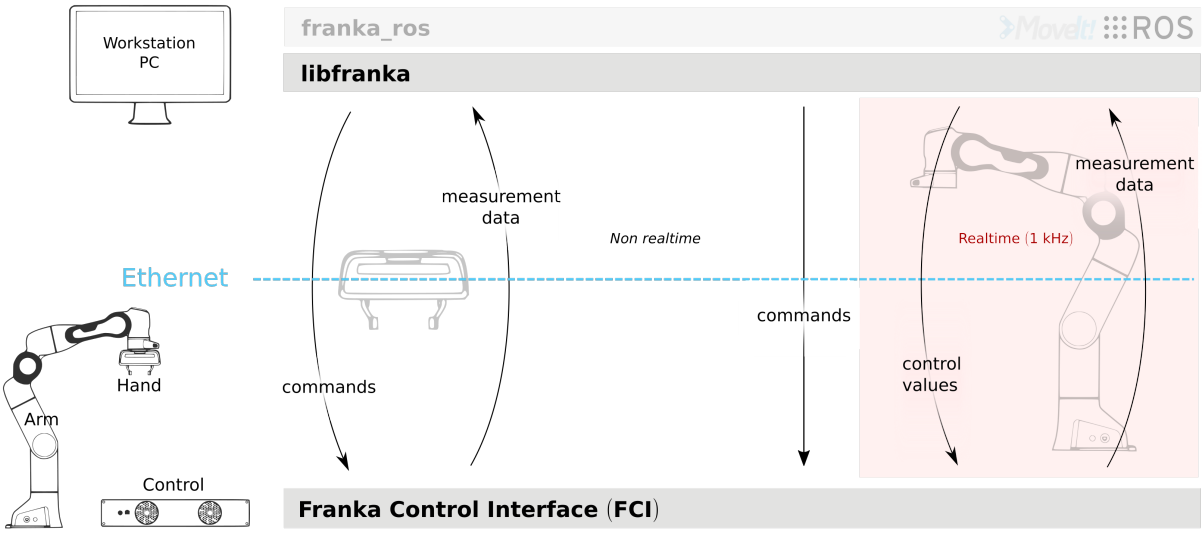


Figure 5.2: Franka Control Interface (FCI)

the velocity of around 30 frames per second, following a sequence of strawberries from left to right. If two or more strawberries are on the same branch, then the order is from top to bottom. For each strawberry 10 trajectories are collected. This adds up to a total of 1000 collected trajectories. Moving the robot, a wide range of solutions to reach the strawberry were covered, in order to obtain a wide probabilistic distribution of trajectories and mimic human behaviour. All the trajectories were saved both for joint states and task space, respectively as a tri- or bi-dimensional array of shape $(t, 1, 7)$ and $(t, 7)$; t is the time taken for each experiment. Task space is the space where the task that the manipulator has

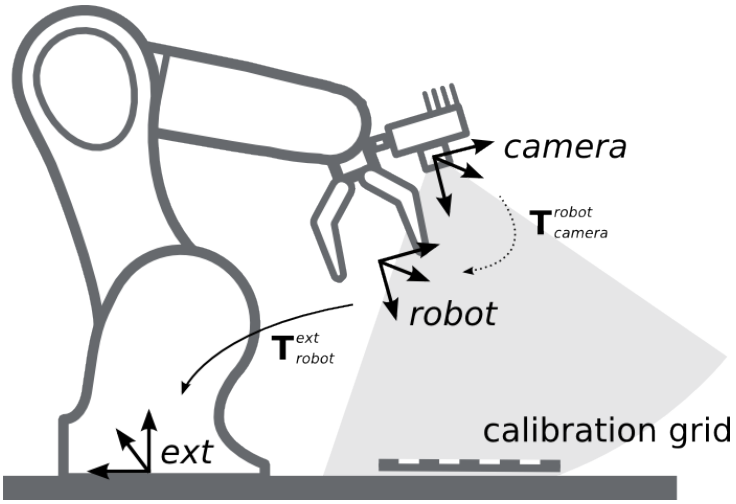
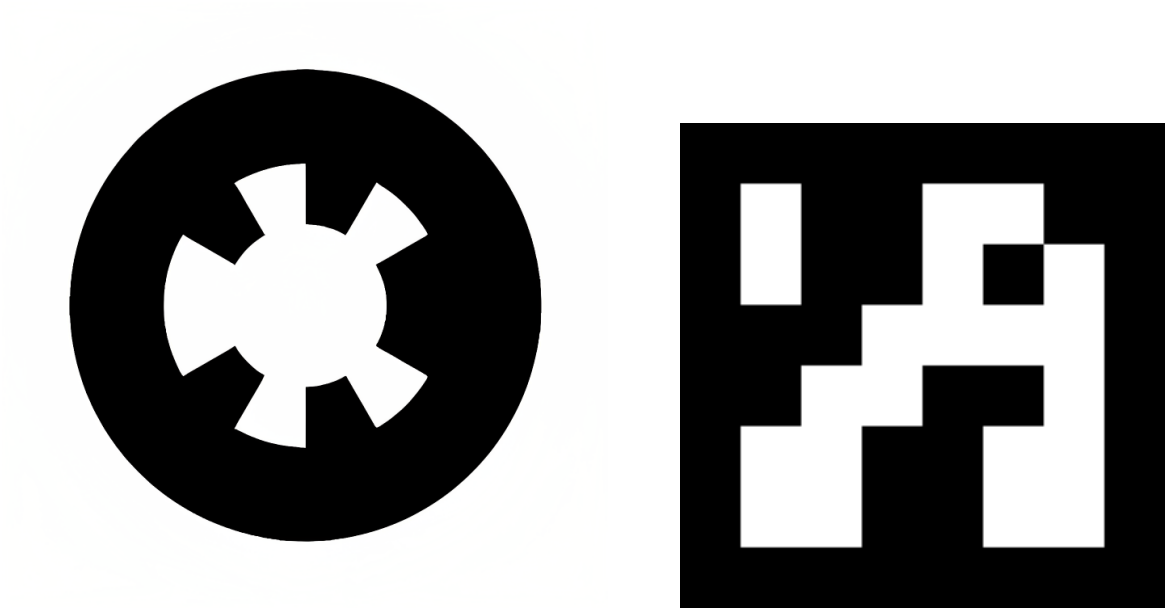


Figure 5.3: Visualusation of the robot and camera frame. T_{camera}^{robot} is the output of the calibration. Image is taken from [84].

to accomplish is specified, and it's defined by the posture $x = \begin{bmatrix} p \\ \phi \end{bmatrix}$, with p position and ϕ orientation. Joint space is defined by the vector of joint variables $q = \begin{bmatrix} q_1 \\ \vdots \\ q_7 \end{bmatrix}$, where $q_i = \theta_i$ for rotating joints, and $q_i = d_i$ for prismatic joints.

Mounting a camera on top of the robot's end effector adds weight; this makes the robot's arm not able to sustain itself while in torque control. So the gripper mass was tuned to include the camera. From the camera sensor, it can be retrieved where an object is in the camera frame. To retrieve this information in the robot frame, camera calibration is needed.

For this operation, WhyCon markers [81] were used (Figure 5.4 (a)): they are registered markers with known dimensions, and they can be easily detected by the camera. To estimate the new frame transformation, the robot is asked to move the end effector in different positions given in the robot frame and to save each time the perceived position of the markers in the camera frame. The output of the calibration is a translation and a rotation as numbers in meters to characterize the frame. Now there's a static transformation between the camera frame and the robot frame. To test the correctness of the calibration, ArUco markers [82, 83] were used (Figure 5.4 (b)). The robot was asked to reach the marker, which position was now detected in the robot frame. If the robot is able to reach the target, it means that the calibration is correct.



(a) (b)

Figure 5.4: Example of (a) WhyCon marker [81] and (b) Aruco marker [82, 83]

5.1.2. Data processing

The goal is to get compressed information on the trajectory distributions' mean and covariance.
The ProMP's weights are defined as:

$$weight_{1\times b} = [(\Phi^T_{b\times t}\Phi_{t\times b} + 1^{-12}I_{b\times b})\Phi^T_{b\times t}T_{t\times 1}]^T \tag{5.1}$$

where b is the number of basis functions chosen to be 8; t is the time of the duration of the trajectory; Φ is the Gaussian basis function; I is an identity matrix; T is the trajectory for a single degree of freedom.
The weights for all ten trajectories taken to reach the same strawberries are computed and stacked in a $(n \times b)$ matrix, where n is the number of trajectories. The mean and covariance can now be computed. Covariance indicates the level to which two variables vary together. The mean vector consists of the mean of each variable, and it has shape $(1 \times b)$, while the covariance matrix C has shape $(n \times b)$.

PCA is a statistical procedure that allows for the reduction of the dimensionality of a data object, without losing important information. To apply this procedure to the trajectories' covariance, Singular Value Decomposition (SVD) needs to be computed.
SVD is a linear algebra operation that computes the factorization of a matrix $C_{n\times b}$ in three matrices $U_{n\times n}$, $S_{n\times b}$, and $V_{b\times b}$. Since C is real, U and V are real orthogonal matrices. S

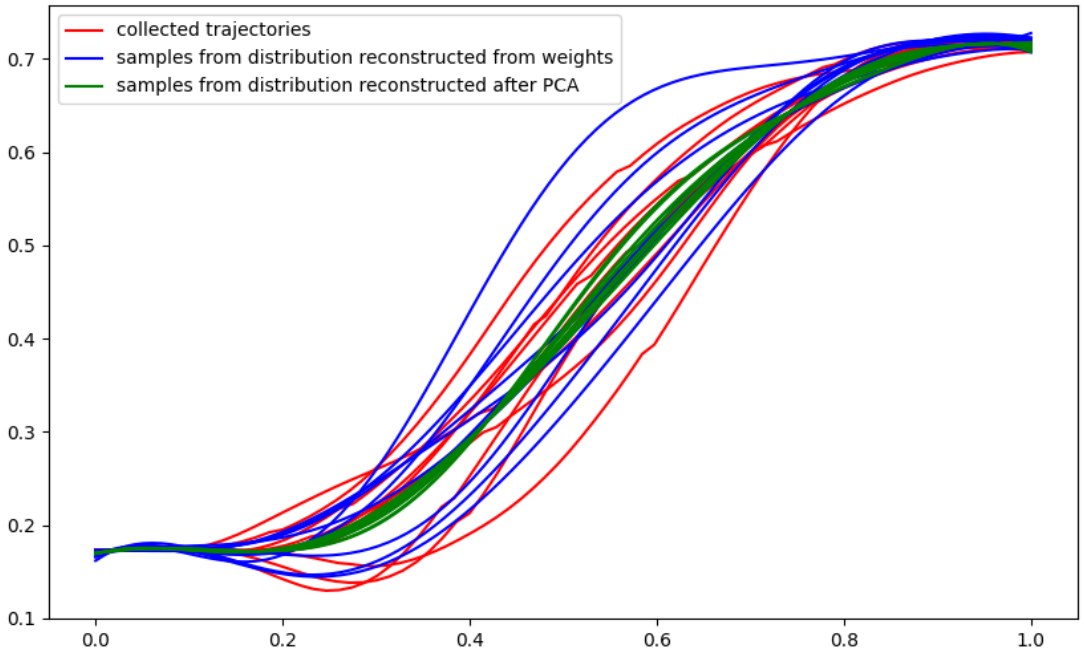


Figure 5.5: Trajectories comparison

is a diagonal matrix that contains in decreasing order the singular values, meaning C^TC's (or CC^T) eigenvalues' square root. U's column are the eigenvectors of CC^T, V's columns are the eigenvectors of C^TC. The eigenvalues represent the dimensionality of the data. The greatest the eigenvalue, the more relevant its corresponding data. For example, in the first collected trajectory of the first joint, the singular values contained in S are:

$$s = [2.49\text{x}10^{4}, 5.31\text{x}10^{2}, 3.48\text{x}10^{0}, 9.23\text{x}10^{-1}, 3.09\text{x}10^{-3}, 6.08\text{x}10^{-4}, 2.49\text{x}10^{-5}, 1.05\text{x}10^{-8}]$$

This means that the first singular value contains more than 95% of the information of C, so the others can be discarded. This was verified for all the trajectories, in both task and joint space. By stacking together the first element of S $s1$, the first row of U $u1$, and the first row of V^T $v1$, a vector of dimension $1 + n + b = 19$ is obtained, from which it can be reconstructed a covariance of $8 \times 8 = 64$ elements as $C = u1 * s1 * v1^T$. This means that with this dimensionality reduction only $7 \times b = 56$ values are needed for the mean and $7 \times (1 + n + b) = 133$ for the covariance of a trajectory distribution, with a total of 189 elements for all seven DoFs. In Figure 5.5 there are plotted ten trajectories taken from a distribution that has the original mean vector and covariance C, confronted with the ten trajectories from which weights were computed. As can be seen, PCA removed some noise and some redundancy in the distribution.

(a) (b)

Figure 5.6: Figure (a): data sample. Figure (b): annotated image via [85, 86]; the numbers on the bounding boxes indicate the chosen scheduling label.

5.2. Scheduling decision

5.2.1. Dataset description

The dataset was collected by MSc students from the University of Lincoln. It consists of 2000 images of clusters of strawberries. Figure 5.6 (a) shows a sample of the data. For each "raw" image they generated a corresponding annotated image (Figure 5.6 (b)), and a JSON file containing the annotations about the ripe fruits. The structure of the annotation is composed by name and size (width×height) of the image file, and this information for each bounding box:

- shape attributes:
  - x, y pixel coordinate of the x, y position of left down corner of the bounding box;
  - *width*, *height* of the bounding box.
- region attributes:
  - *scheduling*: label chosen by taking into consideration the position of the strawberry in the cluster and the occlusion properties;
  - *occlusion*: property selected between "occluded", "occluding", "occluded/occluding" and "neither".

The JSON annotations were uniformed in a Common Object in Context (COCO) format [87]. COCO is one of the most popular large-scale labelled image datasets for public use. It stores data in a JSON file formatted by info, licenses, categories, images, and annotations.

5.2.2. Data as Graphs

Data needs to be represented as graphs, where the idea is to use strawberries as the nodes that exchange information about their representations. The input graphs of Graph Neural Networks are formed by nodes and edges, characterized by their respective features.

- Node features:
  - bounding box coordinates (normalized, to make information simpler to use);
  - percentage of occlusion, considering the overlapping of the bounding boxes;
  - leaf occlusion, binary indicated (0 if the berry is not occluded by a leaf, 1 otherwise);
  - ripeness, binary indicated (0 if ripe, 1 if unripe);
  - patch of the image inside the bounding box, compressed with EfficientNet [88] - a competitive CNN classifier that takes images as input; to retrieve just the compressed information of the inputted image, the last layers are not used, since those are for the classification task.
- Edge feature:
  - pixel-wise euclidean distance between the bounding boxes (normalized, to make information simpler to use)

The adding or removing of different node features was tuned during training.

There are no graphs with less than two nodes: images with just one strawberry were dismissed, as they represent a trivial case in scheduling prediction. Table 5.1 shows the number of strawberries present in the dataset for each scheduling label. There are more "first-to-be-picked" berries than "eleventh-to-be-picked". This is of course because there are graphs with different numbers of nodes.
It is important to avoid over-smoothing: this is something that can happen when information is shared and aggregated too many times between nodes so that they all converge the same information. The number of edges was changed multiple times to see what configuration was giving the best performance: fully connected graphs, with strawberries connected in a line, in a polygon, or with just the diagonals of this polygon.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1321	1321	785	512	302	186	106	67	37	28	12	6	6	4	3	3	2

Table 5.1: Dataset distribution.
Number of strawberries for each scheduling label

(a) (b) (c)

Figure 5.7: The robot snaps a picture from home position (a). This is fed to the algorithm to retrieve a target berry as the one in (b). The target is given to the model for the trajectory generation, so the robot can execute it (c)

5.3. Robot test of the models' integration

This section illustrates how to integrate the perception, scheduling, and trajectory generation models for autonomous harvesting with an RGB image input. A representation of this is in Figure 5.8.
The raw image taken by the robot in its home position is input to two object detection models, to retrieve information on the bounding boxes of ripe and unripe strawberries and the occlusion properties of the ripe ones. This information is then represented in a graph format. The graph is fed to the scheduling decisions model, to get the order of picking. This data is then given to trajectory generation Improved-d-PMP model developed in the parallel work introduced in Chapter 3.1. Now, the robot can execute the reach-to-pick motion. The user can choose between a Pick-All or Pick-One algorithm, deciding if the robot will execute the picking of all the ripe strawberries in the shooted image or if the order of picking needs to be recomputed after the collection of each strawberry.

The setup is the same as for the trajectory dataset collection. The robot is asked to take an image. This is fed to the algorithm above explained, which outputs the trajectory that the robot will execute in position control. After reaching the goal position, it returns to the home configuration. Figure 5.7 shows an example of performing.

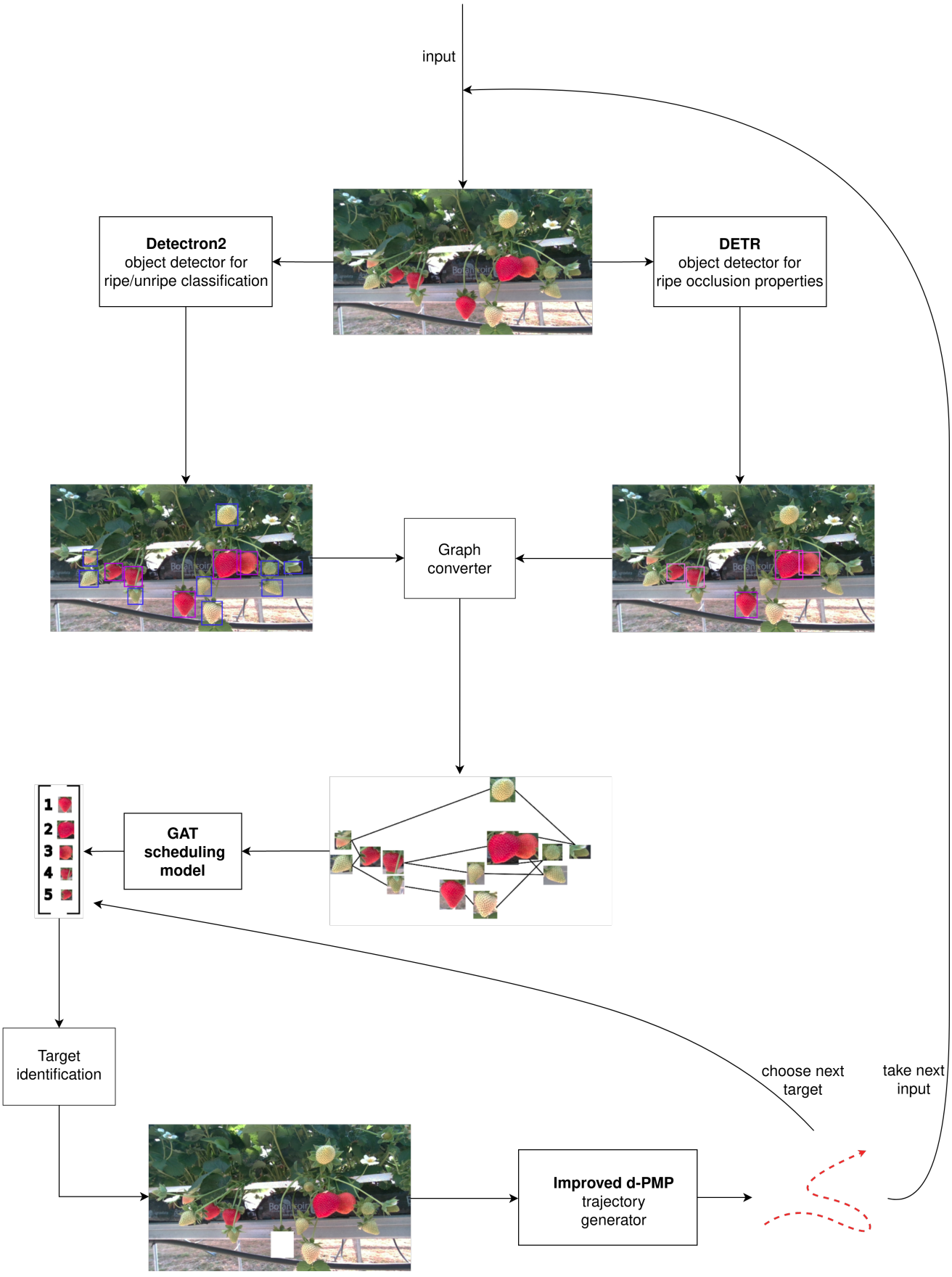


Figure 5.8: Logical scheme for a real-life application

6 | Results and Discussion

This Chapter discusses the results of the models used in this thesis for the scheduling decision problem, both for the detection and the harvesting order prediction parts.

6.1. Object Detection

As described in Chapter 4.1, this work provides a DETR model for the visual detection of the strawberries and classification of their occlusion properties. The gain of using such a model with a self-attention mechanism will be underlined even more in the next chapter, but Table 6.1 shows its cons: without a huge amount of data and computational power state-of-the-art performances can never be achieved.

DETR-DC5 model	GPUs	data	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Benchmark	8	123k	43.3	63.1	45.9	22.5	47.3	61.1
Customized	1	2k	21.2	29.8	24.1	30.0	35.0	46.5

Table 6.1: DETR comparison. Benchmark data are taken from [53].
AP is the abbreviation of "average precision". The subscript number is the threshold of bounding box overlap. S, M and L stand respectively for small, medium and large.

The trends between the average precision values are similar to the state-of-the-art. For example, the average precision values are higher for the bigger identified object. This is a known issue with DETR, already addressed in the original paper [53]. This matter is here somewhat discouraged. Since the given dataset has all the annotated bounding boxes "small", meaning smaller than a quarter of the image size, the model is trained more to detect little targets. Consequently, the AP_S value of the customized model outperforms the benchmark one.
For the specific application of this thesis research, these limited results were not a restriction since the training of the GAT scheduling model was done with the information in the annotations and not with the outputs of the perception models as it happens in the test exposed in the previous chapter. Moreover, many other models for a perception task

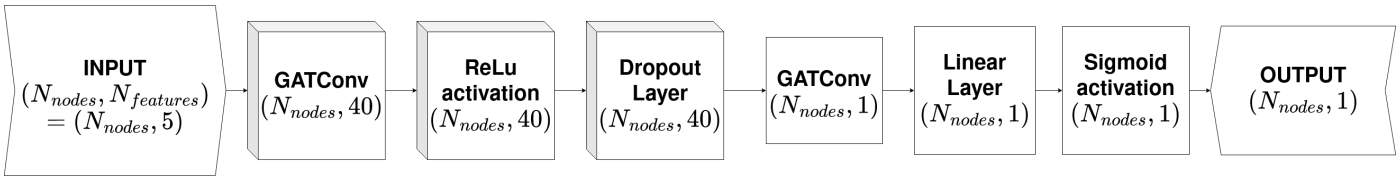


Figure 6.1: Visualization of simpler GAT scheduling classification model

can be adapted to the provided strawberry dataset without the need for more data or GPUs. For example, the Detctron2 model used to add information on unripe berries is easily adaptable to detect occlusion properties as well.

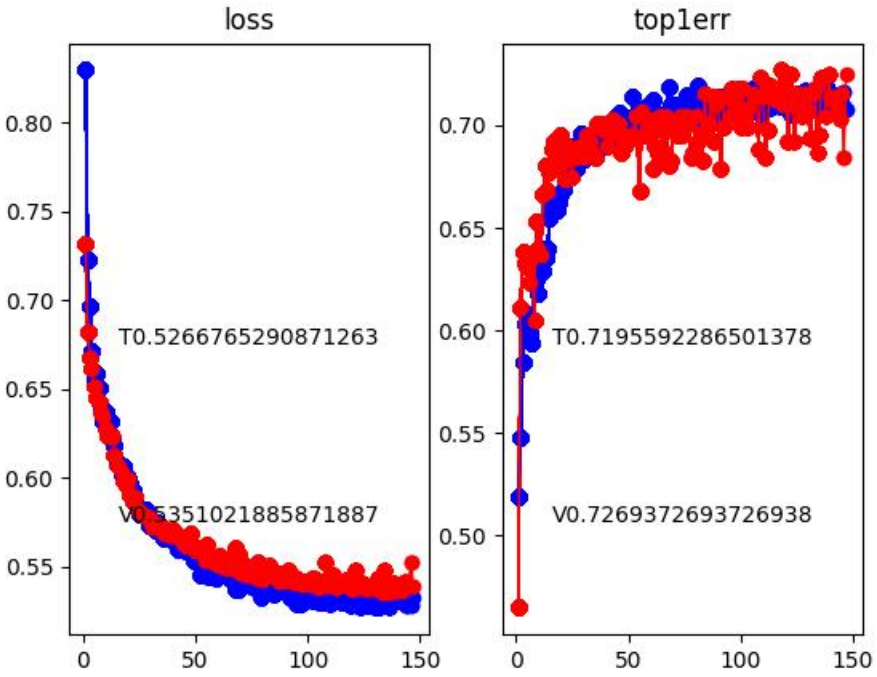
6.2. Graph Attention Network

The GAT scheduling models designed in the beginning were very simple, and more enriching features were added building upon those. The first models were trained on data graphs without information on unripe berries or image patches, meaning every node carried only five features 6.1. The most important hypothesis was that the first two berries of the annotated scheduling order were both labelled as first-to-be-picked, to have a more balanced dataset (see Table 6.2) which is a very helpful feature in a classification problem. The justification behind this choice is that the aim of the research was training a network to identify the easiest target to pick with human-like reasoning.
The simplicity of the model allowed faster training, giving the possibility of confronting the performance of different configurations by varying the number of layers and neurons and by tuning hyperparameters.

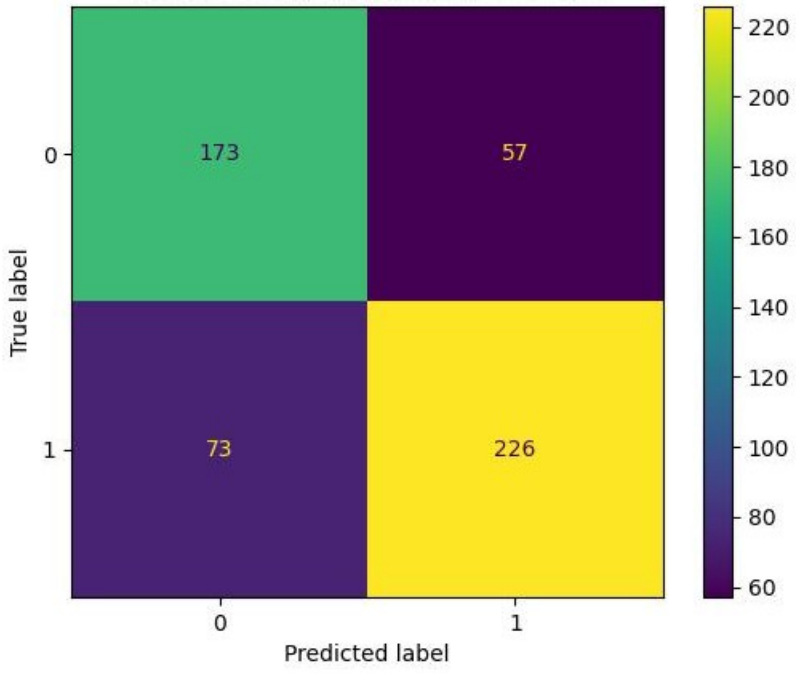
Table 6.2: Dataset distribution after balancing

First to be picked (1)	2642
Not first to be picked (0)	2059

Figure 6.2 (a) shows the variation of BCE loss and accuracy of prediction during training of the best simpler model. The values printed on the plot represent the loss and accuracy on the train and validation sets of the model with the smallest validation loss.
The confusion matrix of Figure 6.2 (b) reports a precision of $\frac{TP}{TP+FP} = 0.6976$ and recall of $\frac{TP}{TP+FN} = 0.7457$. So this model has a good enough accuracy for recognising the first strawberry to pick in a cluster, but the loss and the prediction of the whole scheduling order can be improved.

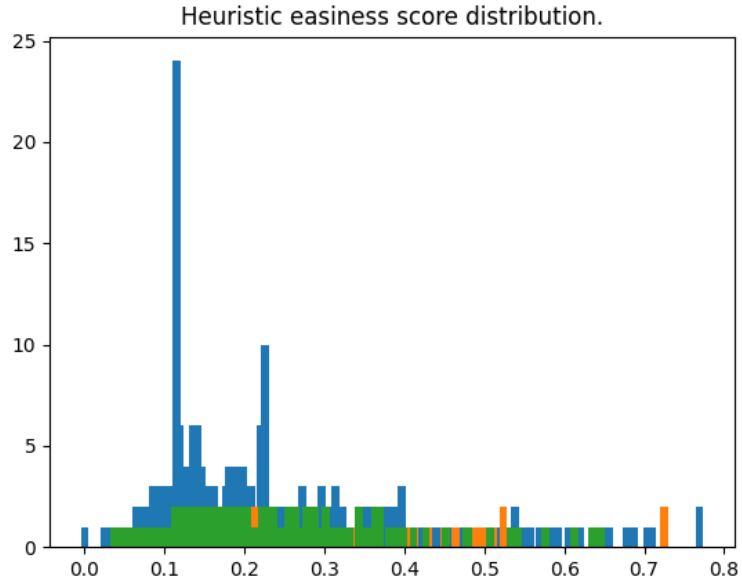


(a)

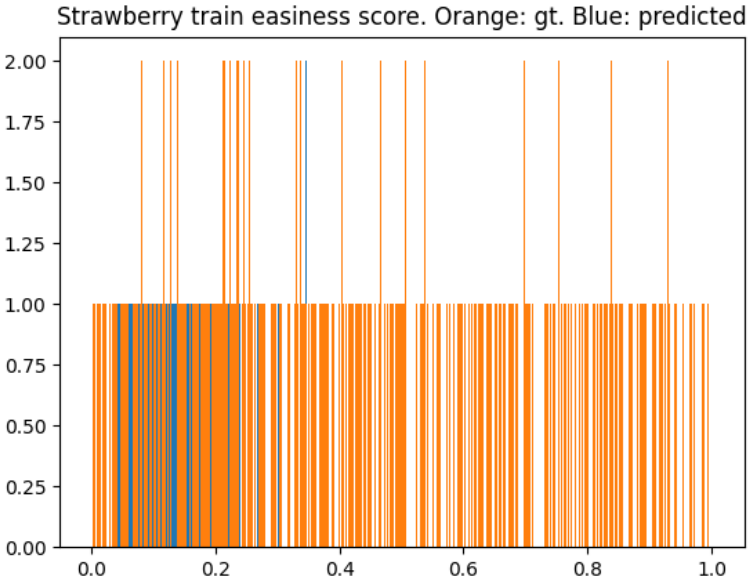


(b)

Figure 6.2: Simpler GAT scheduling model loss and accuracy training plot (a), and confusion matrix on the test set (b).



(a)



(b)

Figure 6.3: This data bar shows the number of strawberries (y-axis) for each rounded easiness score (x-axis). Figure (a): in blue is represented the train, in orange the validation, and in green the test set. Figure (b): in blue are represented the predictions, and in orange the train set.

The models for the prediction of an absolute easiness score for each fruit had some complications. Figure 6.3 (a) shows the distribution of the scores in the dataset. The predictions of the model in Figure 6.3 (b) are all values close to zero. An explanation behind this could be that the MSE loss favours small values. To discourage this, a custom LeakyReLu activation function was introduced (see chapter 4.3). Another cause is the imbalance in the dataset, which led to reasoning on a new easiness score computation with less weight on the pixel-wise Euclidean distance between the strawberries.

The removal of the hypothesis for the forced balancing and the introduction of the unripe strawberries in the images did give more knowledge of the whole cluster to the model, but it also brought more imbalance in the dataset. In fact, unripe berries are now nodes in the graphs with a label, and they will be classified by the model. Improving the binary node classification model with these new features, the accuracy of the prediction of the first berry to be picked diminished, but the overall scheduling decision is more accurate, as shown in Figure 6.4. Again, every element of the cell $c_{i,j}$ indicates the number of strawberries with human-decided scheduling label i and predicted scheduling label j; if the two schedulings coincided, the heatmap would be a diagonal matrix. Here the percentage of correspondence is 38.7%.
Figure 6.5 displays the occlusion properties of the scheduled strawberries. Each cell $c_{i,j}$ indicates the number of berries with scheduling label j which have occlusion property i, chosen between "non-occluded", "occluded by a leaf", and "occluded by a berry". This shows that the majority of non-occluded strawberries are usually picked first by the model, as expected. A surprisingly high amount of strawberries occluded by a berry are picked early in the scheduling. This suggests that a more aggressive weight should be given as occlusion in the node features.
Figures 6.6, 6.7 are heatmaps to compare the predicted scheduling trained on human annotation with the heuristic scheduling computations. These numbers indicate how the overall scheduling sequence is less accurate, with a percentage of overall correspondence of roughly 7%, but there's more accordance in the prediction of the easiest strawberries to collect. It is a promising result since during the testing of the in-development work it was noted that after the picking of a single berry, the configuration of the cluster might change because the manipulator might collide with leaves or other fruits. Although being able to output directly the whole sequence of order of picking is an interesting challenge, it could be fairly important to evaluate just which is the easiest element to be picked.

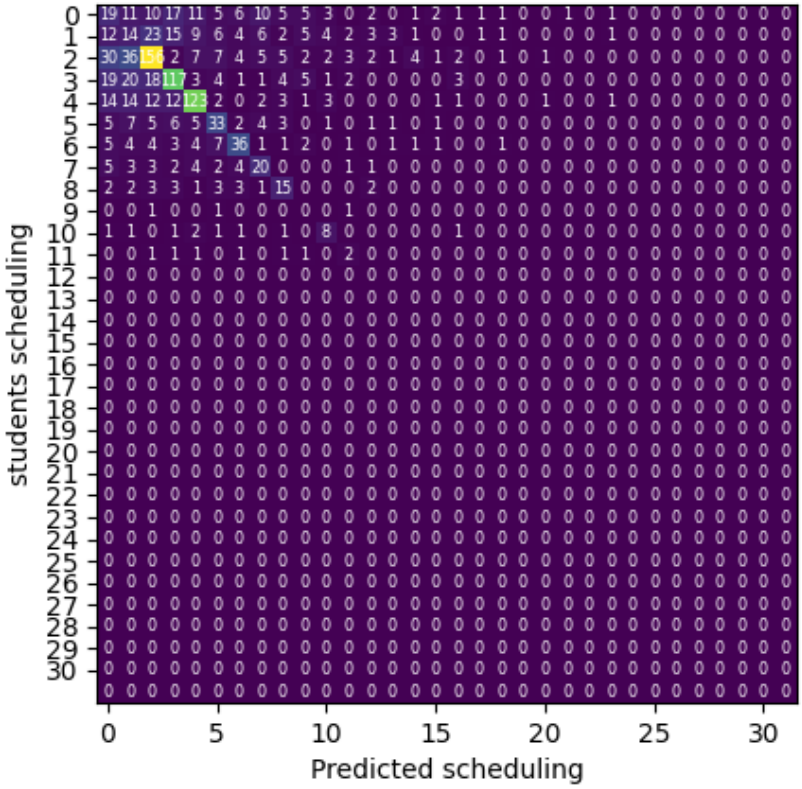


Figure 6.4: Comparison between students scheduling choice and GAT scheduling model prediction

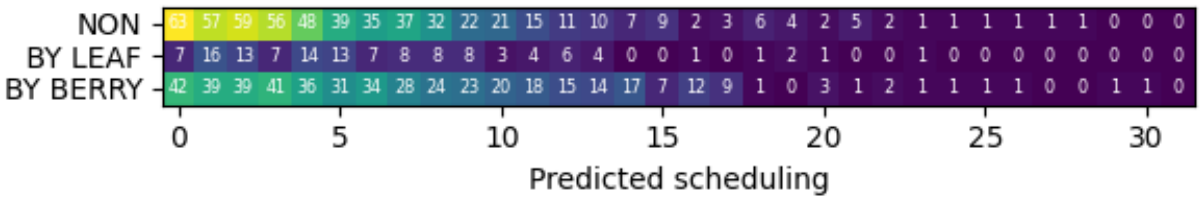


Figure 6.5: Occlusion properties of the scheduled strawberries

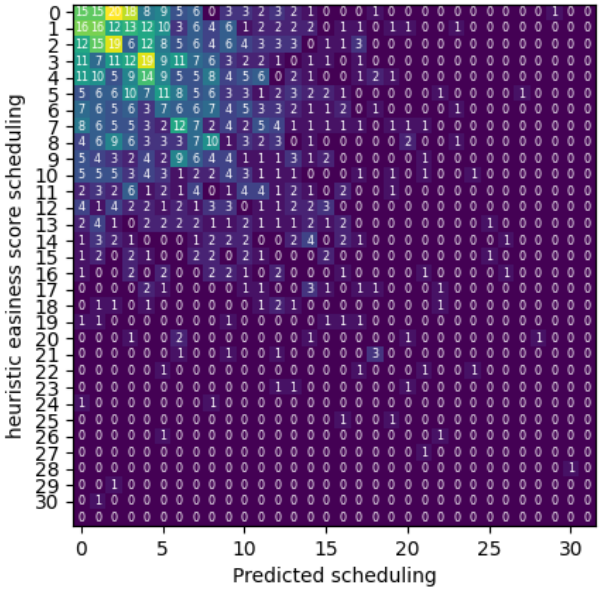


Figure 6.6: Comparison between scheduling from heuristic easiness score and GAT scheduling model prediction

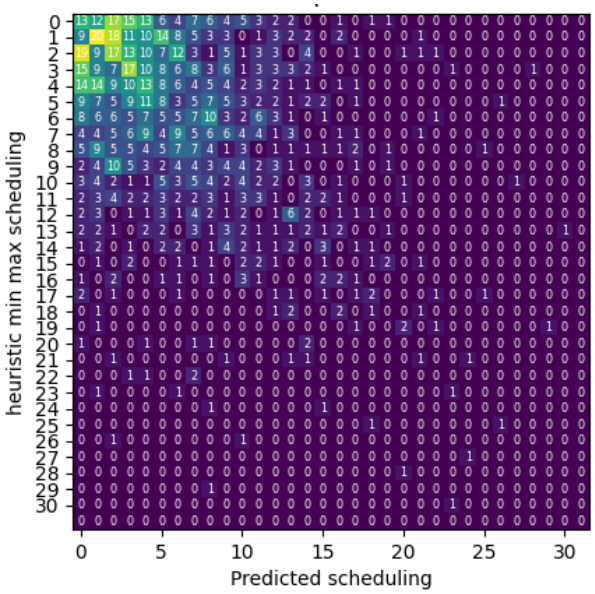


Figure 6.7: Comparison between heuristic min-max scheduling and GAT scheduling model prediction

7 | Conclusions and future developments

This thesis work proposes solutions to some of the main problems of autonomous robotic strawberry harvesting.
In particular, the reach-to-pick task from visual input was improved starting from the work of d-PMP [30]. The progress sets the basis for the generation of more precise trajectories with joint correlation. Since the labour shortage is a growing problem [7] and labour represents the largest cost and also a large operational uncertainty for strawberry farmers [18], this work suggests another way how automation can help in the agricultural field.
To reduce failures in autonomous picking, the choice of the target berry in a cluster is crucial. The scheduling decision prediction from perception is a novelty in the harvesting of small fruit. GAT scheduling model is presented in two main versions, to accurately predict the first strawberry to be picked or the whole harvesting sequence.
The overall procedure of target identification and trajectory generation from visual input was tested on a Franka Emika manipulator provided with a RealSense camera.

Since picking scheduling prediction with GNNs is a new topic, many future improvements can grow from this work. Starting from the perception problem, a new dataset would be needed to train a model that recognises both the ripeness and the occlusion of the strawberries.
Another improvement could surely be to give graphs depth perception. Analyzing the results of the easiness score, it turns out that pixel distance and bounding box size are not enough to capture the separation between berries. As already introduced in the previous chapter, a new scheduling score could benefit from this information. Moreover, a custom loss could be introduced for the GAT scheduling model that assigns an absolute easiness score to each strawberry, since the MSE method favours small values and is slowing down the learning of correct prediction.
An interesting improvement in the graph representation would be exploiting more DETR for the node features. In fact, the decoder part of the transformer outputs an attention map for every detected object, that measures how pixels attend to each other. A

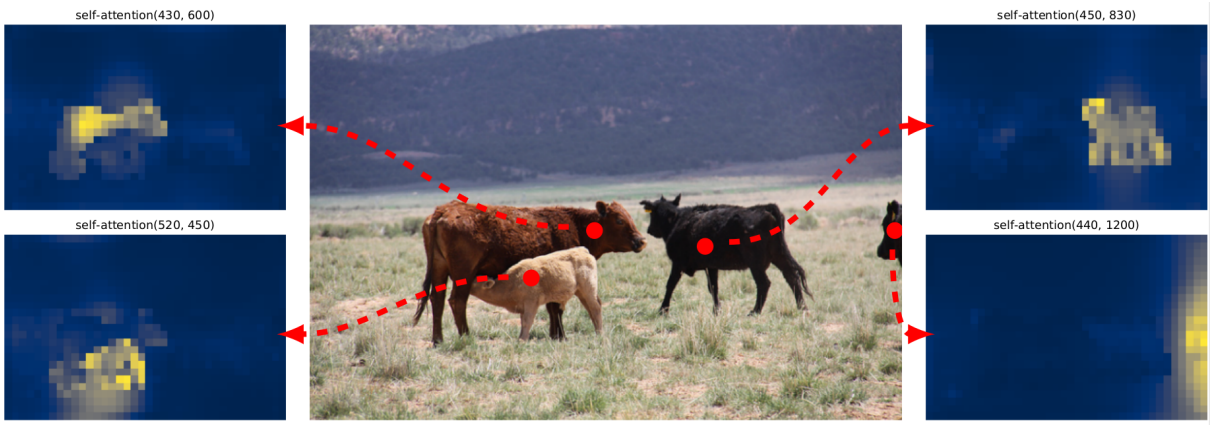


Figure 7.1: From "End-to-End Object Detection with Transformers" [53]: encoder self-attention for a set of reference points. The encoder is able to separate individual instances. Predictions are made with baseline DETR model on a validation set image.

representation of this is shown in Figure 7.1. Including this information in the graphs would be another measure of how strawberries are related to one another in a cluster. It could be tested if Attention is enough in this case or if other parameters such as occlusion properties and distance are necessary to understand a scheduling order.

Bibliography

[1] Tom Duckett, Simon Pearson, Simon Blackmore, Bruce Grieve, Wen-Hua Chen, Grzegorz Cielniak, Jason Cleaversmith, Jian Dai, Steve Davis, Charles Fox, et al. Agricultural robotics: the future of robotic agriculture. *arXiv preprint arXiv:1806.06762*, 2018.

[2] Linly Ku. How automation is transforming the farming industry. `https://www.plugandplaytechcenter.com/resources/how-automation-transforming-farming-industry/`, 2022. Accessed: 2023-03-09.

[3] Minli Yang. Agricultural robotics for climate-resilient food production. `https://aiforgood.itu.int/event/agricultural-robotics-for-climate-resilient-food-production/`, 2021. Accessed: 2023-03-09.

[4] Laure Ducos. Agricultural robotics for climate-resilient food production. `https://www.fnh.org/wp-content/uploads/2022/06/Agriculture-food-war-report.pdf`, 2022. Accessed: 2023-03-09.

[5] Coronavirus triggers acute farm labour shortages in europe. `https://www.spglobal.com/commodityinsights/en/ci/research-analysis/article-coronavirus-triggers-acute-farm-labour-shortages-europe.html`, 2020. Accessed: 2023-03-09.

[6] Sam Fleming. Eu must prepare for 'era of pandemics', von der leyen says. `https://www.ft.com/content/fba558ff-94a5-4c6c-b848-c8fd91b13c16`, 2021. Accessed: 2023-03-09.

[7] Agricultural robots market size, share, trends, growth 2023-2028. `https://www.expertmarketresearch.com/reports/agricultural-robots-market`, 2022. Accessed: 2023-03-09.

[8] Max Roser. Employment in agriculture. *Our World in Data*, 2013. https://ourworldindata.org/employment-in-agriculture. Accessed: 2023-03-09.

[9] Robotnik. Robotics applications in agriculture. https://robotnik.eu/robotics-applications-in-agriculture/, 2022. Accessed: 2023-03-14.

[10] Robotic Industries Association. Robots' benefits in the agricultural industry. https://www.controleng.com/articles/robots-benefits-in-the-agricultural-industry/, 2017. Accessed: 2023-03-14.

[11] Sigma Consulting. Cos'è l'agricoltura 4.0. https://www.sigmaconsulting.it/it/prodotti-servizi/agricoltura-4-0/, 2023. Accessed: 2023-03-14.

[12] Louis Columbus. 10 ways ai has the potential to improve agriculture in 2021. https://www.forbes.com/sites/louiscolumbus/2021/02/17/10-ways-ai-has-the-potential-to-improve-agriculture-in-2021/, 2021. Accessed: 2023-03-14.

[13] Facts Factors. Market size of ai in agriculture is projected to reach usd 2,400 million by 2026, according to facts factors. https://www.globenewswire.com/en/news-release/2021/02/02/2168016/0/en/Market-Size-of-AI-in-Agriculture-is-Projected-to-Reach-USD-2-400-Million-by-2026-According-to-Facts-Factors.html, 2021. Accessed: 2023-03-14.

[14] Jason Dorfman. Fields of automation. https://www.economist.com/technology-quarterly/2009/12/12/fields-of-automation, 2009. Accessed: 2023-03-14.

[15] Jeff Daniels. From strawberries to apples, a wave of agriculture robotics may ease the farm labor crunch. https://www.cnbc.com/2018/03/08/wave-of-agriculture-robotics-holds-potential-to-ease-farm-labor-crunch.html, 2018. Accessed: 2023-03-14.

[16] Mateus Cruz, Samuel Mafra, Eduardo Teixeira, and Felipe Figueiredo. Smart strawberry farming using edge computing and iot. *Sensors*, 22(15):5866, 2022.

[17] Libelium. Smart strawberries crop increases the quality and reduces the time from farm to market. https://www.libelium.com/libeliumworld/success-stories/smart-strawberries-crop-increases-the-quality-and-reduces-the-time-from-farm-to-market/, 2016. Accessed: 2023-03-14.

[18] Satoshi Yamamoto, Shigehiko Hayashi, Hirotaka Yoshida, and Ken Kobayashi. Development of a stationary robotic strawberry harvester with a picking mechanism that approaches the target fruit from below. *Japan Agricultural Research Quarterly: JARQ*, 48(3):261–269, 2014.

[19] Abhisesh Silwal, Joseph R Davidson, Manoj Karkee, Changki Mo, Qin Zhang, and

Karen Lewis. Design, integration, and field evaluation of a robotic apple harvester. *Journal of Field Robotics*, 34(6):1140–1159, 2017.

[20] Dyson. Strawberry production. `https://dysonfarming.com/strawberries/`, 2023. Accessed: 2023-03-14.

[21] Soran Parsa, Bappaditya Debnath, Muhammad Arshad Khan, et al. Autonomous strawberry picking robotic system (robofruit). *arXiv preprint arXiv:2301.03947*, 2023.

[22] Ya Xiong, Cheng Peng, Lars Grimstad, Pål Johan From, and Volkan Isler. Development and field evaluation of a strawberry harvesting robot with a cable-driven gripper. *Computers and electronics in agriculture*, 157:392–402, 2019.

[23] Ya Xiong, Yuanyue Ge, Lars Grimstad, and Pål J From. An autonomous strawberry-harvesting robot: Design, development, integration, and field evaluation. *Journal of Field Robotics*, 37(2):202–224, 2020.

[24] Alessandra Tafuro, Adeayo Adewumi, Soran Parsa, Ghalamzan E Amir, and Bappaditya Debnath. Strawberry picking point localization ripeness and weight estimation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2295–2302. IEEE, 2022.

[25] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 44(1.2):206–226, 2000.

[26] Erik A Billing and Thomas Hellström. A formalism for learning from demonstration. *Paladyn, Journal of Behavioral Robotics*, 1(1):1–13, 2010.

[27] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. *Advances in neural information processing systems*, 26, 2013.

[28] Aleš Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, 26(5):800–815, 2010.

[29] Oluwatoyin Sanni, Giorgio Bonvicini, Muhammad Arshad Khan, Pablo C López-Custodio, Kiyanoush Nazari, et al. Deep movement primitives: toward breast cancer examination robot. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 12126–12134, 2022.

[30] Alessandra Tafuro, Bappaditya Debnath, Andrea M Zanchettin, and E Amir Ghala-

mzan. dpmp-deep probabilistic motion planning: A use case in strawberry picking robot. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8675–8681. IEEE, 2022.

[31] Gaudenz Boesch. Object detection in 2023: The definitive guide. `https://viso.ai/deep-learning/object-detection/`, 2023. Accessed: 2023-03-17.

[32] Chiagoziem C Ukwuoma, Qin Zhiguang, Md Belal Bin Heyat, Liaqat Ali, Zahra Almaspoor, and Happy N Monday. Recent advancements in fruit detection and classification using deep learning techniques. *Mathematical Problems in Engineering*, 2022:1–29, 2022.

[33] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.

[34] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part II 9*, pages 428–441. Springer, 2006.

[35] Andreu Moreno, Eduardo César, Andreu Guevara, Joan Sorribes, Tomàs Margalef, and Emilio Luque. Dynamic pipeline mapping (dpm). *Lecture Notes in Computer Science*, 5168:295–304, 2008.

[36] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[38] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[39] Anima Pramanik, Sankar K Pal, Jhareswar Maiti, and Pabitra Mitra. Granulated rcnn and multi-class deep sort for multi-object detection and tracking. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(1):171–181, 2021.

[40] Steven W Chen, Shreyas S Shivakumar, Sandeep Dcunha, Jnaneshwar Das, Edidiong Okon, Chao Qu, Camillo J Taylor, and Vijay Kumar. Counting apples and oranges with deep learning: A data-driven approach. *IEEE Robotics and Automation Letters*, 2(2):781–788, 2017.

[41] Juan Fernando Villacrés and Fernando Auat Cheein. Detection and characterization of cherries: A deep learning usability case study in chile. *Agronomy*, 10(6):835, 2020.

[42] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[43] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[44] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[45] Raymond Kirk, Grzegorz Cielniak, and Michael Mangan. L* a* b* fruits: A rapid and robust outdoor fruit detection system combining bio-inspired features with one-stage deep learning networks. *Sensors*, 20(1):275, 2020.

[46] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022.

[47] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. You only learn one representation: Unified network for multiple tasks. *arXiv preprint arXiv:2105.04206*, 2021.

[48] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 13029–13038, 2021.

[49] Weikuan Jia, Yuyu Tian, Rong Luo, Zhonghua Zhang, Jian Lian, and Yuanjie Zheng. Detection and segmentation of overlapped fruits based on optimized mask r-cnn application in apple harvesting robot. *Computers and Electronics in Agriculture*, 172:105380, 2020.

[50] Anand Koirala, KB Walsh, Zhenglin Wang, and C McCarthy. Deep learning for real-

time fruit detection and orchard fruit load estimation: Benchmarking of 'mangoyolo'. *Precision Agriculture*, 20:1107–1135, 2019.

[51] Pedro Azevedo. Object detection state of the art 2022. `https://medium.com/@pedroazevedo6/object-detection-state-of-the-art-2022-ad750e0f6003`, 2022. Accessed: 2023-03-17.

[52] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. Accessed: 2023-03-09.

[53] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020.

[54] Vikas Kookna. Semantic vs. instance vs. panoptic segmentation. `https://pyimagesearch.com/2022/06/29/semantic-vs-instance-vs-panoptic-segmentation/`, 2022. Accessed: 2023-03-17.

[55] Benjamin Sanchez-Lengeling. A gentle introduction to graph neural networks. `https://distill.pub/2021/gnn-intro/`, 2021. Accessed: 2023-03-19.

[56] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[57] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[58] Stefano Bosisio. Graph neural networks: A learning journey since 2008— part 1. `https://towardsdatascience.com/graph-neural-networks-a-learning-journey-since-2008-part-1-7df897834df9`, 2021. Accessed: 2023-03-18.

[59] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[60] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[61] Stefano Bosisio. Graph neural networks: A learning journey since 2008— deep

walk. https://towardsdatascience.com/graph-neural-networks-a-learning-journey-since-2008-deep-walk-e424e716070a, 2021. Accessed: 2023-03-18.

[62] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[63] Stefano Bosisio. Graph neural networks: A learning journey since 2008— graph convolutional networks. https://towardsdatascience.com/graph-neural-networks-a-learning-journey-since-2008-graph-convolution-network-aadd77e91606, 2021. Accessed: 2023-03-18.

[64] P Sajitha and A Diana Andrushia. Banana fruit disease detection and categorization utilizing graph convolution neural network (gcnn). In *2022 6th International Conference on Devices, Circuits and Systems (ICDCS)*, pages 130–134. IEEE, 2022.

[65] James Atwood and Don Towsley. Diffusion-convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

[66] James Atwood, Siddharth Pal, Don Towsley, and Ananthram Swami. Sparse diffusion-convolutional neural networks. *arXiv preprint arXiv:1710.09813*, 2017.

[67] Stefano Bosisio. Graph neural networks: A learning journey since 2008— diffusion convolutional neural networks. https://towardsdatascience.com/graph-neural-networks-a-learning-journey-since-2008-diffusion-convolutional-neural-networks-329d45471fd9, 2022. Accessed: 2023-03-18.

[68] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[69] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th international conference on web search and data mining*, pages 519–527, 2020.

[70] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*, 2020.

[71] Anoushka Vyas and Sambaran Bandyopadhyay. Dynamic structure learning through graph neural network for forecasting soil moisture in precision agriculture. *arXiv preprint arXiv:2012.03506*, 2020.

[72] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[73] Yu Li, Chao Huang, Lizhong Ding, Zhongxiao Li, Yijie Pan, and Xin Gao. Deep learning in bioinformatics: Introduction, application, and perspective in the big data era. *Methods*, 166:4–21, 2019.

[74] Will Badr. Auto-encoder: What is it? and what is it used for? (part 1). `https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726`, 2019. Accessed: 2023-03-16.

[75] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[76] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[77] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

[78] Jessica Stringham. Kl divergence. `https://jessicastringham.net/2018/12/27/KL-Divergence/`, 2018. Accessed: 2023-03-26.

[79] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[80] Akshay Kumar. An overview of visual servoing for robot manipulators. `https://control.com/technical-articles/an-overview-of-visual-servoing-for-robot-manipulators/`, 2020. Accessed: 2023-03-25.

[81] Matias Nitsche, Tomas Krajnik, Petr Cizek, Marta Mejail, Tom Duckett, et al. Whycon: an efficient, marker-based localization system. 2015.

[82] Francisco J Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and vision Computing*, 76:38–47, 2018.

[83] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Rafael Medina-Carnicer. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern recognition*, 51:481–491, 2016.

[84] Roboception GmbH. Hand-eye calibration. `https://doc.rc-visard.com/v1.6/en/handeye_calibration.html`, 2018. Accessed: 2023-03-26.

[85] A. Dutta, A. Gupta, and A. Zissermann. VGG image annotator (VIA). http://www.robots.ox.ac.uk/ vgg/software/via/, 2016. Version: 2.0.12, Accessed: 26-03-2026.

[86] Abhishek Dutta and Andrew Zisserman. The VIA annotation software for images, audio and video. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, New York, NY, USA, 2019. ACM.

[87] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[88] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

List of Figures

1.1 Number of people employed in agriculture [8] 2
1.2 Dyson strawberries poly-tunnel farm [20] 4
1.3 Franka configuration, in the robotics laboratory of the Univesity of Lincoln 5

2.1 Fully-connected Neural Network 10
2.2 DETR Architecture, from the paper "End-to-end object detection with transformers" [53]: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "no object" class prediction. 14
2.3 Illustration of a graph neural network. (A) A typical example of graph data. (B) The embedding space. In this embedding space, each data point is represented by a vector while the original topological information in (A) is preserved in that vector. (C) The graph neural network for embedding the network in (A). Let's use nodes a and b as examples. The internal properties of each node are considered as the original representations. In each layer, the nodes aggregate information from their neighbours and update the representations with averaging and activation functions. The output of layer 2 is considered as the embedding result in this example. Notice that the parameters within the same layer between different trees are shared so this method can be generalized to the previously unseen graph of the same type. Image and description are taken from Li et. al [73] 17

3.1 Starting from the left: compact weight representation of a trajectory for each basis function; Gaussian basis functions; trajectory of a single DoF . . 20
3.2 Starting from the left: weight distribution of a trajectory for each basis function; Gaussian basis functions; trajectory distribution of a single DoF . 21
3.3 d-PMP architecture from [30] 22

3.4 Visualisation of KLD between two distributions p and q. Graphic inspired by [78] 23

4.1 Eye-in-hand VS Eye-to-hand configuration [80]. 26
4.2 Fine-tuned DETR output: strawberries identification and occlusion properties classification. 27
4.3 Fine-tuned DETR output: strawberries identification and scheduling identification. The integer numbers represent the predicted order of picking, the other number is the probability of the prediction 27
4.4 From the paper: Strawberry picking point localization ripeness and weight estimation [24], where ripe strawberries are labelled as "pluckable", and unripe as "unpluckable" 28
4.5 Distance between bounding boxes for boxes' distance computation 29
4.6 Comparison between students scheduling and heuristic min-max computation over the dataset 30
4.7 Comparison between (a) students scheduling choice and heuristic easiness score computation, and (b) heuristic min-max computation and heuristic easiness score computation over the dataset 31
4.8 Visualization of GAT scheduling classification model 32
4.9 Visualization of GAT scheduling score predictor model 34

5.1 Shot taken from the camera of the robot in home position 38
5.2 Franka Control Interface (FCI) 38
5.3 Visualusation of the robot and camera frame. T_{camera}^{robot} is the output of the calibration. Image is taken from [84]. 39
5.4 Example of (a) WhyCon marker [81] and (b) Aruco marker [82, 83] 40
5.5 Trajectories comparison 41
5.6 Figure (a): data sample. Figure (b): annotated image via [85, 86]; the numbers on the bounding boxes indicate the chosen scheduling label. . . . 42
5.7 The robot snaps a picture from home position (a). This is fed to the algorithm to retrieve a target berry as the one in (b). The target is given to the model for the trajectory generation, so the robot can execute it (c) . 44
5.8 Logical scheme for a real-life application 45

6.1 Visualization of simpler GAT scheduling classification model 48
6.2 Simpler GAT scheduling model loss and accuracy training plot (a), and confusion matrix on the test set (b). 49

6.3 This data bar shows the number of strawberries (y-axis) for each rounded easiness score (x-axis). Figure (a): in blue is represented the train, in orange the validation, and in green the test set. Figure (b): in blue are represented the predictions, and in orange the train set. 50
6.4 Comparison between students scheduling choice and GAT scheduling model prediction 52
6.5 Occlusion properties of the scheduled strawberries 52
6.6 Comparison between scheduling from heuristic easiness score and GAT scheduling model prediction 53
6.7 Comparison between heuristic min-max scheduling and GAT scheduling model prediction 53

7.1 From "End-to-End Object Detection with Transformers" [53]: encoder self-attention for a set of reference points. The encoder is able to separate individual instances. Predictions are made with baseline DETR model on a validation set image. 56

List of Tables

5.1 Dataset distribution. Number of strawberries for each scheduling label . . 44

6.1 DETR comparison. Benchmark data are taken from [53]. AP is the abbreviation of "average precision". The subscript number is the threshold of bounding box overlap. S, M and L stand respectively for small, medium and large. 47

6.2 Dataset distribution after balancing 48

Acknowledgements

I want to thank Professor Rocco for giving me the amazing opportunity to work on this thesis, and for his support and availability even during the months I spent abroad. The earnest acknowledgements to Professor Amir Ghalamzan, for his professional guidance, for all the time and effort he dedicated to me, and for boosting my confidence. My gratitude to all the Intelligent Manipulation Lab and Lincoln Center for Autonomous Systems, for their technical help and above all for being my friends. I can't help holding you in my heart, together with my fellow adventurers Francesco and Gabriele, and with all the dear friends from the church and the climbing team I found in Lincoln.

I'm very grateful to my family, for their warmth and encouragement and for being my inspiration, and to Emanuele for his kindness and patience and for always being with me. An owing acknowledgement to my countless study mates, thank you for all the hours spent together, I cherish them all. I wouldn't be where I am and who I am today also without every other friend from Modena and Milano, for the forbearance they had toward my shyness, and for the certainty they helped me gain. Non nobis.