Open-Ended Affordance Discovery in Robotics Using Pertinent Visual Features

Pierre Luce-Vayrac

Thèse de Doctorat
de Sorbonne Université

Spécialité : **Informatique (EDITE)**

Présentée par : **M. Pierre Luce-Vayrac**

Pour obtenir le grade de
Docteur de Sorbonne Université

Open-Ended Affordance Discovery in Robotics Using Pertinent Visual Features

Thèse dirigée par **Raja Chatila**

Soutenue le 5 Juillet 2019

Rapporteurs	David Filliat	-	ENSTA Paristech, Saclay
	Justus Piater	-	UBIK, Innsbruck
Examinateurs	Mårten Björkman	-	KTH, Stockholm
	Stéphane Doncieux	-	ISIR, Sorbonne Université
	Eva Crück	-	Direction Générale de l'Armement
Directeur	Raja Chatila	-	ISIR, Sorbonne Université

Abstract

Scene understanding is a challenging problem in computer vision and robotics. It is traditionally addressed as an observation only process, in which the robot acquires data on its environment through its exteroceptive sensors, and processes it with specific algorithms (using for example Deep Neural Nets in modern approaches), to produce an interpretation: 'This is a chair because this *looks like* a chair'.

For a robot to properly operate in its environment it needs to understand it. It needs to make sense of it in relation to its motivations and to its action capacities. We believe that scene understanding requires interaction with the environment, wherein perception, action and proprioception are integrated. The work described in this thesis explores this avenue which is inspired by work in Psychology and Neuroscience showing the strong link between action and perception.

The concept of **affordance** has been introduced by James J. Gibson in 1977. It states that animals tend to perceive their environment through what they can accomplish with it (what it *affords* them), rather than solely through its intrinsic properties: 'This is a chair because I can *sit* on it.'.

There is a variety of approaches studying affordances in robotics, largely agreeing on representing an affordance as a triplet *(effect, (action, entity))*, such that the effect *effect* is generated when action *action* is exerted on entity *entity*. However most authors use *predefined* features to describe the environment. We argue that building affordances on predefined features is actually defeating their purpose, by limiting them to the perceptual subspace generated by these features. Furthermore we affirm the impracticability of predefining a set of features general enough to describe entities in open-ended environments.

In this thesis, we propose and develop an approach to enable a robot to learn affordances while simultaneously building relevant features describing the environment. To bootstrap affordance discovery we use a classical interaction loop. The robot executes a sequence of motor controls (action a) on a part of the environment ('object' o) described using a predefined set of initial features (color and size) and observes the result (effect e). By repeating this process, a dataset of *(e, (a, o))* instances is built. This dataset is then used to train a predictive model of the affordance.

To learn a new feature, the same loop is used, but instead of using a predefined set of descriptors of o we use a deep convolutional neural network (CNN). The raw data (2D images) of o is used as input and the effect e as expected output. The action is implicit as a different CNN is trained for each specific action. The training is *self-supervised* as the interaction data is produced by the robot itself. In order to correctly predict the affordance, the network must extract features which are directly relevant to the environment and the motor capabilities of the robot. Any feature learned by the method can then be added to the initial descriptors set.

To achieve open-ended learning, whenever the agent executes the same action on two apparently similar objects (regarding a currently used set of features), but does not observe the same effect, it has to assume that it does not possess the relevant features to distinguish those objects in regard to this action, hence it needs to discover and learn these new features to reduce ambiguity. The robot will use the same approach to enrich its descriptor set.

Several experiments on a real robotic setup showed that we can reach predictive performance similar to classical approaches which use predefined descriptors, while avoiding their limitation.

Contents

Chapter 1

Introduction

Contents

1.1 The Problem of Scene Understanding

Nowadays robots are being used in more and more contexts that goes way beyond the historical use in industry lines. They appear in household as helpers, from cooking robots to vacuum cleaners. In shopping malls, to guide customers. On the road as autonomous vehicles. One denominator to all these applications is the requirement for the robots to perceive their environments to be able to perform their tasks. A cooking robot must be able to find the tools it needs. A guide in the shopping mall should be able to avoid persons and obstacles (lest it might try swimming in decorative pools). And obviously it is mandatory for autonomous vehicles to be aware of its surroundings, should it be pedestrians or other vehicles.

What these robots require, more than simply perceiving their environment, is to make sense of it. To perceive can be as limited as to 'see' a bunch of pixels. To make sense of it is to know that this bunch of pixels is in fact a car. This process is called **scene understanding** and can be seen as interpreting the raw sensory inputs into abstract knowledge.

Scene understanding has been a long standing question in robotic, and has yet to be solved. In order to act in an environment a robot must be able to understand it. That is to make sense of the environment, of the scene that surrounds it. What is the context? Where am I? In what position am I? Is there movement around me? Many more questions could be considered, each of them being useful or necessary for a certain context or behavior. The problem is then to enable the agent to answer these questions.

How does a robot construct this knowledge? Essentially it consists in interpreting the raw sensory input perceived by the robot into higher dimension representations. If we except the cases where the robot acts blindly, every interaction requires

(to some extent) a minimal knowledge of the environment to be accomplished. It ranges from the global context, to the precise target for the action, passing by the local state of the agent. Moving requires the agent to be able to distinguish obstacles from safe ground. Grasping requires a model of objects and a way of segmenting said objects from the background. Activating a switch requires a physical model of the switch, and a representation of the global state that it will alter. Generally speaking, the more complex the interaction is, the more knowledge it will require.

Humans are proficient in scene understanding, they quickly and intuitively make sense of their surroundings (Oliva (2005). They are very efficient to select a relevant information from a large sensory input, and ignore the rest. Yet the problem remains open in robotics, with some tasks being very hard to solve. And it is often tasks that seem easy to humans that are the most challenging for robots. For instance we could consider the robot DeepBlue (Campbell et al. (2002)), able to best Garry Kasparov at playing chess, while still requiring a human help to move the pieces. The manipulation of the pieces on the board would first require to be able to identify the board itself and to distinguish it from the background. The robot would require a visual model of each pieces in order to recognize them. Lastly it would require precise motor controls to properly pick up pieces in a clustered board. Each task in itself is a challenge even for nowadays robotics. Especially if we consider the large variety of shapes, sizes and colors a chess board can come in. Therefore any method trying to tackle these problems has to be somewhat resilient to that change.

Traditionally in robotic scene understanding would be done by engineering the representations that will be necessary for the agent to complete its intended goals (Hanson (1978)). You would design a robot for a specific task, and therefore provide it with the required tools to perform this task. Which includes the perceptive functions for computing any necessary information from the raw sensory input. The main advantage is that the provided representations can be tested and validated. Furthermore since they are tailor-made to the task, they usually have a high level of performance. Consequently this approach requires to have beforehand knowledge of the context and specifics of the task. Which limits its usefulness to scenario where the information available to the engineer is sufficient. Sufficient regarding the complexity of the task, or the level of risk implied by the robot action, etc.

For instance industrial robots which operate in a strictly controlled environment. Their controllers and sensors are fine-tuned to accomplish a task with little to no room for un-predicted variations. In the other hand, autonomous vacuum cleaners need to operate in various homes but only require proximity sensors to do so. A certain degree of similarity is assumed regarding the disposition of apartments, and no particular risk is raised by the light weight and slow moving robot itself.

Thus scene understanding becomes increasingly difficult the further we stray robots from controlled and known environments. As you cannot predict the en-

vironment the robot will have to face, it is impossible to accurately design its behavior. Therefore it becomes necessary for the agent to be able to build new representations, or at least adapt pre-existing ones to the specifics of the actual environment it will have to operate in. Scene understanding becomes an active process. Moreover it becomes a life-long process.

1.2 Features Learning through Ambiguity Reduction

This thesis address the problem of scene understanding through the spectrum of interactive perception (Bohg et al. (2017)), and more precisely through the concept of affordances (Gibson (1977)). The concept of affordances states that one's perception of the environment is based on what one can accomplish with it rather than based on its intrinsic properties. As an agent, I perceive the environment through what I can do with it, through what it affords me.

In a first attempt to tackle this problem we proposed a Bayesian approach for learning affordances through interaction. However this approach revealed several limitations that conducted us to propose a second method for learning affordances through the reduction of ambiguities. What we call ambiguities are the states in which an agent observe contradictory perceptual inputs after interacting with the environment.

The main contribution of this thesis lies in the proposed approach (chapter 4) to simultaneously learn affordances while constructing the relevant visual features.

1.3 Contributions

In this manuscript we aimed to answer the following question:

> *How can a robotic system be enabled a life-long ability to discover affordances in an unconstrained environment?*

Which has led to two main contributions:

- A probabilistic representation of affordances and of their combinatory nature.
- An ambiguity based method to construct pertinent visual features using CNN simultaneously to discovering affordances.

1.4 Thesis Structure

This manuscript is structured as follows:

Chapter 1 is a general introduction to the thesis. We present the overall context, and outline the motivation and goal of this study.

Chapter 2 presents a quick history of Active Perception methods and the state of the art in Affordance theory applied to robotics. We firstly try to introduce the concept of Active Perception and details some of the reasons why nowadays robotics could require such approaches. Then we express the link and relationship between Active Perception, Interactive Perception and Affordance theory. We emphasis the way all three approaches pervade and refine each other. Finally we present a short state of the art of applied affordance theory to robotics, and more specifically for learning visual representations of the environment.

Chapter 3 presents our base approach at learning affordances using Bayesian Networks. We propose a method for learning probabilistic affordances through interactions. The interactions produce a dataset of sensorimotor information that we use to learn the structure of a Bayesian Network. The structure of this network represents the discovered affordances. Then we discuss the possibility of composing / decomposing affordances through the composition / decomposition of objects.

Chapter 4 proposes an extension to the method presented in 3 by removing the constraint on predefined features and proposing an ambiguity based approach to discovering affordances. We emphasis the importance of any a priori information given to the agent in orienting or limiting the discoverable affordances. We then attempt to reduce this limitation by proposing a method to learn both descriptors of the environment and affordances simultaneously.

Chapter 5 aims to draw some conclusions about the contributions and limitations of this study. We discuss the extensions and future work. As well as the possible case of use of the presented methods.

Finally in Annex A the reader will find some background information about the methods that we used but were not described in length in previous chapters, additional results regarding chapter 4, and details of algorithms presented throughout this manuscript.

Chapter 2

Context: From Perception to Affordance

Contents

2.1 Introduction

This chapter presents a state of the art regarding the methods of perception in robotic. The focus of this thesis is to discuss the acquisition of sensorimotor representations (affordances) by a robot through interaction. Therefore the main part of this chapter will be about the history of affordances learning in section 2.5. In which we will discuss the different methods to leverage the theory of affordances for enabling a robot to better understand it's environment.

However we chose to consider first the history of perception in its whole, and how the concept of Affordance might have pervaded and influenced the domain.

That is to go from Passive Perception to Interactive Perception through active perception. We will try to present the cumulative progress in these fields and how they are directly useful to building affordances in a robot.

The goal is to explain what all three domains have in common, and how they differ, in order to better emphasis what each one brought to the overall domain of perception, and how it can be leveraged to learn affordances.

Passive, Active and Interactive Perception are methods of perception, while Affordance is a paradigm on how the environment is perceived. In other words, we use AP methods to build affordances.

Thus this chapter will try to give a short history of Perception in robotics, to introduce the concepts of Active Perception and Interactive Perception. And then we will present a review of the current state of the art in Affordance theory applied to robotics.

2.2 Passive Perception

Passive Perception is mostly defined by opposition to Active Perception. However and in order to set a sort of baseline for the upcoming sections, we will try to give a formal definition of it.

A perceiver is passive when there is no active process regarding the acquisition process of the information.

The way the information is perceived or gathered is not altered by the motivation of the agent or its current mind state. And the agent does not take any action in order to specifically perceive a certain information.

Most current robotic applications are passive observer. Industry line robots possess sensors to regulate the timing of their sequence of control commands or to detect a breach of safety measures. Vacuum cleaner robots use infrared sensors or contact bumpers to detect obstacles. Yet none of them actively decide to perceive. They receive a constant flow of information that they interpret based on a fixed set of rules. They do not change the way their sensors work nor do they execute a motor control sequence in order to receive a specific signal. In other word they never interfere in the process of perception. Although *what* they perceive might influence the actions they take, they do not take actions to influence *what* or *how* they perceive.

In many contexts this approach is more than enough to enable a robot to perform its intended tasks. And it also bring a few advantages. Since the perception process is constant, the behavior of the robot is equally predictable. Furthermore there is no additional computation cost to select the perception process.

This approach fit well any reactive programming methods (Nilsson et al. (2002)). Where an agent is designed to react to its environment by executing predefined motor controls based on predefined perceptive cues.

However it is unsuited for unconstrained and varying environments. Any changes in the environmental conditions will test the resilience of the program, to the point where it will require some degrees of adaptation. Thus the study of computer vision requires to integrate an active aspect.

Furthermore the field of ecological psychology has long been claiming that perception is not at all passive, but the result of a sensorimotor process. Sensorimotor describing here the pairing of the sensor and motor systems.

For instance J.J. Gibson proposes that we animals perceive their environment in terms of actions possibilities (Gibson (1977), Gibson (1979)), which he called *affordances*.

Similarly K. O'Regan proposes that the ability to perceive is the result of a learning process, during which the agent learns to identify sensorimotor contingencies. A *sensorimotor contingency* represent the correlation between an action and sensory input. By exploring its motor capabilities, the agent also explores the related sensory contingencies. It is through this exploratory process that the agent is able to develop a representation of its environment.

These studies about natural perception have inspired the field of computer vi-

sion, as we will discuss in the next section. For a more complete history of perception the reader should refer to Pastore (1974), and to Wade (2000) for natural history of vision.

2.3 Active Perception

As defined in Bajcsy et al. (2018), the essence of active perception is:

> *"to set up a goal based on some current belief about the world and to put in motion the actions that may achieve it."*

In the process of perception, an active observer is one that not only receives sensory input, but which *acts* in order to sense.

As stated in Bajcsy et al. (2018),

> *"The fundamental difference between an active perception system and other perception systems lies in action, or lack of it. Whereas both types of systems include decision-making components, only the active system includes dynamic modulations to the overall agent's behavior, both external (via motors) and internal (via parameter configurations)".*

Among the earliest work on active perception is the thesis dissertation of J.M. Tenenbaum (Tenenbaum (1970)). In this study the author proposes to adapt a computer vision system to its environment by controlling its camera's parameters. The difference from traditional image processing systems lies in the fact that here the sensor is automatically accommodated and treated as an integral part of the recognition process.

Although one of the first author to directly discuss the concept of Active Perception is Ruzena Bajcsy in Bajcsy (1988).

In 1988 R. Bajcsy wrote:

> *"Active sensing is the problem of intelligent control strategies applied to the data acquisition process which will depend on the current state of data interpretation including recognition."*

Active perception is the problem of actively controlling an otherwise passive process, sensing, in order to collect data that will be adapted to the agent's current state of mind. Two aspects of perception can be controlled, the tuning of the sensors and the interpretation of the sensed data.

The tuning of sensors allows the agent to alter the intrinsic parameters of its sensors. Thus maximising the quality of perception. While the interpretation could be compared to the post-processing of a raw flux. The agent is continuously irrigated through its sensors, the interpretation process executes a selection and interpretation of the low level feed into a higher dimension abstraction.

For instance our eyes adapting to changes of illumination is a control of sensor. While the monkey business illusion (Simons (2010)) is a very good example of how the state of mind influences the interpretation and filtering of the sensory input.

In more recent work (Bajcsy et al. (2018)) they propose a refined definition of an active perceiver:

> "An agent is an active perceiver if it knows why it wishes to sense, and then chooses what to perceive, and determines how, when and where to achieve that perception."

They emphasis the notion of active perceiver around 5 criterion, *why*, *what*, *how*, *when* and *where*. Which they define in the following table: 2.3.

Active Perception	Definition
Why	The current state of the agent determines what its next actions might be based on the expectations that its state generates. These are termed Expectation-Action tuples. This would rely on any form of inductive inference (inductive generalization, Bayesian inference, analogical reasoning, prediction, etc.) because inductive reasoning takes specific information (premises) and makes a broader generalization (conclusion) that is considered probable. The only way to know is to test the conclusion. A fixed, pre-specified, control loop is not within this definition
What	Each expectation applies to a specific subset of the world that can be sensed (visual field, tactile field, etc.) and any subsequent action would be executed within that field. We may call this Scene Selection
How	A variety of actions must precede the execution of a sensing or perceiving action. The agent must be placed appropriately within the sensory field (Mechanical Alignment). The sensing geometry must be set to enable the best sensing action for the agent's expectations (Sensor Alignment, including components internal to a sensor such as focus, light levels, etc.). Finally, the agent's perception mechanism must be adapted to be most receptive for interpretation of sensing results, both specific to current agent expectations as well as more general world knowledge (Priming)
When	An agent expectation requires Temporal Selection, that is, each expectation has a temporal component that prescribes when is it valid and with what duration
Where	The sensory elements of each expectation can only be sensed from a particular viewpoint and its determination is modality specific. For example, how an agent determines a viewpoint for a visual scene differs from how it does so for a tactile surface. The specifics of the sensor and the geometry of its interaction with its domain combine to accomplish this. This will be termed the Viewpoint Selection process

Table 2.1: The five main constituents of an actively perceiving agent are defined Bajcsy et al. (2018)

Although AP does not limit the range of action to solely modifying the inner state of the agent. It does not either emphasis the need to interact with the environment. Which is why the field of Interactive Perception has diffenrentiated itself.

2.4 Interactive Perception

[citation to incorporate: Battaglia et al. (2016) interaction networks to learn physics, Högman et al. (2016) sensorimotor learning framework for object categorization,]

Interactive Perception differs from **Active Perception** exactly on that emphasis about interaction. Interactive Perception pose as a prerequisite that some information is inaccessible through observation alone. Thus it is only through interaction that the environment will truly make sense to the agent (Bohg et al. (2017)).

This concept is deeply linked to the psychological concept of affordance (Gibson (1977).

We argue that Interactive Perception constitutes an extension to Active Perception, rather than being of a complete different nature. It stress the importance of

interaction in perception. To gather pertinent information about the environment requires to interact with it.

The main incentive for Interactive Perception (IP) over Active Perception (AP) is that it allows the user to gather more information about the environment. Information that could not be obtained solely through vision, like the texture or weight of objects. The agent needs to *interact* with the environment in order to gather such information.

Whereas in AP the perceiver does not necessarily interacts with the environment, IP emphasis the importance of interaction to gather information. Either because the information might be hidden otherwise (texture, weight), or because the information is temporal, and relative to an action (i.e. an effect). Thus the very nature of the information changes. It is not only a question of observing the environment, but to modify it in order to gather new knowledge.

In the recent study Bohg et al. (2017), they talk of *forceful interaction* to describe an IP process. Which they define as *"Any action that exerts a potentially time-varying force upon the environment is a forceful interaction"*.

In order for the 'agent' to interact with the 'environment', the 'agent' needs to have a sense of its own *embodiment*. By that we mean that it needs to be able to differentiate 'itself' from its environment.

To that effect the concepts of **Proprioception** and **Exteroception** are used. They distinguish the perceptive capabilities of an agent in terms of its own embodiment, to draw the frontier between what is 'internal' and 'external' to it.

Proprioception describes all the sensory inputs that are internal to an agent's embodiment. Respectively **Exteroception** describes all the sensory inputs that are external to the agent's embodiment.

The overall objective of IP is to improve the process of perception by coupling proprioception and exteroception.

Through forceful interactions new sensory signals are created. Signals that could not be obtained through an observation process alone. These signals encapsulate the relation between action, sensory input, and time. Thus it enables the perceiver to learn to predict the consequences to its actions. By multiplying the interactions over time, the agent is able to discover the regularities of the action-sensory loop. Thus enabling the agent to model the causal relationship between actions and sensory inputs.

2.5 Affordances

Naturally IP finds its way to the concept of **Affordance**. To learn affordances is to discover the relata between one's actions and one's perception. Thus the process of learning affordances is by definition an IP process. We go further by saying that IP is a process to acquire information, while the theory of affordances is a theory of how the world is perceived.

The concept of Affordances was first introduced by psychologist James J. Gibson

in the book "**The Senses Considered Perceptual Systems**" in 1966 Gibson (1966). However the concept as we use it was more clearly refined in his 1979 book, "**The Ecological Approach to Visual Perception**" Gibson (1979). In which it is defined as follows:

> "The affordances of the environment are what it offers the animal, what it provides or furnishes, either for good or ill. The verb to afford is found in the dictionary, the noun affordance is not. I have made it up. I mean by it something that refers to both the environment and the animal in a way that no existing term does. It implies the complementarity of the animal and the environment. (p. 127) ... [Objects] can all be said to have properties or qualities: color, texture, composition, size, shape and feature of shape, mass elasticity, rigidity, and mobility. Orthodox psychology asserts that we perceive these objects insofar as we discriminate their properties or qualities. Psychologists carry out elegant experiments in the laboratory to find out how and how well these qualities are discriminated. The psychologists assume that objects are composed of their qualities. But I now suggest that what we perceive when we look at objects are their affordances, not their qualities. We can discriminate the dimensions of difference if required to do so in an experiment, but what the object affords us is what we normally pay attention to. The special combination of qualities into which an object can be analyzed is ordinarily not noticed. (p. 134)"
>
> —Gibson (1979, p. 127)

To learn affordances is to construct representations of the environment that will be directly linked to the motor capabilities of the agent. Therefore by definition to learn affordances can only be done through interactions with the environment.

Affordances naturally find their way in robotics. They represent and express the embodiment of intelligence in an animal, therefore they can be used to create such an embodiment of a robot's own "intelligence". It enables the robot to build representations and concepts that will be grounded in its own physical and perceptive capabilities. It constrains the model to integrate the limitations of the agent as well as it's specificity. Which in turns means that the constructed representations are more meaningful to the agent because they are specific to it's own embodiment and motivation.

In other terms, a model of the theory of affordances applied to robotics should satisfy two main criteria:

- Different robots, with different motor and perceptive capabilities should not construct the same representations of the environment. There is no reason for the tall pinch-gripper robot to acquire the same affordances as the small vaccum-gripper one.
- Two identical robots, in the same environment, but with different motivations should not construct the same representations. Even in the same context and

with identical capabilities, the motivation is the primary drive to learning, and therefore should differentiate the learned affordances.

Over the years the fields of affordances in robotics as seen many contributions, as the recent study from Zech et al. (2017) shows. However the field is not recent and his prone to debate as to what exactly is and affordance (Jones (2003), Dotov et al. (2012)).

Chapter 3

Probabilistic Learning of Affordances through Interaction

The results and text of this chapter have been published in the following articles.

- Chavez-Garcia, R. O., Andries, M., Luce-Vayrac, P., and Chatila, R. (2016a). Discovering and Manipulating Affordances. 2016 International Symposium on Experimental Robotics (ISER).
- Chavez-Garcia, R. O., Luce-Vayrac, P., and Chatila, R. (2016b). Discovering Affordances through Perception and Manipulation. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3959–3964. IEEE.
- Chatila, R., Renaudo, E., Andries, M., Chavez-Garcia, R.O., Luce-Vayrac, P.,Gottstein, R., Alami, R., Clodic, A., Devin, S., Girard, B., and Khamassi, M. (2018). Toward Self-Aware Robots. Frontiers in Robotics and AI, 5:88.

Contents

3.1 Introduction

The problem of scene understanding has been addressed in AI and Robotics since the 1970's. Despite numerous results over this long time period, the problem of

actually **understanding** a scene by a robot is still open. Understanding is indeed not just about attaching labels on image regions, it's actually about grounding perceptual representations in the reality of the world. This cannot be achieved by a process of observation alone and requires interaction between the agent and its environment.

This chapter presents an approach for enabling robots to understand their environment by interacting with it. We study and develop sensorimotor representations and scene interpretation processes based on visual and proprioceptive inputs when the robot physically interacts with objects. The processes build models of objects based on perceptual clues and effects of robot actions on them, which relate to the notion of affordances.

In this work we follow a bottom-up approach that starts from low-level data from sensors and actuators, up-to learning relations between higher-level representations. The probabilistic nature of the work maintains the uncertainty characteristic of the perception-action cycle. Our sensorimotor representation encodes, through the learning of affordances, effects, objects and actions in the same feature space. It enhances works such as Lopes and Santos-Victor (2007) and Ugur et al. (2015) by including an information-based methodology to find the relations in the combined feature space instead or relying on a preferred-relation list. Following the categorization proposed by Bohg et al. (2017), our work can be considered a multimodal Interactive Perception approach with given priors on the robot dynamics, and on the observations. It has as goals automatic object segmentation, estimation of intrinsic object parameters, sensorimotor learning, and eventually semantic categorization.

This chapter 3 is organised as follows: In section 3.2 we present the affordance model that we use and the corresponding definitions, along with the method of sensory perception. In section 3.3 we present the affordance learning architecture. In section 3.4 we extend the model to discuss the nature of affordances and their mutual relations. Finally in section 3.5 we make a partial conclusion and discuss the limits of the proposed method.

Parts of the texts and results of this chapter have been published in the following articles Chavez-Garcia et al. (2016b), Chavez-Garcia et al. (2016a) and Chatila et al. (2018).

3.2 Affordance Model and Definitions

In this section we present the model used in our approach. In part 3.2.1 we define the model of affordance that we use. In part 3.2.2 we present the way the robot will actually perceive its environment through sensors.

3.2.1 Affordance Definition

As presented during the introduction, the concept of affordance has been largely studied, and has therefore taken a lot of different aspects (Zech et al. (2017)). Each

aspects fitting a different context or a different question to which the authors are trying to give an answer. This is mainly possible due to the abstract nature of the concept of affordance. Although the idea is based on work in psychology and neuroscience (Gibson (1977), Gibson (1979), O'Regan and Noë (2001)), the actual implementations in robotics are usually very distant from it.

Which contributes to the popularity of this idea as it enables authors to fit the concept of affordance to their specific approaches.

In this work we chose to use the formalism described in Sahin et al. (2007), where an affordance is defined as an $(effect, (entity, behavior))$ tuple such that:

> *"An affordance is an acquired relation between a certain* ***effect*** *and an* ***(entity, behavior)*** *tuple, such that when the agent applies the* ***behavior*** *on the* ***entity****, the* ***effect*** *is generated."*

In this formalism the affordance is considered from the point of view of the agent, therefore the agent is implicit in the definition, and all components are assumed perceived by the agent. An explicit version would be to formulate an affordance as $(agent, (effect, (entity, behavior)))$. Which means that from the point of view of *agent* executing *behavior* on *entity* is expected to produce a certain $effect$. We could argue that in essence an affordance should only be considered from the point of view of the agent. To explicit the agent in the formalism implies that the *behavior* and / or *entity* are defined independently of it. That there is a shared perception or concept of this behavior and entity between the agent and the observer.

We chose to use this formalism because of the high level of abstraction it offers. This representation can be used in a large variety of context as the elements that composes it (entity, behavior and effect) are of extremely general purpose. A **behavior**, an **entity**, or an **effect** encapsulate a wide variety of concepts.

Two triplets example could be (lifting, (end-effector, lifted)) and (lifting, (switch, lights on)). In the first case the behavior of lifting is applied to the end-effector, and the result is the 'lifted' state of the end-effector, while in the second example the behavior of lifting is applied to a switch, which results in the effect of turning on the lights. In one case the end-effector is considered as the target of the action, while in the other it is part of it.

Both examples are equally valid in the aforementioned affordance paradigm, and point out the diversity of implementations in the affordance literature.

We believe it is important to stress this fact as the definition of affordance is extremely prone to interpretations. For instance in this basic example one could say that the target entity for pushing was not the room but the switch. Which raises the question of the *tools*.

In this particular work we focus on discovering the relations between effects, entities and behaviors, rather than discovering or constructing those components themselves. Therefore we make assumptions on effects, entities, and behaviors that will be described respectively in 3.2.1.3, 3.2.1.1 and 3.2.1.2.

3.2.1.1 Object

In the formalism described in Sahin et al. (2007) an *entity* is the *target* of an action. It is a part of the perceivable environment upon which an action can be executed.

In this work we chose to focus on learning affordances of physical entities, which we will call (for lack of a better word) objects from now on. While an entity can encapsulate abstract concepts, like ideas or state of mind, we mean by object a **physical** entity with which the robot can **physically** interact. Which therefore limits us to objects that are reasonably proportional to the size of the agent. We will use tabletop scenarios with objects roughly the size of the end-effector of the robot, but the whole architecture and therefore conclusions remain general.

To keep the general idea that objects are defined through affordances and do not pre-exist them, we define an *object* as a small part of the environment that is perceived by the agent as interesting for interaction. To do so requires the agent to possess some segmentation capabilities that will allow it to separate from its raw perceptual input what we will define here as a 'foreground' (what can be interacted with) from a 'background' (what cannot be interacted with). This capability could in itself be considered as an affordance, which could be formulated as "what in my environment affords me 'interactability' ", and is in fact a research topic in itself. However we chose to consider this capability innate in our agent, and therefore provide for him a few set of rules to do this segmentation. Those rules can be considered as the agent's innate sensitivities to regularities and convexity.

Using this set of rules the agent construct a set of objects hypotheses which are then described with a set of predefined features (color, size, shape).

To summarize in our approach an object o from a triplet $(e,(a,o))$ is defined as a set of features values such that $o = \{o_{color}, o_{size}, o_{shape}\}$ with o_{color}, o_{size} and o_{shape} respectively the color, size and shape of object o.

The exact details of the method to segment object hypotheses from the perceived environment will be detailed later in 3.2.2.1 and the features definitions and extraction will be presented in 3.2.2.2.

Using a set of features to describe objects serves two main objectives. It reduces the dimensionality of an object hypothesis, and it 'anonimizes' objects so that we learn affordances regarding the properties of the objects rather than regarding an object specifically.

3.2.1.2 Action

Similarly to the definition of object, we refine the definition of behavior towards the one of action. In Sahin et al. (2007) a behavior is defined as:

> "...the part of the agent that is generating the interaction with the environment that produced the affordance. Ideally, the agent's relata should consist of the agent's embodiment that generates the perception–action loop that can realize the affordance. We chose the term behavior to denote this. In autonomous robotics, a behavior is defined

> as a fundamental perception–action control unit to create a physical interaction with the environment. We argue that this term implicitly represents the physical embodiment of the interaction and can be used to represent the agent's relata."

In our approach we consider three levels of motor controls, motor primitives, actions and skills. That we define as following:

- A motor primitive is a direct control of one degree of freedom from the robot. It corresponds to a direct motor commands. For instance rotating a joint by 10 degree is a motor primitive. Each motor primitive is defined in the frame from the activated joint.

- An action is a sequence of motor primitives. And is defined in the global frame of the robot. For instance advancing the end-effector forward is an action, that combines the activation of several joints in the arm.

- A skill represent a goal and the sequence of motor primitives (or action) that can accomplish this goal. It is defined in the world reference frame. For instance pushing is a skill, the goal is to push a target and one possible action is to apply a force on the target through the movement of the end-effector.

Making an analogy to a newborn rough motor abilities Li and Meng (2015), we assume that the robot is built with a set of basic motor capabilities, which we call actions. In this set of basic actions $A = \{a_1, ..., a_n\}$, each action can be defined as follows:

$$a_i(V^*, \gamma, \sigma_{a_i}), \tag{3.1}$$

where V^* represents the desired value for the robot controlled variables V, γ is its propioceptive feedback and σ_{a_i} represents the parameters for the particular action a_i.

3.2.1.3 Effect

In Sahin et al. (2007) an effect is defined as:

> "...the interaction between the agent and the environment should produce a certain effect. More specifically, a certain behavior applied on a certain entity should produce a certain effect, for example, a certain perceivable change in the environment, or in the state of the agent."

Refining this definition we define an effect as a possible correlation between an action (as defined in 3.2.1.2) and a change in the perceptual state of the environment, which includes the agent itself. For instance, when a robot interacts with an object, it can perceive changes related to the state of the object, to the proprioceptive values of the actuators and to the feedback from the end-effectors.

The concept of effect is an important element in our sensorimotor fusion and its detection (or lack of it) plays the role of common ground for perception and action frames.

In the same fashion that we equipped our agent with pre-existing object descriptors and action primitives, we provide for the agent a set of basic effect detector. The robot's innately detectable effects are divided in two groups: perceptual-based (object's linear movement); and propioceptive-based (end-effector linear force, distance between gripper's fingers and effector's linear movement).

3.2.2 Sensory Perception

In previous section 3.2.1 we defined the framework to represent affordances in our work, this section will focus on presenting the methods used to perceive the environment and, in fine, fill those representations.

It includes two main aspects, (1) the segmentation of the environment into objects hypothesis 3.2.2.1, and (2) the detection of effects during interactions 3.2.2.4.

3.2.2.1 Objects Segmentation

In order to interact with objects the agent must first be able to identify them as such. Which mean that using the perceptive capabilities available to it, it must be able to distinguish them from the background and from each other. However a core idea of our approach is to build a system that would adapt to different contexts and environments, therefore we need to provide as little information as possible to the agent. Especially information that would be specific to the current situation, as it would completely defeat the purpose of a generic approach to affordance discovery and learning. Therefore we limit the a priori knowledge of the agent to planar extraction and a bottom up segmentation approach to create objects hypothesis.

Usually, segmentation algorithms only consider low-level information from the image or point cloud. Recent semantic segmentation methods take advantage of high-level object knowledge to help disambiguate object borders Van Hoof et al. (2014); Silberman et al. (2012). However, the computational cost of inference on these methods rises with the increasing data resolution.

We chose to use a bottom-up approach to segmentation based on an over segmentation algorithm Papon et al. (2013). Supervoxels are formed by over-segmenting a 3D image into small regions based on local low-level features, reducing the number of nodes which must be considered for segmentation. We use a 3-D version of the Voxel Cloud Connectivity Segmentation (VCCS) presented in Papon et al. (2013), which takes advantage of 3D geometry provided by RGB+D cameras to generate supervoxels evenly distributed in the observed space, rather than the projected image plane. VCCS uses a seeding methodology based on 3D space and a flow-constrained local iterative clustering which uses color and geometric features. The seeding of supervoxel clusters is done by partitioning 3D space. This ensures that supervoxels are evenly distributed according to the geometry of the scene. Due to

ensuring strict partial connectivity between voxels, this algorithm guarantees that supervoxels cannot flow across boundaries which are disjoint in 3D space.

Supervoxels features are represented by 39-dimension vector composed of spatial coordinates (x, y, z), color information (Lab color space), and 33 elements from an extension of the Point Feature Histogram Papon et al. (2013):

$$F = [x, y, z, L, a, b, FPFH_{1..33}] \tag{3.2}$$

This offers a multi-dimensional pose-invariant representation based on the combination of neighboring points. For each supervoxel, in an outward direction, we calculate a normalized spatial distance D_s, a normalized color distance D_c and the distance in the FPFH space D_{HIK} Papon et al. (2013), from the center of the supervoxel (cluster) to the adjacent voxels. If the distance is the smallest seen, this voxel and its neighbors (in the adjacency graph) become part of the supervoxel.

The result, as is shown in figure 3.1, is an over-segmented cloud where each supervoxel (segment) cannot cross over object boundaries that are not actually touching in 3D space. Supervoxels tend to be continuous in 3D space, since labels flow outward, at the same rate, from the center of each supervoxel Papon et al. (2013).

Figure 3.1 shows how supervoxels are still considered representations of individual patches. A clustering process is needed to group the supervoxels that possibly correspond to the same object without relying on *a priori* information of the number of objects. Regarding the feature representation detailed here, we use the non-parametric technique described in Comaniciu and Meer (2002) combined to a locally convex criterion presented in Christoph Stein et al. (2014) to find the shape of the object hypotheses based on the set of supervoxels.

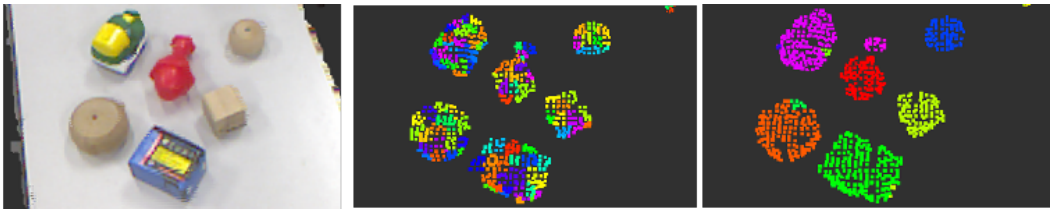


Figure 3.1: Results from the sensory perception process. RGBD cloud of points from the real scenario (left); over-segmentation results from point cloud (middle); results from intrinsic clustering (right).

Figure 3.1 shows the result of the clustering method. The result of this intrinsic clustering is a set of labels $L_{hyp}(t)$ for a group of supervoxels that represent hypotheses of objects in the current scenario.

The set of generated hypotheses from Section 3.2.2.1 are built only using the sensory data. This means that segmentation issues can appear in the form of incomplete, divided and false segments of real objects in the scenario. We perform a tracking-by-detection approach to reduce the number of false positive segmentations. Only the active segments hypotheses with tracks lengths over a threshold

τ are considered as confirmed object for our sensory-perception task. Each object is represented by its centroid, which offers a point of interaction (*poi*) for the interaction task.

3.2.2.2 Features extraction

In this work, we assume that the robot has innate perceptual capabilities that allow it to discretize the environment. This capabilities are related to the segmentation approach. It can differentiate from color values. It has geometrical notion of position, continuity of segments and normal extraction for surfaces. Therefore, the robot can extract higher level features for the description of confirmed objects. By analyzing the cloud of points representing the object, we focus on three main features: *color*, *size* and *shape*. Transforming from RGB to HSV color model, we extract the dominant hue of the object. Size of the object is obtained from the distance between the start and end of the largest segment of the cluster representing the object. Four-dimensional templates are used to select the form of the object from a set of fixed three-dimensional forms: *cube*, *cuboid*, *sphere*, *irregular*. Our architecture allows for expanding and learning the set of perceptual features.

3.2.2.3 Sensorimotor Learning

In addition to the perceptual information, object manipulation allows the robot to learn sensorimotor correlations between the sensor inputs fused in the objects descriptions O, robot basic actions A and the salient changes represented by the effects E. Starting from built-in actions, the development of the environment is captured by perception through the information provided by effect detectors, e.g. object movement detection and proprioceptive feedback. The goal is to learn from regularities in the occurrences of elements in O and E when an action $a_i \in A$ is triggered.

3.2.2.4 Effect Detectors

In the same way that we consider innate action capabilities, we consider the agent enable to detect effects. Therefore we provide effect detectors to the robot, so that it can observe the results of its actions.

By effect detector we mean a module that given temporal inputs can compute an 'effect'. I.E. evaluate if a certain event occurred during the interaction.

In our case an effect detector take two inputs, the initial and final states of the scene, and compute an effect value.

3.3 Affordances Learning

3.3.1 Proposed Method

An affordance is an *acquired* relation between two interacting elements E and T, where E is a set of effects and T is a tuple composed of a capability (in our case an action) in A over an entity in O. One can state that when an agent g applies its capability a over an entity o, an effect e is generated Sahin et al. (2007). From an agent's perspective, from now on the robot, the i_{th} affordance is defined as follows:

$$\alpha_i = (e_l, (o_k, a_j)), \text{ for } e_l \in E, a_j \in A \text{ and } o_k \in O. \tag{3.3}$$

Figure 3.2 shows an example of a relation between an entity *toy* perceived by the agent *robot*, and the application of its capability *grasp*, implying that there is a potential of generating an effect *grasped*. We can label this relation using its semantic value, $grasp - ability$.

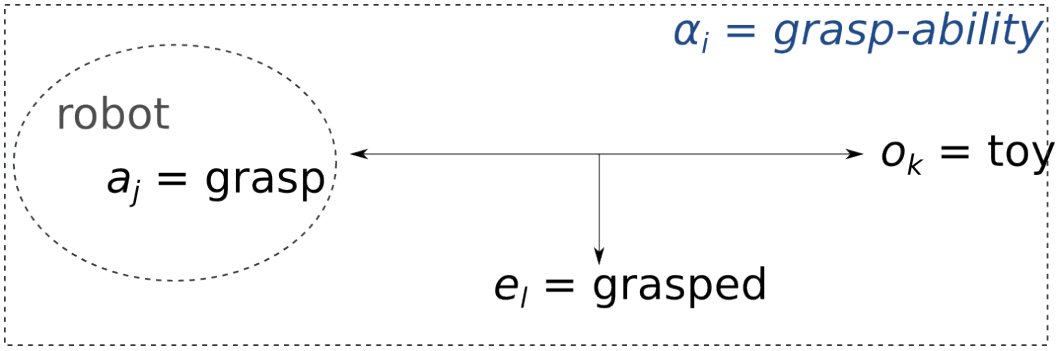


Figure 3.2: Representation of an affordance relation labeled $grasp - ability$.

In sections 3.2.1 and 3.2.2 we have defined the three elements mentioned in our affordance definition. We can state our problem as learning the set of relations $\mathscr{A} = \{\alpha_1, ..., \alpha_n\}$ for a set of data extracted from E, O, and A.

3.3.1.1 Structure Learning for Affordances Discovery

Let us represent the members of the set of elements E, O and A as discrete random variables of a Bayesian Network(BN) $\mathscr{G}$. Therefore, we can define these elements as the discretization $E = \{e_i\}$, $O = \{o_j\}$ and $A = \{a_k\}$. Let us assume that through the cycle of perception-interaction we obtained instances of these variables generating a data set $\mathscr{D}$. Our problem of discovering the relations between E and T can be translated to finding dependencies between the variables in $\mathscr{G}$, i.e., learning the structure of the corresponding BN from data $\mathscr{D}$. Using the BN framework we are capable of displaying relationships between variables. The directed nature of its structure, allows us to represent cause-effects relationships. It can handle uncertainty through the establish probability theory. In addition to direct dependencies, we can represent indirect causation. Therefore here the structure of the network represents the induced affordances discovered by the robot through interaction.

One approach for inducing BN structures from data is the score-based technique, especially for the purpose of probability distribution function estimation. The process assigns a score to each candidate BN that measures how well that BN describes the data set $\mathscr{D}$. For a BN's structure $\mathscr{G}$, its score is defined as the posterior probability given the data $\mathscr{D}$:

$$Sc(\mathscr{G}, \mathscr{D}) = P(\mathscr{G}|\mathscr{D}). \tag{3.4}$$

A score-based algorithm attempts to maximize this score. Usually, this score is rewritten using Bayes' rule as:

$$Sc(\mathscr{G}, \mathscr{D}) = \frac{P(\mathscr{D}|\mathscr{G})P(\mathscr{G})}{P(\mathscr{D})}, \tag{3.5}$$

where algorithms only need to maximize the denominator, since $P(\mathscr{D})$ does not depend on $\mathscr{G}$. If we assume a uniform prior over the structures, we can focus only on $P(\mathscr{D}|\mathscr{G})$. Usually, score functions work on the logarithmic space, i.e., $log(P(\mathscr{D}|\mathscr{G}))$. These algorithms select various structures for examination and score them. The structure with the highest score is selected. In this work, we implement an information-based score.

3.3.1.2 Information compression score

We can define the score of a BN as the compression rate of the data $\mathscr{D}$ with an optimal code induced by the BN. $\mathscr{D}$ represents the interactions of the robot with values for the variables in O, A and E. Using Shannon's noiseless coding theorem, we establish the limits of the compression rate. Therefore, as the number of independent and identical distributed random variables tends to infinity, no compression of the data is possible for a rate less than the Shannon entropy, without losing information Shannon (1948).

Bayesian Information Criterion (BIC) is a generalization of the Minimum Description Length (MDL) score, which uses a penalization based on the number of bits needed to compress $\mathscr{D}$, preferring simple BN over more connected and complex ones. We calculate the quality of $\mathscr{G}$ as:

$$\eta(\mathscr{G}|\mathscr{D}) = I(\mathscr{G}|\mathscr{D}) - s(N)|\mathscr{G}| \tag{3.6}$$

where I is the log-likelihood score that measures the number of bits needed to describe $\mathscr{D}$ given $P(\mathscr{G})$, and $|G|$ denotes the network complexity. BIC uses a penalization defined as $s(N) = \frac{log(N)}{2}$ to represent the number of bits needed to encode $\mathscr{G}$. In order to increase the likelihood of a structure, we can add parameters, which can result in overfitting. BIC penalizes structures with larger number of parameters Schwarz et al. (1978).

3.3.1.3 Search algorithm

Our implementation is based on the hill-climbing technique, for learning BN structures. As inputs, this algorithm takes values for the variables in E, O, and A obtained from robot's interaction.

The procedure *EstimatePDFs* estimates the parameters of the local pdfs given a BN structure. Typically, this is a maximum-likelihood estimation of the probability entries from the data set, which for multinomial local pdfs consists of counting the number of tuples that fall into each table entry of each multinomial probability table in the BN. The algorithm's main loop consists of attempting every possible single-edge addition, removal, or reversal, making the network that increases the score the most the current candidate, and iterating. The process stops when there is no single-edge change that increases the score. There is no guarantee that this algorithm will settle at a global maximum, but there are techniques to increase its reaching possibilities, we use simulated annealing Tsamardinos et al. (2006).

3.3.1.4 Motivational system

Interaction is a task that can easily become intractable due to the large number of interaction possibilities. We need a mechanism that guides this interaction and promotes the perception learning process based on intrinsic stimuli. Starting with random exploration, we use a motivational system to focus on object hypotheses closest to the end effector to generate (object, action) couples.

3.3.2 Experiments and Results

The interest of the experiments is mainly in understanding and relating the effects generated by the set of basic actions. Our experiments rely on two assumptions. First, when the robot repeatedly performs a particular action over a particular object, the obtained effect is mostly the same. Second, explicit information regarding the success of an action is not provided; it is obtained through inference over the learned structure of a BN. Therefore, our experiments are carried in an unsupervised fashion.

3.3.2.1 Experimental setup

Our Baxter robot (see Figure 3.3) is equipped with 2 arms with 7 degrees of freedom, a sensor array of one camera in each arm and one on the head. One electrical gripper is attached to each arm for manipulation purposes. Additionally, a range camera (Microsoft kinect sensor) captures RGB-D information. For the sensory perception we use the kinect; for the environment interaction we use the left arm and its gripper.

Our training set was generated autonomously by the robot's perception-action interaction. Several objects were used for dataset generation, highlighting the variance in their perceptual information and in their effects with relation to the action set, e.g., objects with different perceptual descriptions and similar expected effects, different expected effects for similar actions. Learning of affordances, as described in section 3.3 was done online using data instances obtained periodically.

Figure 3.3: Experiment setup. Baxter robotics platform (left). Kinect sensor (middle). Subset of objects of interest (right).

3.3.2.2 Results

In figure 3.4, we present the evolution of the network seen from the point of view of the log-likelihood loss. This results come from the 10 fold cross validation approach over the current set of data. We can see how the expected loss is reduced with the number of interactions. Which means that the learned structure is generalizing better for related scenarios. Although most of the network relations evolve with the number of tries, there are some dependency connections that are confirmed in early stages of the experiment.

Figure 3.5 shows some examples of the relations learned by our approach during the evolution of the experiment. The first relation was learned from the beginning of the experiments. It shows a dependency between the variables representing the perceptual information. The second relation shows a strong casual dependency of the *state of the gripper* (*g_state*) with respect to the actions *open* and *close gripper* (*open_g*, *close_g*). Finally, the third relation shows an example of an affordance relation. It connects the perceptual nodes with the action *lean toward*(*lean_t*) and the effect *poi_obj_mov* which indicates a movement of a point of interest from an object perceived by the robot.

Using probabilistic inference over a set of variables in the learned BN, the robot is able to provide information for effects prediction $P(E|O, A)$, feedback in action selection $P(A|O, E)$ or object recognition given its behavioral description $P(O|A, E)$. For example, when we fix a set of perceptual evidence to define an object, and a desired effect, we can obtain a probability distribution of the available actions. The blue object from figure 3.3, has the highest predicted action probability, for the effect *poi_obj_mov*, of $P(lean\_toward|obj_{blue}, poi\_obj_{blue}\_mov) = 0.1577$ while the green object which is fixed to the table has zero probability for either manipulation action, due to its nature. Blue object has also a high probability for *poke* action. These results are coherent with the two affordances (or lack thereof) on these objects: *poke-ability* and *lean-ability*.

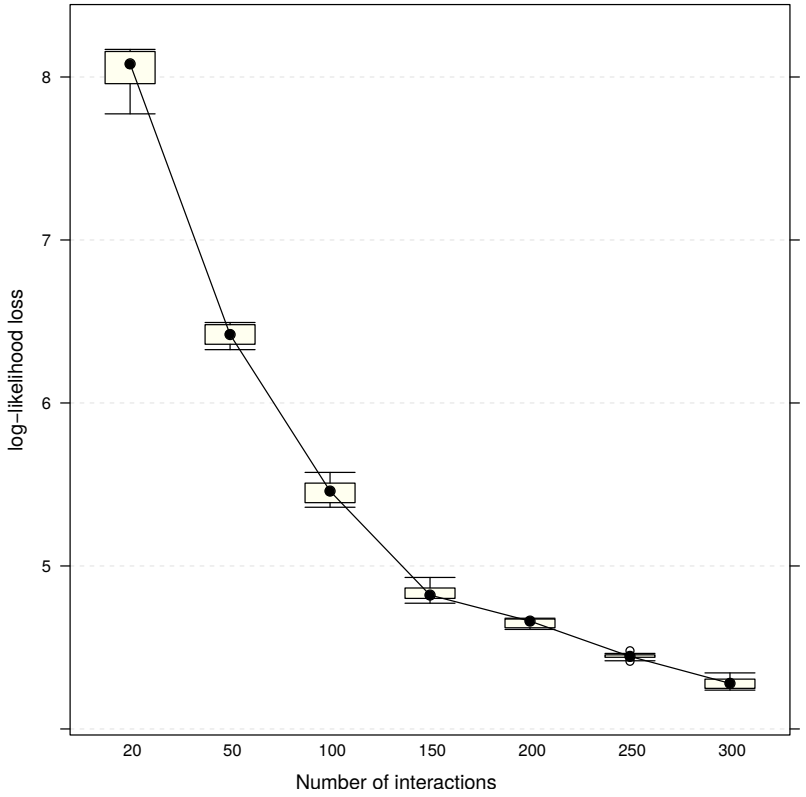


Figure 3.4: 10-fold cross validation evaluation of the learned structure. x-axis represents the number of instances (robot interactions) and y-axis accounts for the log-likelihood loss function.

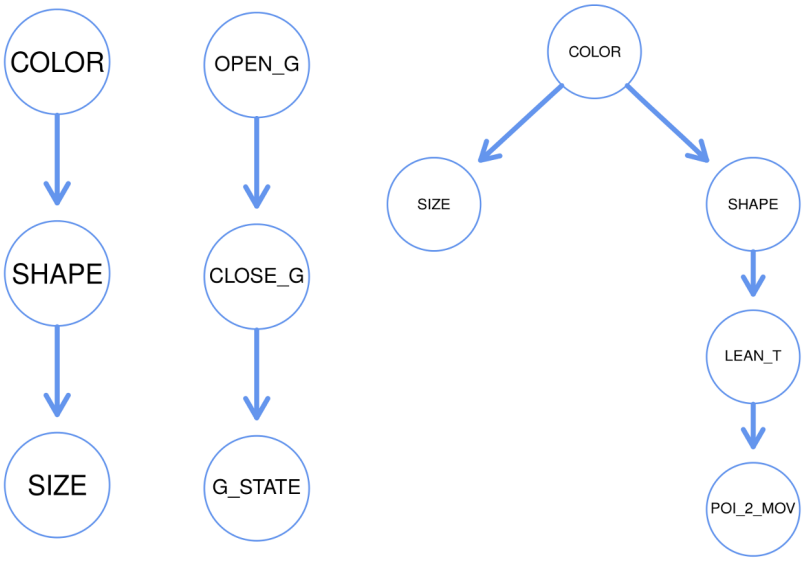


Figure 3.5: Examples of relations learned with the proposed approach. Left relation appeared at 30 interactions, middle relation since the iteration 50, right relation appeared at 150 iterations.

3.3.3 Partial Conclusion

So far in this chapter, we have presented a general architecture for learning probabilistic sensorimotor representations between perception and action from unsupervised interactions. Bayesian framework captures the relation between the three elements of the affordance definition: *effects*, *objects* and *actions*. Our approach does not rely on *a priori* dependencies assumptions between them, although relying on a priori knowledge regarding the descriptors for objects and effects, and motor primitives. It allows the robot to infer the dependencies between the elements while interacting and combining perceptual and proprioceptual information. The learned sensorimotor representation along the Bayesian framework allows the robot's motivational system to make predictions about elements in the environment. Moreover, this inferred information can be used for future planning tasks.

We have shown the connection between the three elements of affordance, which allows to represent the learned knowledge in a fused feature frame. This architecture offers a base for further work as will be presented in the next section 3.4 and chapter 4.

3.4 Composition of Affordances

In this section we try to extend the previously proposed model by considering the concept of composite objects and the corresponding composition of affordances. How does the prediction of the affordance of a composite object differs from the affordance of its components?

3.4.1 Affordances of Composite Objects

Bayesian inference in our discrete BN provides the probability that an affordance α_i is present. However, it does not provide a mechanism to quantify the affordance with regard to the specific environment situations that triggered it.

We believe that by preserving the continuous aspect of the elements in the affordance (3.2.1), we also maintain the necessary information for an affordance quantifying approach, i.e., Bayesian inference over a Gaussian BN (GBN).

Relations in (3.2.1) can be represented as a multivariate normal distribution of continuous random variables, i.e., the affordances elements.

Continuous variables are modeled as linear regressions in a Gaussian BN, where the relevant parameters of each local distribution are the regression coefficients (for each variable *parent*) and the standard deviation of the residuals.

By preserving continuity we also introduce the possibility of predicting the affordance of composite objects, by trying to learn the model that would explain the relation between an object's affordance and it's components affordances.

3.4.1.1 Affordance Definition

In order to fit the purpose of those new experiments, the definition of affordance is slightly altered. The whole formalism is recalled here.

We consider an agent (robot) endowed with a set of innate actions A, and a set of innate feature extractors P, that can be augmented through learning. In addition, O is the set of all the objects in the environment, and E is the set of all the possible observable effects. When the agent applies action $a \in A$ to an entity (object) $o \in O$ in the environment, a salient change (effect) $e \in E$ is generated, we call this acquired relation an affordance Sahin et al. (2007). From the agent's perspective, a resulting affordance is defined as follows:

$$\alpha^{agent} = (e, (o, a)), \text{ for } e \in E, o \in O, \text{ and } a \in A \tag{3.7}$$

more generally, this agent will gradually build a set of affordances *Aff* composed of the affordances α_i:

$$\alpha_i^{agent} = (e_j, (o_k, a_l)), \text{ for } e_j \in E, o_k \in O, \text{ and } a_l \in A \tag{3.8}$$

An object o_k is defined as the set of values for the n innate properties extractors $\rho \in P$:

$$o_k = \rho_1(cluster), \rho_2(cluster), ...\rho_n(cluster) \tag{3.9}$$

where cluster represents the object hypothesis obtained by the visual perception module.

Actions are a set of motor capabilities $A = a_1, ..., a_m$, defined as:

$$a_k(V^*, \gamma, \sigma_{a_k}), \tag{3.10}$$

where V^* is the desired value for the robot control variables V, γ its proprioceptive feedback and σ_{a_k} the particular action parameters.

Effects are a set of salient changes in the world ω detected by robot's innate detectors e:

$$E = e_1(\omega), e_2(\omega), ..., e_q(\omega) \tag{3.11}$$

which means that effects can be related to objects and agents, allowing to detect exteroceptive and proprioceptive changes.

3.4.2 Loss and Preservation of Affordances

The goal of our experiments is to identify a formalism that could infer the affordances of composite objects, based on prior knowledge about the affordances of the primary objects that constitute them. We state that Bayesian networks, through structure learning, can not only discover affordances, but also capture their quantitative aspect, by employing continuous variables in the representation of actions and effects, that are represented as continuous variables.

The experiments will help us demonstrate this. Our experimental procedure is composed of four steps:

- (1) performing a certain action with a set of objects (separate and composite) and observing the effects,
- (2) defining the random variables corresponding to the observed objects, actions, and effects inside the Bayesian network,
- (3) feeding the interaction data to the structure learning algorithm of the Bayesian network, and
- (4) interpreting the structure of the Bayesian network that best fits the recorded data according to calculations.

First, we consider the discovery of affordances. From our experiments, we can interpret the model learned by the discrete Bayesian Network as a qualitative aspect of an affordance, regarding the presence or absence of a relation between the elements of an affordance (e.g. an object is pushable, i.e. it goes a certain distance from its original location). We further attempt to attach a quantitative dimension to the learned affordance by representing its elements as continuous random variables. This allows not only to predict that the affordance is present, but also to infer the parameter values of its elements that influence this affordance (e.g. the effect of the push action on the object is a function of the action's input parameters).

Three experiments are analysed in this section, all related to the inference of affordances of composite objects: (1) affordance acquisition, (2) affordance maintenance, and (3) affordance loss. Since our experiments focused on the composition of objects, we performed them on objects specifically designed for that: toys that can assemble and disassemble. These experiments are detailed in the following sections, and are illustrated in 3.6, which shows the objects employed, as well as their composition method.

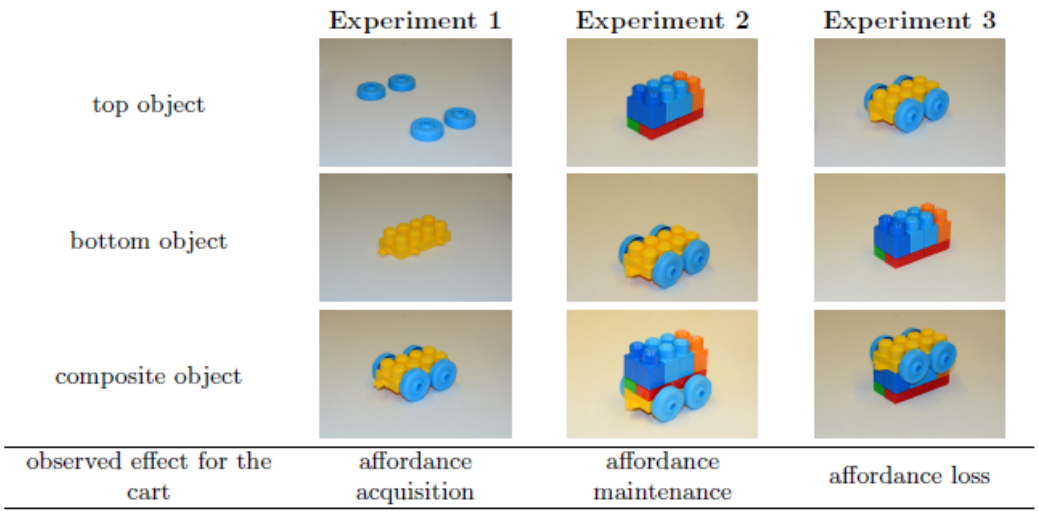


Figure 3.6: Objects used in the experiments, and their composition order.

3.4.2.1 Affordance Acquisition

Following the experiment description from figure 3.6, column Experiment 1, each object is described by two elements: one describing the number of atomic perceptual properties that forms it, and the other the position of the atomic property inside it (top or bottom). These properties allow to represent atomic objects (with only one property) and possible composite objects. In this scenario, we have two atomic perceptual properties **wheel** and **cartFrame** and together they can combine to form a **cart**. The robot performed random interactions with the action a_{poke} and the atomic objects **wheel**, **cartFrame** and with the composite object **cart** (50 interactions with each object). The effect detector developed was based on the distance that an object moves after the action is executed. We use Gaussian random variables to represent the perceptual properties and the distance effect.

We use nominal variables to represent the action undertaken(poke, no action), and the objects employed. The object composition was represented using 2 variables: **objectBottom** and **objectTop**, representing respectively the atomic object at the bottom of the composite object, and the one on top. Figure 3.8 shows the resulting network after the learning process. We can notice that the parameters influencing the distance, over which an object travels after an interaction, are correctly inferred. The action **poke** influences this distance, while the action **noAction** does not. The object at the bottom also influences this distance: **wheels** roll further than the **cartFrame** after poking. The object at the top is also linked to the distance variable, since the distance travelled by the cart (i.e. **wheels** with **cartFrame** on top) differs from the one travelled by the individual **wheels**. Let us use the relationships learned in this example, to infer the affordances of a similar composite object.

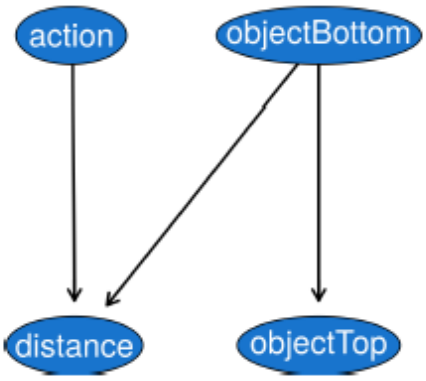


Figure 3.7: The bayesian network obtained after feeding the interaction data with the atomic objects **cart** and **blockLoad**

3.4.2.2 Affordance Maintenance and Loss

The second and third experiments consist in learning the correct structure of the Bayesian network, so as to correctly predict the maintenance or loss of affordances of atomic objects that form the composite object. In this example, we will consider two new objects: the **cart**, and the **blockLoad** that we can put on or under the **cart** (see Experiments 2 and 3 in figure 3.6).

We feed the BN the data obtained in the interactions with these new atomic objects (50 interactions with each object), but not for their composition, and obtain the BN seen in Figure 3.8.

We stated the acquired nature of an affordance in eq. 2, and for this reason, the inferred affordance will be considered an estimation until the robot,by interaction, validates it. If we represent the composite object $obj_{composite} = object_{Bottom} = cart, object_{Top} = blockload$, we can obtain an estimation of its affordance by calculating $P(distance|objectBottom = cart, objectTop = blockload)$ from the learned BN.

In our experiment, the probability distribution for this calculation is similar to $P(distance|objectBottom = cart)$ showing experimentally that the estimated affordance **movable** of the composite object is similar to the affordance of one of its elements.In the BN represented in Figure 4, if the value of the **objectBottom** variable is known, the variables distance and the **objectTop** are conditionally independent. This means that the upper part of a composite object does not influence the distance that this composite object traverses after a **poke** action. This can be interpreted as an affordance loss. On the other hand, the bottom part of a composite object (i.e. **objectBottom**) does impact the distance it traverses after a poke, suggesting that its affordance is maintained.

This is confirmed experimentally: after a **poke**, the atomic objects **cart** and **blockLoad** travel an average distance of 45 centimeters and 9 centimeters, respectively. The composite object with the cart at the bottom travels an average distance

of 28.4 centimeters, while the one with the **blockLoad** at the bottom travels an average distance of only 3.8centimeters.

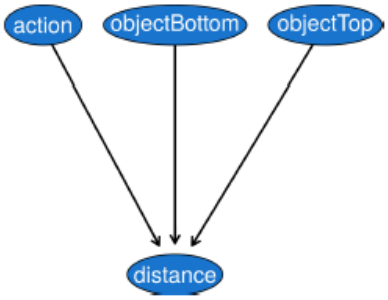


Figure 3.8: Conditional linear Gaussian network obtained after learning process.

3.4.3 Experiments and Results

The goal of our experiments is to infer the relations that exist between affordances, which would allow to refine the definition and formalization of an affordance.

First, we consider the discovery of affordances. From our experiments, we can interpret the model learned by the discrete Bayesian Network as a qualitative aspect of an affordance, regarding the presence or absence of a relation between the elements of an affordance (e.g. an object is *push-able*). We further attempt to attach a quantitative dimension to the learned affordance by representing its elements as continuous random variables. This allows not only to predict that the affordance is present, but also to infer the parameter values of its elements that influence this affordance (e.g. the effect of the *push* action on the object is a function of the action's input parameters).

By decomposing an object offering a specific affordance into its constituent parts, we may wonder what are the affordances of the obtained parts. Answering this question requires us to introduce a mathematical operator, which would be able to estimate the affordances of an object obtained through the decomposition of an object, or through the composition of objects. It is yet unclear if this mathematical operator would apply to the objects and their properties (identifying their affordances as a consequence), or if it would apply to the entire affordance relation $(E, (O, A))$. This opens a whole new domain of inquiry about the relations between affordances.

We designed three experiments in order to test our hypothesis.

Following (3.9), we can define a particular affordance for an object o_i as $\alpha_i = (e_k, (o_i, a_l))$.

Then, we can *decompose* o_i into two *new* objects o_i' and o_i'' by [nullifying] a subset of its property values $\varrho_{null} \in o_i$,

$$\begin{aligned} o_i' &= \{\rho_x | \rho_x \in o_i \setminus \varrho_{null}\} \cup \{\rho_y = null | \rho_y \in \varrho_{null}\}, \\ o_i'' &= \{\rho_x | \rho_x \in \varrho_{null}\} \cup \{\rho_y = null | \rho_y \in o_i \setminus \varrho_{null}\}. \end{aligned} \tag{3.12}$$

Using the learned model from our proposed architecture, we can infer:

$$\alpha'_i = (e_k, (o'_i, a_l)),\ \alpha''_i = (e_k, (o''_i, a_l)). \tag{3.13}$$

If the removal of a property does not influence the affordance of an object ($\alpha'_i \equiv \alpha_i$), then the properties in ϱ_{null} can be considered as non salient for this particular affordance.

In addition, if we can rewrite α_i as:

$$\alpha_i = (e_1, (o'_1 \otimes o''_1, a_1)), \tag{3.14}$$

the computation defined by the operator $\otimes$ suggest the existence of a combination of affordances. Experiments can help to discover the properties of this *composition* operator.

Let us represent the set of salient features from objects o_x and o_y for one of their affordances as $salient_{\alpha_i}(o_x)$ and $salient_{\alpha_j}(o_y)$ respectively, where

$$\alpha_i = (e_{kx}, (o_x, a_{lx})),\ \alpha_j = (e_{ky}, (o_y, a_{ly})). \tag{3.15}$$

If o_x and o_y do not share salient features, $salient_{\alpha_i}(o_x) \cap salient_{\alpha_j}(o_y) = \emptyset$, and $|salient_{\alpha_i}(o_x)| + |salient_{\alpha_j}(o_y)| = n$,

we can construct a new object o_{xy} by *selectively* combining the salient properties of o_x and o_y,

$$o_{xy} = salient_{\alpha_i}(o_x) \cup salient_{\alpha_j}(o_y), \tag{3.16}$$

which by definition should retain affordances α_i and α_j. We can empirically discover the properties of affordances of this new object o_{xy} w.r.t. the properties of o_x and o_y.

Through these experiments (decomposition, composition and selective composition) we will be able to estimate the affordances of combined or de-composed objects, and verify this estimation empirically, shedding light on the nature of these *affordance operators*.

3.5 Conclusion and Discussion

In this chapter we introduced a Bayesian architecture for learning sensorimotor representations by capturing the relations between objects, robot actions, and the generated effects.

We further introduced the concept of primary objects to capture prior knowledge on their affordances.

We later discussed the concept of composite objects, for which we want to identify a relationship between the objects they are composed of, and the way they are assembled in order to automatically infer their affordances. We performed experiments to infer the affordances of composite objects, based on prior knowledge about the affordances of the primary objects that constitute them. The results

from the learned Bayesian network showed information regarding the acquisition, maintenance, and loss of affordances by the employed primary objects, depending on their position in the composite object. The obtained results suggest that it may be possible to define an operator acting on the elements of affordances, which could predict the affordances of new objects, obtained through the combination of known objects.

Although our approach is a statistically based learning technique, it would be interesting to analyse other approaches that could provide statistically similar results or improvement with fewer interactions. It would be interesting to employ algorithms that can identify causal relationships between actions, object features and effects with as few observations as possible (one or two).

However some limitations remains, mostly in the form of the pre-definition of the descriptors for objects ,effects and actions.

In order for the architecture to capture the correct relations we have to provide relevant visual features, relevant effects and relevant actions that we know will produce interesting interactions regarding a given dataset.

- *Actions* are 'relevant' if they produce different effects on the object set. There is virtually no affordances to discover if all objects behave the same. In other word the environment does not afford anything to the agent if all of its actions result in the same sensory perception. Which would be equivalent to no perception at all.

- *Objects descriptors* are 'relevant' if they enable the agent to distinguish objects in regard of effects. If two objects behave differently but have the same descriptors values, the agent cannot learn the relation between descriptors values and behavior. Hence it cannot properly learn the affordance. Furthermore, if descriptors are not correlated to the affordance (e.g. *size* for *graspability*), the agent will learn an affordance specific to the explored environment. For instance: "All graspable objects where red, so i assume red means graspable.". This knowledge will not be generalisable to new environments.

- *Effects detectors* are 'relevant' if they help distinguish the (objects, actions) pairs. The agent must be able to observe dimensions from the effect space that are correlated to what the actions actually produce on objects (e.g. *distance traveled* for *pushing*). For instance measuring color change is irrelevant for grasping behaviors.

What we are trying to emphasis is that there is a circular dependency between actions, effects and objects descriptors. Obviously, as we are considering the discovery of affordances all three parts are linked. But more importantly we argue that a given set of descriptors defines a finite subspace, in which and only in which affordances can be discovered.

So in particular in our experiments we provided a priori information in the form of, (1) two actions *poke* and *push*, (2) selected objects that would react differently

to them (with or without wheels), (3) descriptive features that can differentiate said objects (color, size and shapes), and (4) effect detectors based on movement that distinguish a rolling object from a non-rolling one (distance travelled).

In other words we provided to the agent a set of motor commands (actions) and descriptors (objects features, effect detectors) that were appropriate for the context (objects set), so that we knew that certain affordances could be discovered (fixed, move, roll). It was appropriate in the sense that it enabled and helped the agent discovering those affordances.

The main problem with that approach is that it limits the pertinence of the proposed approach to contexts where the prior information is sufficient. The model will have to be fine-tuned for every environment. Thus the agent will be able to learn affordances in contexts for which it was designed, but will struggle in others if they differ too much. However general or multi-purpose the set of [objects / actions / effects] - descriptors might seem, it will constrain the environments in which the agent can learn meaningful affordances.

Therefore this method is impractical for most of the scenarios that we want autonomous robots to operate in. Scenario where the environment is not, or only partially, known. Or scenario where the environment might change over time. Such scenarios requires the agent to be adaptable to the specifics of each environment, either by adapting current knowledge, or by building new one.

A simple example of such scenarios are kitchen robots. For the robot to cook using the kitchen tools requires it to have several abilities. Of whom the ability to recognize the tools (e.g. knife, pot, spoon) and their functionality (e.g. cutting, heating, scooping), and the ability to manipulate them efficiently (e.g. cut, heat, scoop). However the large variety of shapes, sizes, colors, in which the different tools can exist makes the definition of one-fit-all descriptors hard, if not impossible.

Furthermore from a more theoretical point of view, the pre-definition of descriptors and motor commands somewhat contradicts the very notion of affordance. In this paradigm the agent constructs its own representation of the world based on the complex formed by its embodiment and the environment. However so far in our method the agent only learns the relations between pre-existing *objects*, *effects* and *actions*. Thus the representations built are dependent of (and limited by) a higher level of abstraction.

We argue that to improve the method we need to reduce the need for a priori information. The agent must be able to build its own sets of descriptors according to what the environment could afford it.

In the next chapter we follow this idea and propose a method to remove the limitation on object descriptors (i.e. removing the need for pre-definition). We propose to use pretrained Deep Convolutional Neural Network (CNN) for learning pertinent visual features while simultaneously discovering affordances in order to remove the limitation on object descriptors.

Chapter 4

Ambiguity Reduction and Features Learning

The results and text of this chapter have been published in the following workshop articles.

- Luce-Vayrac, P., and Chatila, R. Learning Relevant Features to Learn Affordances, R:SS IWCMAR 2018
- Luce-Vayrac, P., and Chatila, R. Discovering Affordances Through the Reduction of Ambiguities, IMOL 2017

Contents

4.1 Introduction

4.1.1 The Limits of Predefined Features

Affordances as an approach and framework to autonomous learning is an efficient way of enabling robots to build relevant knowledge of their environments. By linking proprioception and exteroception in the learning process the robot builds representations relative to its own capabilities. And therefore is more able to later adapt or generalize this learning on new situations and environments.

This concept has gained a lot of interest in the robotic community, and has thus been studied in a wide variety of contexts and approaches, as the recent summarizing and analysis work from Zech et al. shows (Zech et al., 2017).

However most authors use predefined features to describe the environment, as we did in chapter 3. We argue that building affordances on predefined features is actually defeating their purpose, by limiting them to a given subspace. Therefore we propose here a method for discovering affordances while simultaneously building visual features.

Affordances are a mean for an agent to represent its environment through what it can do with it, by opposition to solely through the environment's intrinsic properties (Gibson (1977) Gibson (1979)). They are commonly represented as a triplet $(e, (a, o))$ such that the effect e is produced when action a is accomplished on object o (Sahin et al. (2007)). This definition, although generic, is extremely dependent on the definitions of those three components. Different ways of describing objects, effects or actions will dictate the affordances that can be discovered.

To paraphrase the definition from Ruzena Bajcsy in Bajcsy et al. (2018):

> "The learning process depends very much on the assumptions/models of what is innate and what is learned. The theory of the cyber physical system that I have been pursuing predicts that an agent can explore and learn about its environment modulo its available sensors, kinematics and dynamic of its manipulators/end effectors, its degrees of freedom in mobility and exploratory strategies /attribute extractors. It can describe its world with an alphabet of set of perceptual and actionable primitives."

Indeed, a set of given descriptors will generate only a given subspace of the affordance space. In other words only the affordances that can be represented using the given descriptors are discoverable by the agent.

Let us consider for a while what it implies in practice. As we see it, it creates two distinct problems:

- *Non Causal Affordance*: The agent correctly predicts an affordance in its environment, but the properties it is based on are non causal. Example: consider a set of blue cubes and red spheres, a push action, and motion effect detector. In this case the color property of objects is enough to correctly predict rollability, in the sense that color discriminates the dataset in regard of rollability. Therefore in the agent perspective, *red* affords *roll*. However this is only an accidental correlation due to the bias in the environment. It results in a knowledge that is specific to this environment, hence hardly generalisable.

- *Undiscoverable Affordance*: The agent cannot predicts an affordance, because its set of features is not enough to discriminate the environment in regard of that affordance. Example: We keep the same set as before, but this time cubes and spheres are both red and blue. Color is no longer a discriminating feature in regard of the rollability of objects. Thefore the agent cannot learn the affordance.

Therefore we question the relevance of *predefined* e, a and o for autonomous open ended learning systems. It would require a set of features general enough to fit any environment, which we argue is impossible. Hence the need to build new features to enrich this initial set.

Now the question is to know which aspect of an affordance should not be predefined. In other words, which part should be learnt alongside the affordance. As we stated earlier to predefine each aspect leads to limiting the discoverable affordances to a finite subspace. Subspace that may or may not be suited for the context in which the agent will operate, or not suited for the motor and / or perceptive capabilities of the agent. Hence it limits the efficiency of any overlaying learning method to discover affordances. And moreover the ability to discover affordances that are directly related to the agent's own physical embodiment.

In this particular work we focus on removing the constraint on o by using Convolutional Neural Networks (CNNs) to extract non-predefined visual features on objects. Our approach is to consider the discovery of affordances through the reduction of ambiguities :

the agent executes the same action on objects *apparently similar*, but it observes *different effects*, therefore it has to assume that it does not possess the appropriate descriptors to distinguish those objects with respect to this action, hence it decides to learn new descriptors to reduce this ambiguity.

In order to do so we propose a method to detect ambiguities by evaluating the agent capability to predict an object affordance, and to reduce those ambiguities by creating new features using a CNN. This method is then used during a perception / interaction loop where the robot collects proprioceptive data about objects using predefined actions and effects. By predefined effects, we mean that we describe the variation of object's features to observe during interaction. The discretization of effects into classes (used as labels for the CNN training) is done during the loop using an x-means algorithm Pelleg et al. (2000). Thus the whole training process

is self-supervised since the robot acquires itself the data needed for the affordance learning an training of the CNN.

This model is building on previous work Chavez-Garcia et al. (2016b) Chavez-Garcia et al. (2016a), where we considered the acquisition of probabilistic affordances through Bayesian Networks (BN). We use the same global architecture and actions definitions, but we extend effects and we remove the need for predefined objects features.

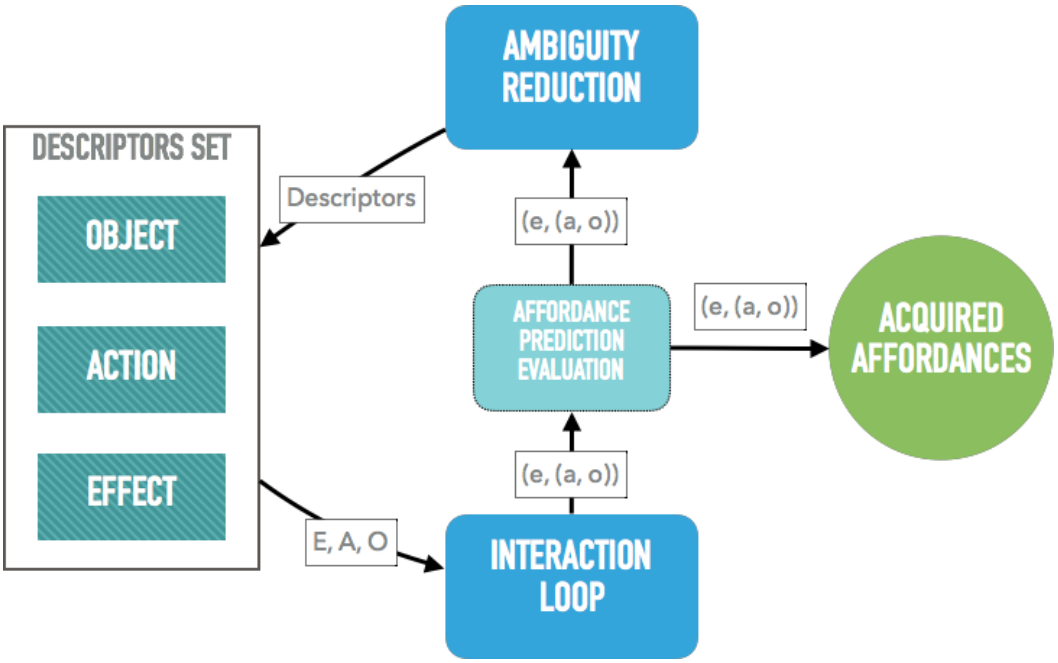


Figure 4.1: Global workflow of the architecture. The interaction loop generates $(e,(a,o))$ triplets using the current *effect*, *action* and *object* descriptors sets (E,A,O). The triplets are evaluated. If an ambiguity is detected a new object descriptor is built.

In summary the main contribution of this work is to enable a robot to simultaneously discover object affordances and build the relevant features, thus removing the need to predefine the later. And furthermore it insures that the agent will perceive its environment using features directly related to its capabilities, through the acquired affordances.

This chapter is organised as follows: in this section 4.1 we briefly summarize related work along with a short introduction to CNN and a presentation of our approach, section 4.2 presents the revised version of the model, section 4.3 presents the interaction workflow, section 4.4 presents the experimental setups and corresponding results, while finally 4.6 draws a partial conclusion regarding the benefits and limits of the proposed approach.

4.1.2 Related Work

Since the introduction of the concept by psychologist J. J. Gibson in 1977 Gibson (1977) affordances have been gradually extended and redefined. Starting from the original concept, an intrinsic properties of the environment, directly perceivable by an agent, to an acquired relation, learned during a sensorimotor exploration. Affordances have legitimately taken an important place in the robotic community as they help ground the perception of an agent in terms of its own capabilities, especially in the developmental approach Lungarella et al. (2003).

Different approaches for learning affordances have been considered. In Ugur et al. (2011) they firstly discretized the effects space, then use it to learn a mapping between the object features, the agent's actions and the categorized effects. In Ugur and Piater (2016) they consider the relation between affordances, more precisely the hierarchy in which they could be organised. And therefore discuss the possibility to acquire complex affordances through simpler ones.

Following the recent development of CNN, and their increasing performance to extract visual features, many authors applied them to the detection of affordances. In Nguyen et al. (2016) and Nguyen et al. (2017) they trained in a supervised manner a network to predict tool affordances, and then used it on a real robotic platform to plan and execute interactions.

The ever increasing need for data samples motivated the work around AfNet Varadarajan and Vincze (2012). Which enables researchers to access and share affordances datasets.

On the contrary, and similar to our approach, in Mahler et al. (2017a) and Mahler et al. (2017b) they limit the need for pre-labeled data by training their model in simulation.

Our method tries to extend that idea by associating pretrained weights and real robots interactions to quickly learn affordances without the need for human supervising.

4.1.2.1 Deep Learning Approach to Ambiguity Reduction

As previously explained we believe that any pre-definition on one or several of the aspects composing an affordance constitute a limitation to the learning capabilities of the agent. Therefore we need to replace the structures used to represent the a priori knowledge of the agent with ones that can be somewhat consider agnostic, and that can learn any relevant features required by the task. In our case we propose to replace the hand designed features used to extract discrete / continuous properties values (color, size, shape) with convolutional neural networks.

CNN have shown tremendous capabilities in the field of pattern recognition and image classification. And are therefore very well suited to be used as visual features.

In the next section 4.2 we will detail the modification applied to our previous model in order to enable the reduction of ambiguity.

4.2 Revised Model and Ambiguity Definition

To extend our model to the newly proposed idea of ambiguity some refinement are required on the architecture and definitions. In this section we firstly present the concept of ambiguity in 4.2.1, in 4.2.2 we detail the changes to our affordance formalism, in 4.2.3 we present the proposed approach to integrate CNN into the model, and finally in 4.2.4 and 4.2.5 we introduce the method to generate training data and use this data to exctract new features.

4.2.1 Ambiguity Definition

One of the question we are trying to address here is the construction of new features during affordances discovery.

As we consider the possibility, and we believe the need, to build features relevant to each action (i.e. affordance), we therefore require a method to evaluate the training, in order to identify and detect the states which would require such new features to be built. For this we propose the concept of *ambiguity*.

We define as an *ambiguity* the following learning states of the agent, that explicit the need to learn new features to properly describe the environment.

- (1) When the agent cannot *learn* to predict the result of an action (an affordance) on a given environment (set of objects). Therefore it has to assume that its current set of features is not rich enough to discriminate the environment in regard of this action (figure 4.2).
- (2) When the agent cannot *generalise* an affordance to a new environment. Therefore it has to assume that its set of features, although sufficient to discriminate the training environment, does not contain the features relevant for this affordance, as it cannot correctly predict the behavior of new (unknown) objects (figure 4.3).

More formally:

$$\mathscr{P}_{a_i}(P_e(a_i, o_j) = G_e(a_i, o_j) | \forall o_j \in O) \leq T \tag{4.1}$$

With $P_e(a_i, o_j)$ the predicted *effect* of *action* a_i over *object* o_j, $G_e(a_i, o_j)$ the observed effect, O a set of objects, and T an arbitrary threshold. O being the training set for definition (1), and a set of unseen objects for definition (2).

Although similar (both ambiguity express the uncertainty in the model) the 2 definitions serve different purposes which are further detailed below.

Definition (1) illustrate that the set of features used does not enable the agent to properly distinguish the objects composing the current environment. Either due to a small number of features or similar objects, the features values of those objects overlap. Objects appear identical to the agent but behave differently when acted upon. Therefore when exploring its environment the agent will have to cope with contradictory affordances triplets $(e, (a, o))$. Which will eventually translate to

poor affordance prediction. For instance if considering the features set size, colour, texture, a cube and a sphere could be identical and yet behave differently when pushed.

The environment is ambiguous because it is insufficiently described (under-fitting or under-segmentation), thus objects descriptions are overlapping (highly similar to each others).

Definition (2) however illustrate the lack of relevant features regarding the current action. The agent can learn to predict the result on a training set, but cannot generalise this learning on a new dataset. Which implies that the features used for the prediction are only relevant to the training set, not to the action.

The environment is ambiguous because it is abundantly described (over-fitting or over-segmentation), but none of the features are actually pertinent.

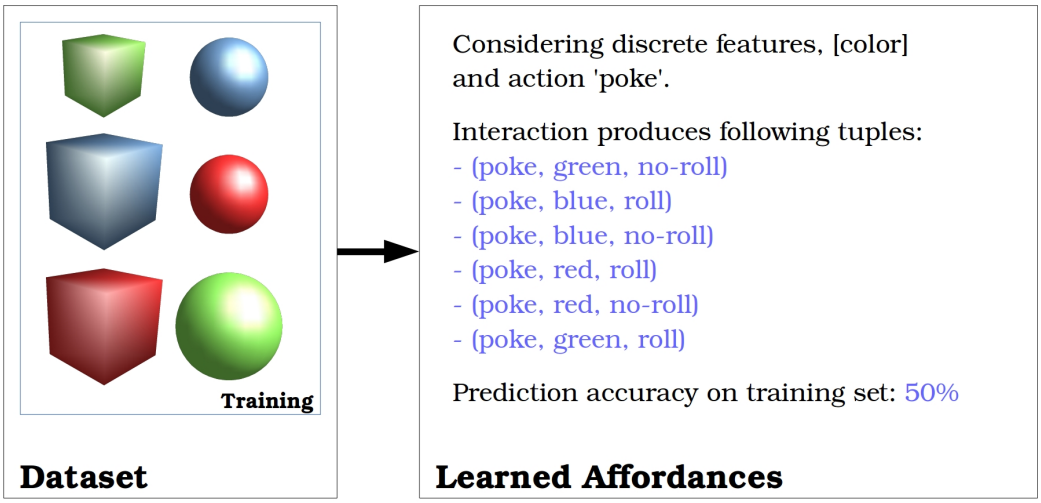


Figure 4.2: Ambiguity 1 Schematic.

Whenever a state of ambiguity is detected, which indicates the lack of relevant features regarding an action, the construction of a new feature is triggered. Therefore over the course of learning, the initial feature set will grow as new features are built, which will reduce the probability of objects having too similar descriptions. In the long term type (1) ambiguity should become less frequent in favour of type (2) ambiguity.

Overall, through this process, we aim to improve the learning of affordances by *linking* it to the construction of *pertinent* features, and *removing* or *limiting* the need for predefined ones.

4.2.2 Affordance Revision

As in chapter 3 we represent affordances using the formalism from Sahin et al. (2007), where an affordance is represented as a $(e, (a, o))$ triplet, encapsulating the

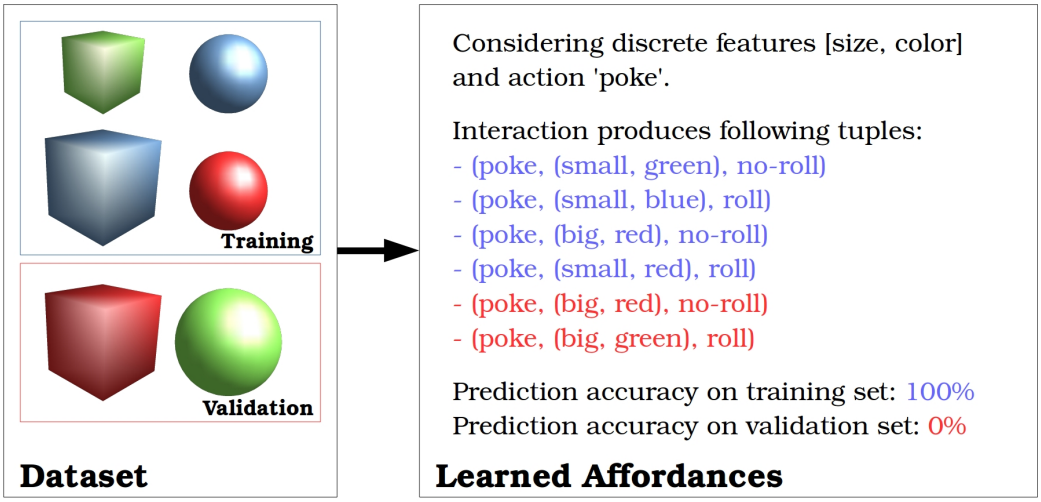


Figure 4.3: Ambiguity 2 Schematic.

relation between an *effect*, an *action* and an *object*. However we know want to learn the features to describe objects simultaneously to the learning affordances. Thus the exact implementation changes, these section presents those alterations.

4.2.2.1 Object

Strictly speaking an *object* in the affordance formalism is the target and/or context for an *action*. And it can therefore encapsulate a broad range of definitions. From physical entities (a door handle, the door itself or even the whole room), to more abstract concepts (such as the relation between two entities).

In our work we focus on discovering affordances regarding small physical entities, with which the robot can interact. Therefore we define an *object* as a discrete area of the environment, upon which we extract local features. Such that for a given set of features F, $o = \{f_i(\omega) | f_i \in F\}$ with f_i the value of feature i for object's o raw sensory input ω.

As in Chavez-Garcia et al. (2016b) the actual localisation and perception of objects is achieved with a RGB-D camera. The objects hypotheses are generated from the perceived point cloud in three steps:

- (1) Remove the background using planar extraction,
- (2) Use an over-segmenting algorithm to create *supervoxels* on the remaining cloud Papon et al. (2013), and
- (3) Supervoxels are clustered together using meanshift Comaniciu and Meer (2002) and a locally convex criterion Christoph Stein et al. (2014).

The resulting points clouds and corresponding 2d patches constitute the raw perception ω of objects on which the features are computed (see 3.1).

Our focus is to create features at the object level, features that would specifically explain the behaviour of those objects in regard of the action capabilities of the agent. And enable the agent to better apprehend its environment.

Therefore the previous simplifications (pre-existing concept of objects, segmentation capability and simple setups with no occlusion) do not constitute limitations for our method.

4.2.2.2 Action

The definition of action remains identical to the previous method presented in chapter 3 section 3.2.1.2. An action is a parameterized sequence of primitive motor control commands (such as move the arm forward, open/close the gripper, rotate joint, etc).

We acknowledge the limitation inherent with such a fixed definition, however we consider this not to be a limit for the results of the method. In this work we mainly focus on constructing the features relevant for an action. Therefore we believe it is enough for the agent to have the knowledge that an effect is possible, rather than exactly know 'how' to produce the effect.

For the work described here, we predefined two action primitives:

1. *"Pushing"*: approach a target with the end-effector, then move forward to push target.
2. *"Poking"*: approach a target with the end-effector, then rotate wrist to poke target.

Each of the previous actions are parameterized by the target position for the interaction.

4.2.2.3 Effect

In order for the agent to be able to perceive the results or "effects" of its actions, it has to perceive the changes or variations in its environment. Similarly to the object representation, descriptors are used to transform raw sensory input into lower dimension features. We define an *effect* as a function over the variation of said features resulting from an action.

Formally, for $d \in D$ the ensemble of descriptors, $d(t)$ and $d(t+1)$ the values of this descriptor at time t and $t+1$, we define an effect as a function $e(d(t), d(t+1))$.

Such that, for example, with d an object's position, we could define:

- The effect "distance travelled by object" as:

$$e(d(t), d(t+1)) = \|d(t) - d(t+1)\|$$

- The effect "object has moved" as:

$$e(d(t), d(t+1)) = d(t) \neq d(t+1)$$

4.2.3 Sensory Perception

According to the revision of the model and definitions, the sensory perception approach presented in 3.2.2 as been slightly adapted.

Planar extraction, objects segmentation and Effect detectors are the same, see 3.2.2 for details.

What differs is the process of feature extraction. So far all features where extracted from the point cloud of each object. However the features set now also include CNN that require 2d images as input. Thus to compute the features value of an object the agent must also gather 2d images of it.

Furthermore the number of features to extract on objects is no longer fixed, but it is growing alongside the learning of affordances. More precisely one new feature is added for each ambiguity reduction. To handle the overgrowth of the feature set, a selection method is required, to choose which feature to use in a specific context. This study is out of the spectrum of this particular chapter, but will be discussed in the conclusion 4.6.

4.2.4 Effect Clustering

As previously stated, the learning of new features revolves around the detection of an ambiguity.

To detect such ambiguities an agent must be able to discriminate objects in regards of the effects that its actions produce upon them, rather than by the intrinsic properties of those objects. In other words two objects o_1 and o_2 are identified as different because it exists an action a, such that (a, o_1, e_1) and (a, o_2, e_2), with $e_1 \neq e_2$. Regardless of o_1 and o_2 descriptors values, the agent knows that those objects differs, as they behave differently in regard of action a.

Therefore the agent needs to be able to compare effects, and differentiate them qualitatively.

In order to do so without providing too much a priori knowledge to the agent, we chose to use an unsupervised clustering algorithm (X-means Pelleg et al. (2000)) to discretize continuous *effects* into *classes of effects*.

So that if we consider an action a and a set of n objects O, interacting with each objects will produce a set of tuples $\{(a, o_i, e_i) | \forall o_i \in O\}$. We can then extract the n e_i effects and create a cluster model $\mathscr{C}_a$ using X-means, and we define $c_{a,o_i} = \mathscr{C}_a(e_i)$ as the class of effect of e_i regarding action a.

4.2.5 Features Extraction

In our approach when an ambiguity is detected, we assumes that the reason for this ambiguity lies in the lack of descriptive features. Or more precisely of adequate features. The environment is ambiguous from the agent perspective because the agent lacks the features pertinent for the complex 'motor capabilities / environment'. In other words, the agent does not possess the features that are relevant to the

affordances that it could be discovered considering the motor capabilities of the robot and what the environment offers.

Hence to reduce the ambiguity the agent must construct new features relevant to what the environment can afford. In order to keep the autonomous aspect the new feature must be constructed by the agent, using only its available sensors and no a priori information about the objects or affordance. To do so we propose the use of a deep convolutional neural network (CNN).

For each ambiguity a new CNN network is instantiated, and trained to predict the objects' classes of effects (i.e. learn to predict the result of the action that triggered the ambiguity on objects). Thus the ambiguity is reduced by the construction of new features in the convolutional layers, and prediction of the affordance in the dense layers.

We take advantage of three main qualities of CNN to fit our method:

1. High performance to extract features on visual data: CNN have established the current state of the art in image recognition, mainly due to the capability of convolutional layers for learning visual features (Krizhevsky et al. (2012)).

2. Generic purpose nature: As the literature shows, CNN have been applied successfully to very diverse contexts (image classification Rawat and Wang (2017), natural language processing Kim (2014), etc).

3. Compatibility with transfer learning: An other important advantage of CNN is the possibility to use weights trained on a task A to initialize a network for another task B, reducing substantially the training time (Yosinski et al. (2014)).

Therefore if we consider these advantages in regard of our method we can expect that:

1. A CNN will be able to extract the pertinent visual features relative to an action, using only 2d images.

2. We will not be required to design an specific architecture for each task, on the contrary a single architecture should fit many tasks.

3. We will be able to use a pretrained CNN, which will shortens the training time, and make the method more resilient to over-fitting (as it often occurs with small datasets).

Our learning workflow is based on the collection of $(e, (a, o))$ triplet through the interaction of the robot with it's environment. The action a is implicit (each action corresponds to a different network). $(o_i,\ c_{a,o_i})$ pairs, with $o_i \in O$, and c_{a,o_i} the label of this object such that $c_{a,o_i} = \mathscr{C}_a(e_i)$. More precisely, the training dataset is constituted of 2d images from the objects, labeled

Each iteration consists of the following steps:

1. An action a_j and object hypothesis o_k are selected.

2. The robot gather a set $\mathscr{I}$ of 2d images of the object.
3. The action is realised, thus generating an $(e_i, (a_j, o_k))$ tuple.
4. The effect e_i is used to update the classes of effects $\mathscr{C}_{a_j}$ for action a_j and the corresponding class $c = \mathscr{C}_{a_j}(e_i)$ is returned.
5. Images in $\mathscr{I}$ are labelled with c and added to the samples dataset $\mathscr{D}$.
6. The network is then trained with $\mathscr{D}$.

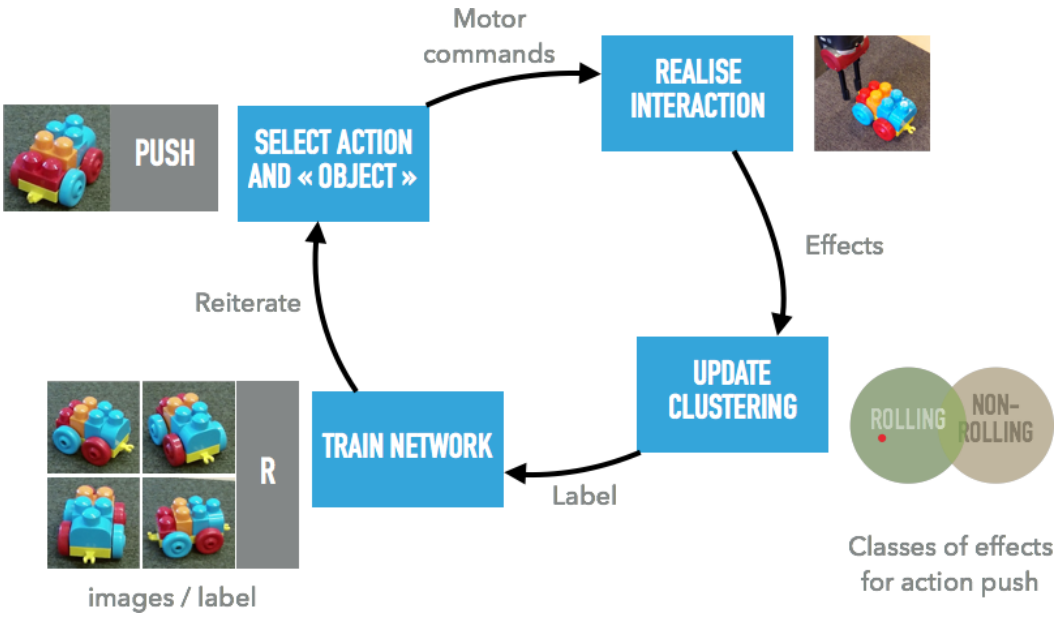


Figure 4.4: Workflow for the resolution of ambiguities through self supervised interaction

We tested several different architectures whose structures can be found in the supplementary material. All networks are based on the same overall structure derivative from a VGG16 network Simonyan and Zisserman (2014), the details of the networks can be found in the next section 4.2.5.1

As a mean to limit the need for this pretraining, only the top two convolutional and dense layers of each networks are specific to each action, the rest is shared. Therefore the learning of simple affordances can initialize the network and bootstrap the learning of more complex ones. See 4.5 for details. In an ideal context, the agent should be able to start building knowledge from very simple interactions. And with each reduced ambiguity a new feature is added to its descriptors set. Thus allowing the agent to segment the environment in new ways, which in turns create new opportunity for interactions.

However our limited number of experimental setups does not allow us a completely agnostic approach. We have to assume that some knowledge has already

been acquired. Knowledge that is either the result of previous interactions. Or knowledge that could have been acquired during an anterior developmental stage.

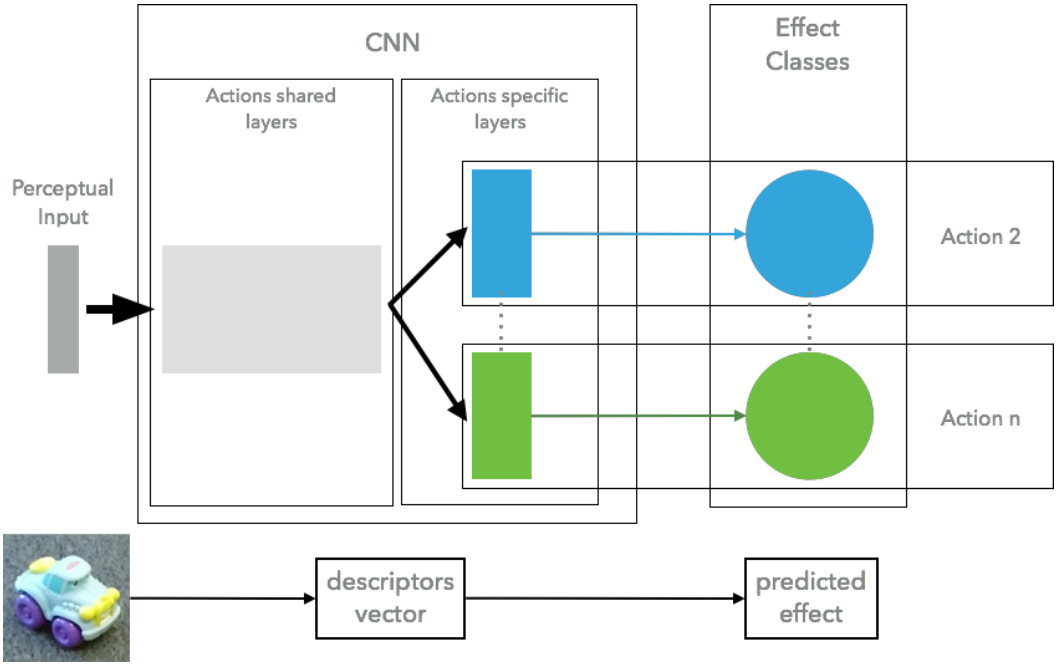


Figure 4.5: Global architecture of the network used to predict objects affordances. From left to right, the 2d image of an object is processed into effects predictions. Firstly through shared convolutional layers. Secondly through actions specific convolutional and fully connected layers. Finally the resulting labels are the predicted effects of each action for this object.

4.2.5.1 CNN Architecture

In this section we present the kind of architectures that we used for learning new visual features. We will describe the types of network and their exact composition, and the reason behind those choices.

As explained before, the main reason we chose to use Convolutional Neural Networks is for their potency to extract regularities from a raw input (in our case, 2d images). Which is important for our task, as we want to build features from images without providing to the agent any insight of what those features might be.

However our approach brings two main constraints:

1. The architecture cannot be task specific, as it would contradict its initial purpose, which is to enable features learning in unforeseen situations.

2. The architecture must be data-efficient, as the agent must be able to learn a new feature in a reasonable set up (i.e. a small number of objects).

Solving (1) is quite straightforward as CNN have proven to be multipurpose, as many research have shown Canziani et al. (2016) LeCun et al. (2015) Krizhevsky et al. (2012). CNN architectures have been applied to many different problems, from image classification to object segmentation, with little to no variation to the inner layers composition. Furthermore our task is essentially a task of classification, thus a simple architecture similar to a VGG16 is sufficient.

Solving (2) however requires several workarounds. A high dropout rate certainly helps Hinton et al. (2012). But that would not be sufficient to train from scratch a deep network. Therefore we have to use pretraining in order to initialize the network. More precisely we use transfer learning. A network is trained on an unrelated task beforehand, and the weights of the convolutional layers are transferred to the actual task network. Lastly we use data-augmentation to increase the size of our dataset.

With that in mind we chose to test 4 different architectures (and variants of them), deriving loosely from a VGG16 structure (several convolutional blocks followed by a few fully connected layers). The different configurations are composed of a succession of convolutional blocks, ended by a fully connected block. The convolutional blocks are constituted of 2 to 3 convolutional layers (respectively the Nconv and Nconv3 variants), followed by a max pooling and a dropout layer. The final fully connected block is constituted of n pair of (fully connected layer, dropout layer), ended by a final fully connected. See figure 4.6 for a visualisation of the base structure of those networks.

- **VGG16** (see figure A.1b, Simonyan and Zisserman (2014))
- **10conv3**, 5 conv blocks (see figure A.6a)
- **8conv3**, 4 conv blocks (see figure A.4b)
- **6conv3**, 3 conv blocks (see figure A.2b)

We chose to use a VGG16 architecture because of its performance and relative simplicity. We designed the **10conv3**, **8conv3** and **6conv3** derivatives to test shallower architectures.

4.2.5.2 Pretraining and Transfer Learning

As introduced earlier, the pretraining of the network is one of the possible workaround to compensate for the small number of objects. The main problem which arises from this limitation is the possible over-fitting of the dataset. Due to the small number of objects the network would most likely learn to recognize them separately, rather than learn what the objects have in common that explains their behavior. In other words, the network would construct features in order to distinguish the objects from one another instead of constructing global features shared between them.

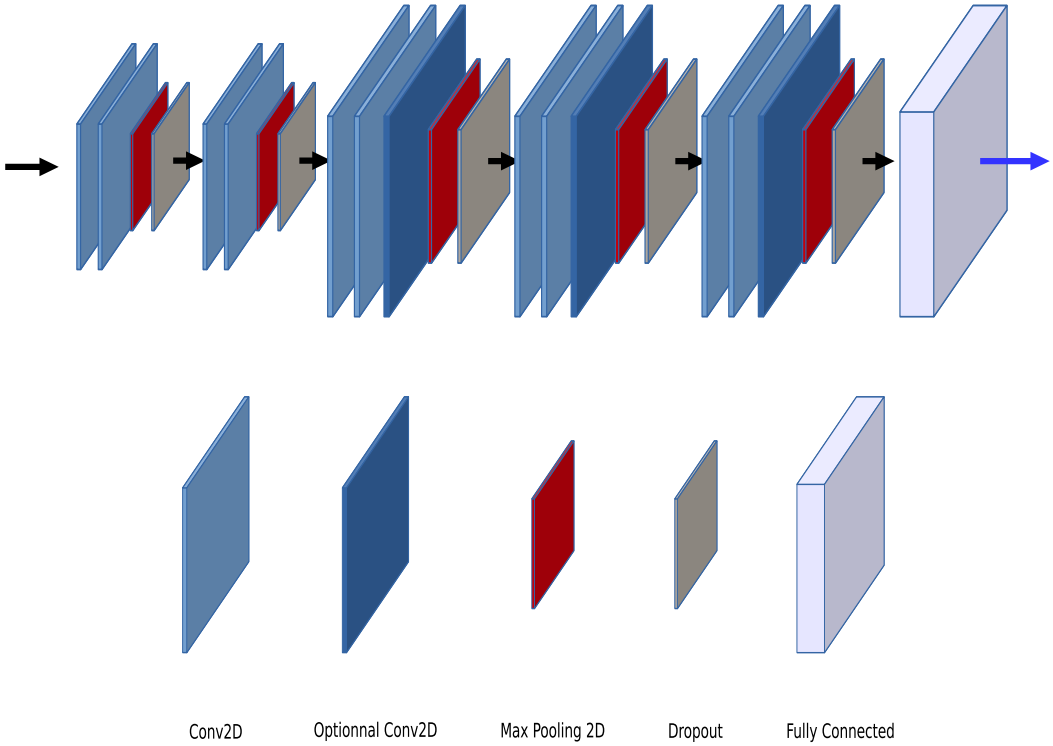


Figure 4.6: Modular structure of the CNNs used to learn new features.

The consequence being that the network will reach peak performance on the training data, but will not be able to generalize the learned model to new objects. Which means that the constructed features would not be relevant to the affordance.

One possible way of overcoming this limitation is to use pretrained weights. The central idea is to profit from the relative general-purpose quality of bottom layers features in a convolutional network. The first layers of a CNN are usually very similar regardless of the training task. They contains features sensitive to simple gradients, later combined to form basic shapes.

Therefore in our method we propose to initialise the convolutional layers of the model with pretrained weights, with two main benefits. Firstly it improves the network's generalization capability by setting the bottom convolutional layers to values that are independent of the objects that will be considered for this task. Secondly it reduces the time required for the network to converge. Only the top layers will have to be fully trained, the convolutional layers will only be fine-tuned to fit the new task.

One could argue that the pretraining of the network is actually defeating the purpose of this method. pretraining can be considered as giving a priori information to the agent, and furthermore this information is used to build low levels features in the convolutional layers of the network. Thus it influences (and more importantly possibly constrains) the construction of any new features.

To answer, we have to consider that low levels features inside CNNs are very

similar no matter the classification task. Therefore the risk of constraint is minimal.

To pretrain our networks, we built a custom dataset of images. Composed of 22 classes of objects (flowers, tools, basic shapes), 14000 images in the training set, 4000 in the validation set.

4.2.5.3 Network Training

As presented above, we propose to reduce the ambiguity by creating new features relevant to the task. To do so for each action a specific network is instantiated and is trained to predict the result of that action based on 2D images of objects.

More precisely, to train the network we use the labeled data that is produced during the robot interaction. During each interaction a series of images of the object is captured, then the object (therefore its images) is labeled by the action's resulting effect. The training dataset is composed of all labeled images from all the interactions with objects.

The network is then trained using a classical supervised learning and stochastic gradient descent. To cope with the small number of interactions (objects images), and as presented above, the network is initialised with pretrained weigths, and we apply data augmentation to the images from the dataset.

Data augmentation consists in applying small variations on the images from the dataset to generate new ones, slightly different, that can be used to train the model, while reducing the risk of over-fitting to the initial dataset [useful?]. In our case to augment the dataset we use the following functions: horizontal shifting, rotation, vertical flip, horizontal flip.

The transfer learning differs depending on the type of network. The VGG16 versions are initialized with weights from ImageNet Deng et al. (2009) and the smaller versions (10conv3, 8conv3, 6conv3) are initialized from models pretrained on a custom task of classification. Both are detailed further in 4.2.5.2.

The specific conditions of the training (learning rates, optimizers, etc) will be discussed in the experiments and results section 4.4.

4.2.5.4 Network Exploitation

In this section we present how networks can be used once they have been trained and therefore theoretically contain newly learned features.

As explained in 4.2.5.3, each network is specific to an action a (at least partially). Each network is trained to classify images of objects based on the observed effects that this action a produced upon said objects.

Therefore the result of this training is a network which can be used to predict an object's class of effect regarding action a. In other word when given an image from object o the network predicts the effect e from relation $(e, (a, o))$.

This network can then be used in two ways. Firstly it can be regarded as a feature in itself. If we consider a simple example, an action 'push' and an movement

detector.

From the point of view of the robot, if the model has been trained to distinguish rollable objects from non-rollable ones, then the network is a feature classifying objects in terms of 'rollable-ness'. Much like we perceive objects as 'pushable' or 'graspable'.

Secondly the content of the convolutional layers and especially the higher layers can be regarded as lower level features specific to the learned affordance. In order to correctly classify images, the convolutional layers were forced to create features that would distinguish the objects in regard of their respective classes of effects. Hence features that are relevant to the learned affordance.

To summarize a high level feature is created in the last layer, and lower level features are created in the convolutional layers.

4.3 Interaction Workflow

In this section we describe the interaction workflow. The sequence of behaviors that the agent executes in order to learn affordances and/or reduce ambiguity.

To summarize, the agent will:

1. Observe the environment (sensory data) 4.3.1
2. Execute an action (sensorimotor data) 4.3.2
3. Interpret effect of action 4.3.3
4. Update model of affordances and/or reduce ambiguity 4.3.4

4.3.1 Object Image Acquisition

In order to train a CNN, we need to generate a dataset of images. We do so by capturing a set of images from each objects that the robot will interact with.

Much like a human or an animal do when confronted with a new or interesting object. It turns around to see the object from different point of views, go closer or further to see the details or the context of the object, or pick it up if possible to observe otherwise inaccessible parts.

In more general terms, the agent uses all its available sensors and motor capabilities to gather additional sensory information about the object. That raw information can then be evaluated through the features already available to the agent, or as in the proposed method here, used to construct new ones.

In our approach we propose to construct new features using the visual appearance of objects (through CNN). Therefore the data that the robot will gather will only be 2d images.

In our experimental setup the Baxter robot uses a wrist mounted 2d camera to capture images from various orientations around the objects. Thus we emulate a

pick and observe behavior without requiring the robot to move. And most importantly we avoid to actually have to pick the object, which remains a complex task in robotics.

The object is detected using the method described in 3.2.2.1, then the robot captures images of it using the following steps:

1. Locate object's centroid c.
2. Compute the coordinates of a 25cm diameter semi-sphere, centered on the object.
3. Select a position p on that semi-sphere,
4. Move the camera to this position, and align it with the axis $[c, p]$.
5. Capture an image.
6. Iterate through (3), (4) and (5) for n different positions on the semi-sphere.

4.3.2 Action Execution

To physically interact with the environment we provide the robot with predefined actions primitives. The actions consists in a sequence of motor controls commands. Approach the target (selected object). Interact with the object (for instance wrist rotation for *poke* primitive). Then retract to the initial position.

4.3.3 Effect Detection and Clustering

As presented in 4.2.2.3 in our approach we use predefined effect detectors. That means that we provide for the robot the dimension in its perceptive domain that it needs to observe. In other words, we provide to the robot the part of the environment that it needs to take into consideration for computing effects.

4.3.4 Ambiguity Detection and Reduction

The main incentive of our work is in the ambiguity detection and reduction. This concept of ambiguity is the drive of our approach, by enabling an agent to recognize that its set of features is too limited to effectively describe the environment in terms of its own capabilities.

As described earlier we identified two main cases of ambiguity 4.2.1:

- When the agent cannot properly learn to predict an affordance on a given set of objects,
- Or when this predictive model cannot be generalized to new objects.

Those definitions corresponds to a static vision of the agent-environment complex. Given a certain set of objects, actions and effects, is the environment ambiguous to the agent? Therefore using the proposed method an agent would eventually reduce all ambiguities by creating enough relevant features.

In a dynamic case an ambiguity can appear with a change in the environment (i.e. new context), in the motor capabilities of the agent (new action) or in the perceptive capabilities (for instance effect perception).

Evidently when the environment changes new possibilities of interactions are introduced. Among those new possibilities some objects may behave in a way that was not seen before or that contradicts the predictive model. Therefore one or more actions that was previously non-ambiguous can become so, and requires the construction of new features to explain the behavior of those new objects.

Similarly, when a new action is introduced it creates a new way to segment the environment. Each action being in itself a way of distinguishing objects based on the effects that it will produce upon them. Therefore new distinctions between objects might appear that might not be explained by the current set of features, hence resulting in an ambiguity.

Lastly, adding a new perceptive ability to the agent means to change the way it describes the environment. Let's put aside the consideration on new sensors as we consider the agent to possess all knowledge about it's motor and sensors from the start, and only consider new effect detectors. Each effect detector allows the agent to evaluate the result of its actions along certain dimensions of the effect space. Therefore a new effect detector corresponds to a new combination of said dimensions, i.e. a new way of measuring the result of an action. Consequently objects or actions that were similar regarding effects so far may differ now.

To summarize, any changes to one or more of the main aspects of affordances (namely *effects, actions, objects*) should be considered as potentially introducing new ambiguities.

For all the above reasons, we need to define a method to evaluate that a model is ambiguous, and a set of triggers to decide when to evaluate the model.

For the evaluation, we propose to use the prediction accuracy of the model. If an agent is not capable to correctly predict the result of an action after training, then it has to assume that its current descriptive features are not relevant for the task.

To trigger an evaluation we need to consider the cases where the prediction accuracy might change, and thus possibly go under the accuracy threshold. We identified two cases, (1) if the model is modified or (2) if the evaluation dataset changes.

The model is only modified during training, and the evaluation dataset changes if new tuples (data, labels) are added/removed. In our case both only occurs during an interaction. Therefore we chose to trigger an evaluation at the end of each interaction cycles.

In the case where the evaluation does not meet the required threshold, an ambiguity is raised.

4.4 Experiments and Results

In this section we present the experimental approaches and setups that we used to test our method, along with the results. In 4.4.2 we present the global setup we used, and the motivation behind the experiments. Then in 4.4.3 and 4.4.4 we present two experiments conducted on a real robot. While in 4.4.5 we present the results of a more thorough investigation of the limits of our approach to features extraction.

4.4.1 Data Collection

The principle of IP methods Bohg et al. (2017) applied to affordances learning is to enable an agent to gather sensorimotor information about its environment. In our case the information is collected under the $(e, (a, o))$ formalism.

The action a is the one used for interaction.

The effect e and object o are captured through the kinect camera (point cloud), and baxter wrist mounted camera (2d images). We use the point clouds to compute the predefined features values of objects (size, color) as well as tracking the object to compute the effect. We use the 2d images to compute the new features values, as well as learning those features in the first place during the ambiguity reduction process.

The collection of images is done through a predefined behavior. The wrist mounted camera of the baxter is used to gather images of the object from various position around it. The effect information is computed using the predefined effect detectors and the corresponding sensors (e.g. RGBD camera for movement).

See section 4.4.1 and section 4.3.3 for more details.

4.4.2 Experimental Setups

To test our approach we used a similar setup to the one employed in chapter 3. Namely a Baxter robot and a set of RGB-D cameras (Kinect V2) on a tabletop scenario where the robot has to interact with various objects in an uncluttered environment. The objects are presented sequentially, the robot executes the interaction protocol (see figure 4.4), then an other object is presented.

The goal of these experiments is triple. Test the ambiguity criterion to detect the situations of ambiguities. Test the ability to reduce an ambiguity by learning new features in a pretrained CNN. And finally validate the newly built features as relevant for the agent by correctly predicting the affordance.

The robot uses prior information:

- An initial feature set to describe objects (color, size, position)
- Action primitives (push, poke, observe)
- Effect detectors (movement)

The goal of those experiments is to place the agent in a context where the initial descriptor set is insufficient. So that ambiguities will arise, thus the agent will have to construct new features, relevant to the task, in order to reduce the ambiguities.

We tested two different scenario:

- $A = push$, objects are either movable / unmovable (fixed to the table).
- $A = poke$, objects are either rollable (wheeled or spherical) / unrollable.

We use the MoveIt algorithms and library to plan the movements of the robot (Sucan et al. (2012), Sucan and Chitta (2015)).

In both scenario the idea is to construct sets of objects that differs in physical properties (and therefore effects) but for which the corresponding visual features is non-trivial. By non-trivial, we mean that it would be hard both for a human to engineer a feature detector and for a CNN to learn it. The goal is to show the interest of proposed method precisely in cases where we need the agent to autonomously build features and where said features are not oblivious.

4.4.3 Experiment 1: Pushable objects

4.4.3.1 Setup

In this first experiment the agent has to learn the *pushability* affordance of a set of objects.

Objects Dataset The set is constituted of 46 objects. 28 of them are 'textured', the remaining 18 are 'smooth' (see figure 4.7 for examples objects from the dataset). Objects are otherwise similar from the agent perspective (i.e. considering the agent's initial set of features), similar colors, similar sizes.

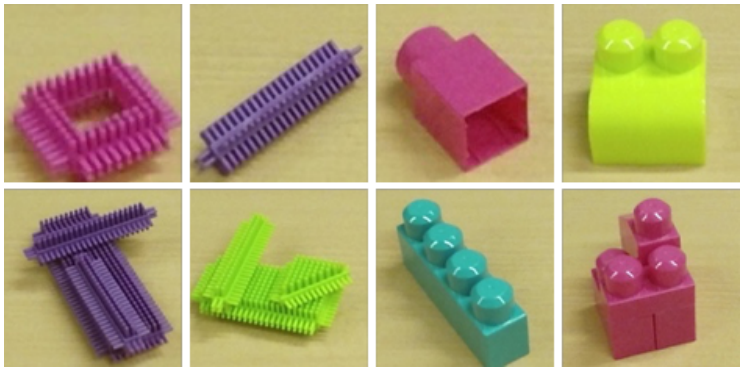


Figure 4.7: Examples of objects from experiment 1, movable / non-movable dataset. On the left are 4 'textured' objects, on the right 4 'smooth' objects.

The 'smooth' objects are made unmovable (bricks are hollow, and fixed to the table using a screw), thus creating a difference in physical property between the two sets. The 'smooth' ones are unmovable, the 'textured' ones are movable.

Push Primitive To interact with the objects the agent is given an action primitive *push*. The *push* primitive consists in the following sequence of motor commands:

1. **Approach**: move the end-effector 5 cm away from the target object. The end-effector is placed perpendicular to the axis between the object's center and the robot base.

2. **Push**: move the end-effector forward in a straight line toward target for 15 cm.

3. **Retract**: return end-effector to the starting position.

Movement Detector To detect the results of the *push* primitive we use a predefined movement detector. The target object is tracked during the experiment, its centroid position is registered before and after the interaction. We then compute the euclidean distance between those two points as the object's distance traveled.

Initial Features The initial features set contains a color and size descriptors (see 3.2.2.2). Considering this initial set of features, objects will appear similar to the robot, while still behaving differently. Hence it will result in an ambiguity that the robot must reduce by constructing a feature relevant to the push affordance, in this case the 'texture' property.

CNN Architectures We selected 4 different architectures, **6conv3**, **8conv3**, **10conv3** and **VGG16**, and their variants. Which respectively contains 7, 10, 13 and 13 convolutional layers. The exact composition of all 4 architectures is detailed below in the annexes A.2b, A.4b, A.6a and A.1a.

4.4.3.2 Results

For this experiment each iteration consists in the robot interacting once with all objects. The objects are presented in a pseudo random manner (i.e. selected by the operator). In our current approach the agent does not takes into account the knowledge already accumulated to select the next target for interaction. Therefore we can execute all interactions, and then train the model offline.

During each interaction a maximum of 10 images of the object are captured (some configurations might not be reachable for the robot). Thus each experiment produces a set of roughly 460 (10 * 46) images.

We use X-means to cluster the dataset composed of the distance traveled by the objects into n classes. Each class is an effect label. In this case we set the number of clusters to be between 2 and 46 (number of objects).

The ambiguity detection lies on the accuracy of the model to predict an object's affordance. We evaluate the model both on the training set (objects used to build the BN), and on the validation set (remaining objects). As described before we already have the interaction information for all objects, so we can train offline, steps by steps.

The threshold is placed at 1.2 times the performance of a random classifier. Which in our case with two identified classes is $60\%(1.2 * 50)$. If the prediction accuracy (either on training or validation sets) remains under this threshold for 5 consecutive interactions, then an ambiguity is raised.

We kept the threshold quite low to allow space for uncertainty in the model. And the 5 consecutive trigger constraint is there to smooth the randomness of the picking order.

The CNN are trained in 3 steps, each step enabling the training of a one more block from the overall structure. In the first step only the fully connected layers are trained. Afterwards each step enables the training of one more convolutional block, starting from the top. Thus the network is progressively trained, avoiding

critical gradient propagation during the training of the first layers, while enabling the fine-tuning of the inner convolutional layers.

The training is done on a Titan Xp, and takes an average of 90 seconds, depending of the size of the dataset and the depth of the architecture. Between 40 sec for 6conv3 architecture with 30% of the dataset, and 120 seconds for VGG16 with 90% the dataset.

Each interaction lasts about 1 minute, it takes approximately one hour to complete a whole experiment.

We repeated the experiment 10 times, and the results are presented below.

We will focus the results on the clustering, ambiguity detection and CNN performance. And we will leave aside the BN as there is no difference regarding the method presented in chapter 3.

Clustering In figure 4.8 we plotted the number of clusters predicted at each step for each run. In figure 4.9 we plotted an example of effect distribution, in this case the run 1.

As we can see in figure 4.9 the 2 classes of effect are quite distinct, with one cluster around 0cm and the other around 10cm. Thus it takes a maximum of 4 objects to stabilise the number of clusters and reach a 100% correct labels. This value depends on the order in which the objects are selected. As soon as at least one object of each class is selected the number of clusters stabilises to 2. This is due to the fact that the two classes of our dataset are very distinct and thus easily linearly separated as we can see in figure 4.9.

Run	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Train Trigger	11	13	9	17	17	13	19	12	13	16
Val Trigger	7	9	7	5	10	9	8	8	6	8

Table 4.1: Exp1 - Movable: Ambiguity Detection Results for run nb 1. The first line is the run ID, second and third lines indicate the step at which the ambiguity was triggered respectively while evaluating the training and validation sets.

Ambiguity Detection We see in table 4.1 that in our scenario, it requires a maximum of 19 steps to trigger an ambiguity. And a maximum of 10 steps if the agent has access to the validation data. As expected it takes more samples to detect an ambiguity when using only the training dataset. In which case the BN model overfits, and learns to predict affordances based on incidentals correlations.

Ambiguity Reduction: CNN Training Following the detection of an ambiguity, a new CNN is instantiated. As we are training offline, we have access to all images and labels from all objects. However we want to emulate the condition of

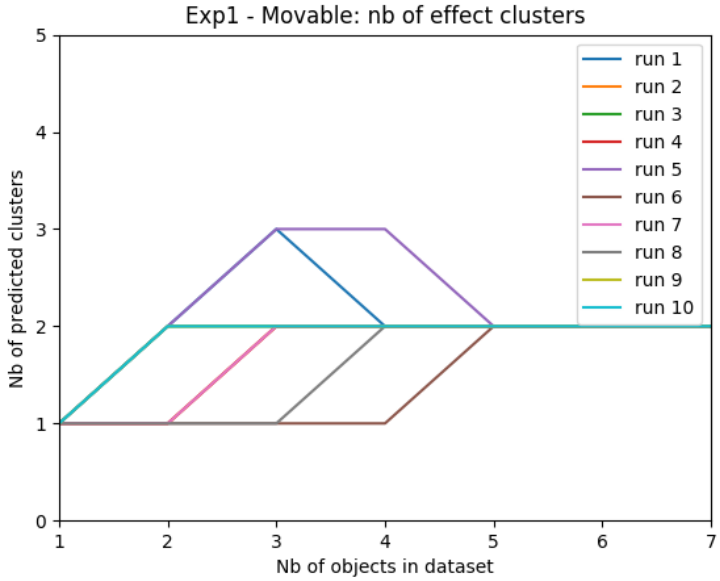


Figure 4.8: Exp1 - Movable: Results of clustering process, in nb of clusters by nb of objects used for the clustering.

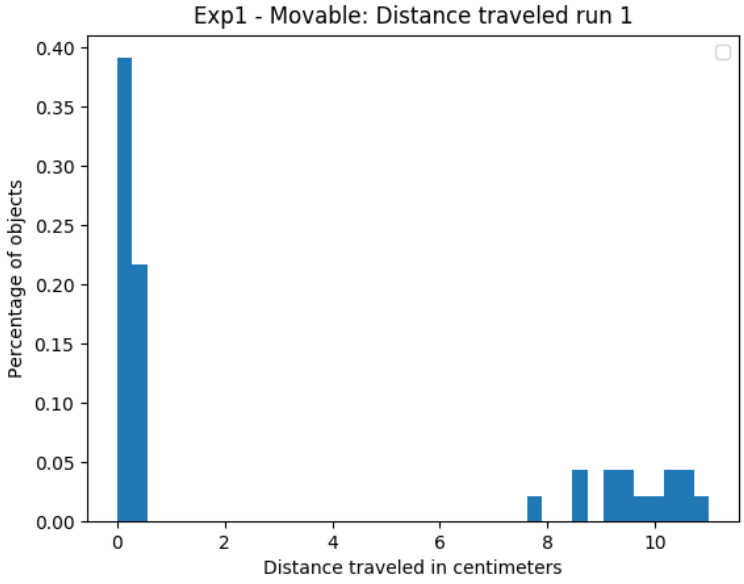


Figure 4.9: Exp1 - Movable: Effects observed in run nb 1, the histogram plots the ratio of objects by distance traveled. We can observe the two distinct population, the non-movable around 0 cm and the movable around 10 cm.

an online learning. Which means that whenever an ambiguity is raised, the agent can only use the data that lead to the ambiguity.

Therefore in a first time, the CNNs were trained with the n first objects, n being the step at which the ambiguity was detected. In a second time, to compare, we also trained the CNNs with 50%, 70% and 90% of the dataset.

To summarise, for each of the 10 runs, we train each architecture, once with n objects, and once with 50%, 70% and 90% of the objects.

The validation performance of the network during training are presented in the following figures 4.10, 4.12, 4.14, 4.16. Each figures contains 4 subfigures, each subfigure plots the results for a different dataset size. The results are the validation performances of the networks for each runs. The validation dataset being composed of the remaining objects unused for training. Thus they contains respectively $(46 - n)$ objects and 50%, 30% and 10% of the dataset.

Similarly the final performances of the networks are presented in the following figures 4.11, 4.13, 4.15, 4.17. Each figures contains 2 subfigures, the top one presents the validation accuracy while the bottom one presents the training accuracy. The histograms show on the x-axis the finale validation accuracy, and on the y-axis the percentage of networks that reached that performance.

The first observation is the overall good performance of the VGG16 network. As we can see in figure 4.10 for all 10 runs the network reach above 90% validation accuracy. In this case it means that the network successfully captured the relevant feature for the task (i.e. the 'texture' of objects). In other terms the ambiguity is correctly reduced, and the agent can know correctly learn the push affordance of these objects.

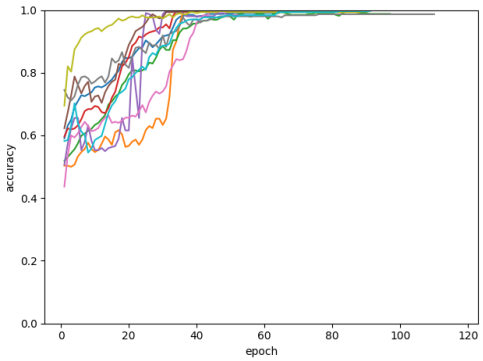
However the same cannot be said for the 10conv3, 8conv3 and 6conv3 architectures, for which the performances are much more unstable:

1. The performance are overall lower, averaging to 70%.
2. We observe that while the VGG16 performed similarly for all 4 datasets sizes, here the network performance is highly dependent of it. The networks performing overall better with a larger dataset as we can see in figure 4.12.
3. The variability between each run is much greater. Which means that the networks are more sensible to bias induced by the randomisation of the training dataset. As we can see in figure 4.16 where the performance for the 70% set vary between 30% and 80%.

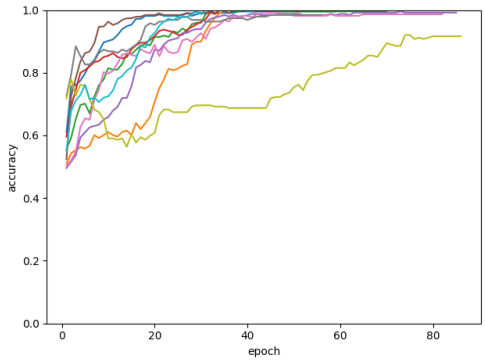
Considering that the architectures of the 10conv3 and VGG16 are almost identical, we can safely hypothesize that the difference in performance comes from the pretraining. Appart from the structure the VGG16 differs from the others by the pretrained weights (1000 classes on ImageNet vs 22 classes in custom dataset). Better pretraining means better features in the lower layers. The network converges faster and generalises better (comments 1 and 2), and is more resilient to bias in the training dataset (comment 3).

An other interesting observation concerns the evolution of performance over training epochs. Let us consider figures 4.14 and 4.14. If we take a closer look at the performance for the 90% set, we can see that several runs have a performance drop after $\tilde{1}0$ epochs, and then again after $\tilde{3}0$ epochs. This can be analysed in regards of our 3-step training method. After each step a new (deeper) block of the network is made trainable, thus the features in this block that were frozen before are now alterable. Although enabling the model to learn lower level features it also increases the risks of over-fitting, even more so if the training dataset is biased.

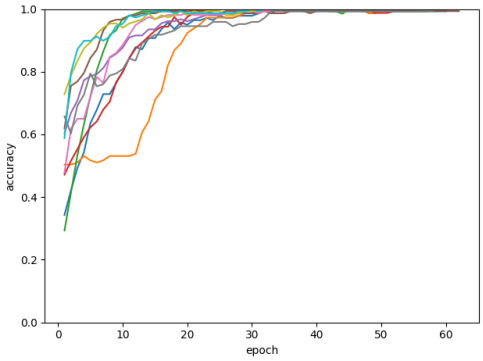
That is why we observe this phenomenom for the 6conv3 and 8conv3 networks (shallow, respectively 2 out of 3 and 2 out 4 conv blocks trained). While the 10conv3 and VGG16 remain more resilient (deeper, 2 out of 5 conv blocks trained).



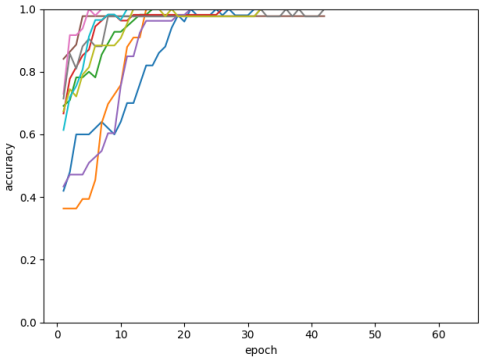
(a) Initial Set



(b) 50% Set



(c) 70% Set



(d) 90% Set

Figure 4.10: Validation performance of the VGG16 architecture over the movable dataset.

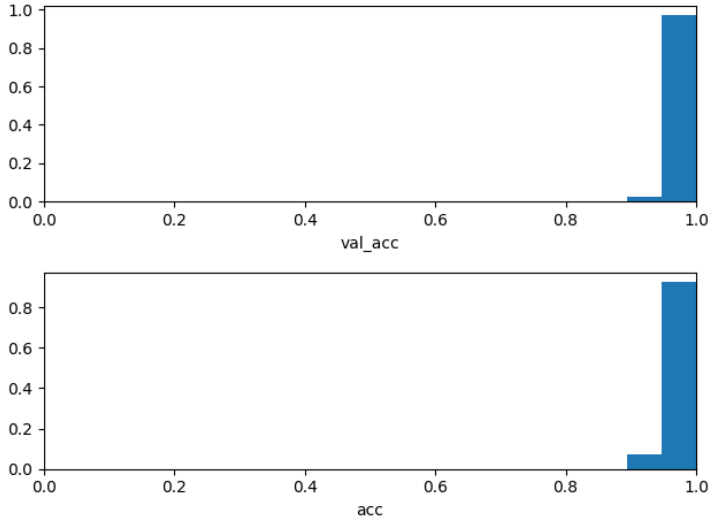
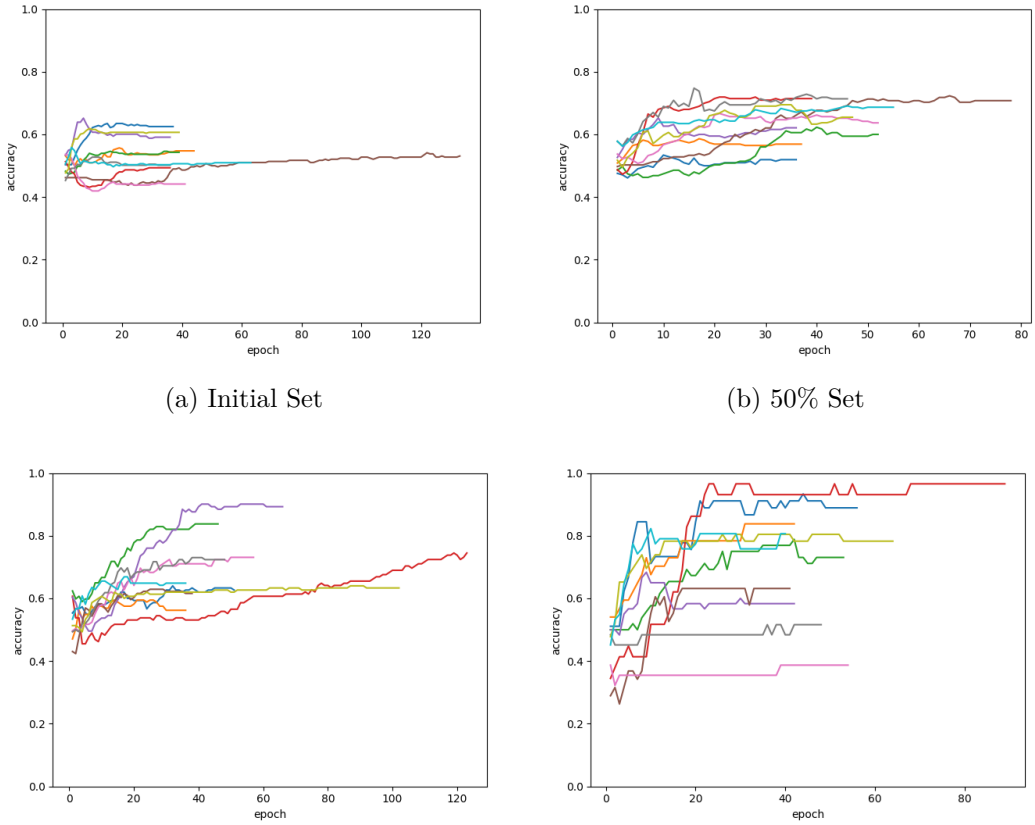


Figure 4.11: Final performance of vgg16 variants on Movable dataset.



(a) Initial Set

(b) 50% Set

(c) 70% Set

(d) 90% Set

Figure 4.12: Validation performance of the 10conv3 architecture over the movable dataset.

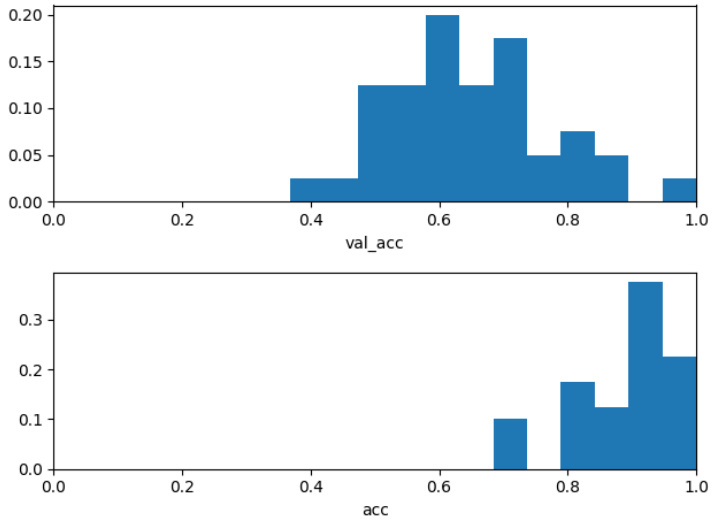
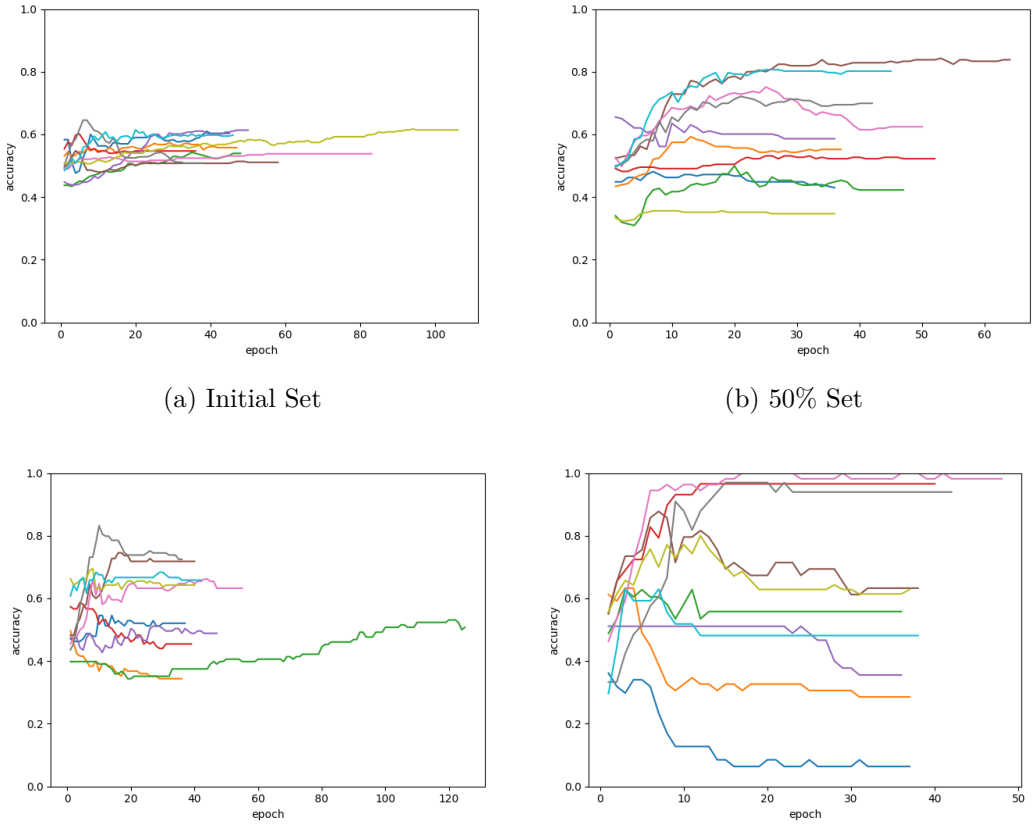


Figure 4.13: Final performance of 10conv3 variants on Movable dataset.



(a) Initial Set

(b) 50% Set

(c) 70% Set

(d) 90% Set

Figure 4.14: Validation performance of the 8conv3 architecture over the movable dataset.

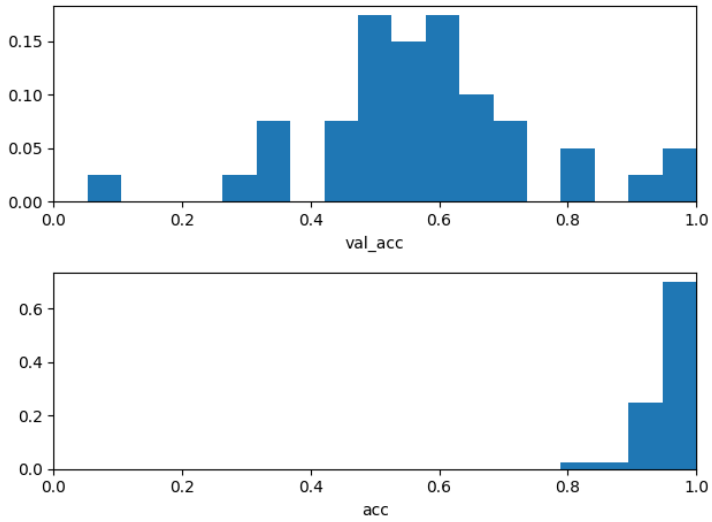
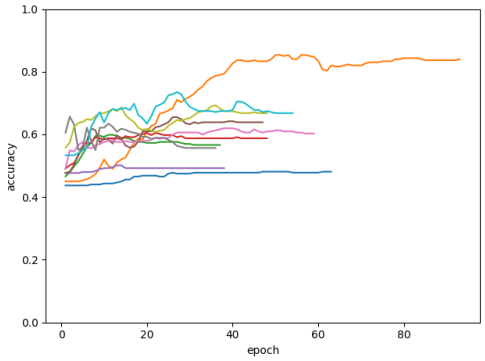
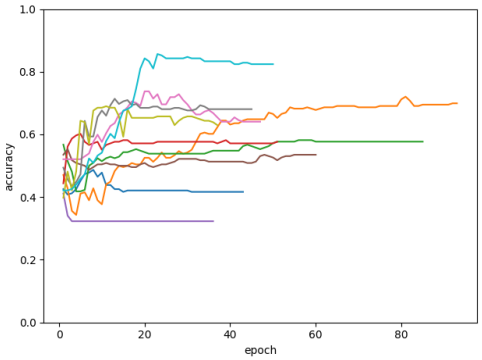


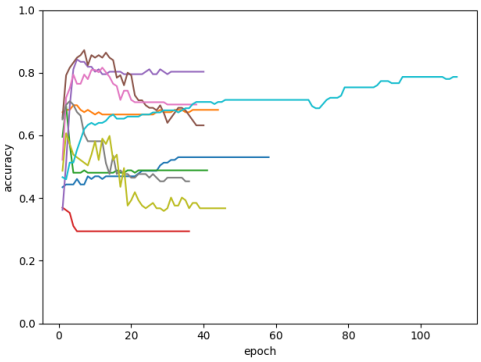
Figure 4.15: Final performance of 8conv3 variants on Movable dataset.



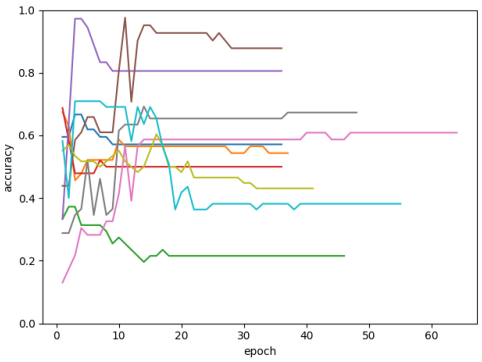
(a) Initial Set



(b) 50% Set



(c) 70% Set



(d) 90% Set

Figure 4.16: Validation performance of the 6conv3 architecture over the movable dataset.

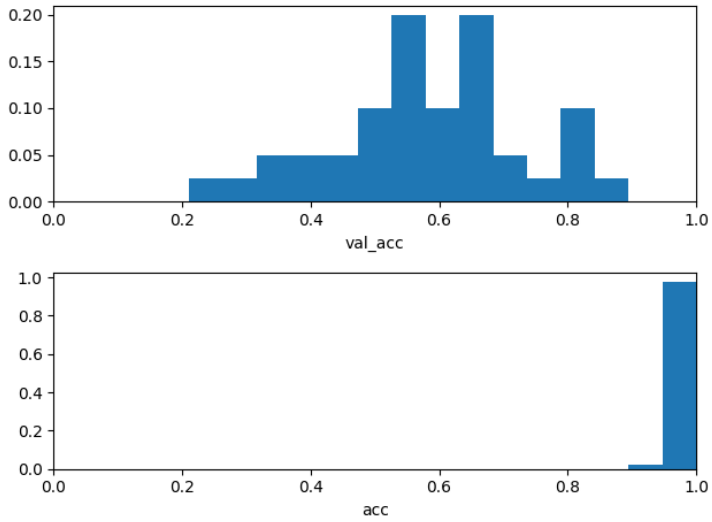


Figure 4.17: Final performance of 6conv3 variants on Movable dataset.

4.4.4 Experiment 2: Rollable objects

4.4.4.1 Setup

In this second experiment the agent has to learn the *rollability* of a set of objects.

Objects Dataset The set is constituted of 37 objects, of whom 18 are non-rollable, and 19 rollable. The rolling objects either possess wheels or are of round shapes. The non-rolling ones are a variety of common items. See figure 4.18 for objects examples.

Figure 4.18: Examples of objects from experiment 2, rollable / non-rollable dataset.

Poke Primitive To interact with the objects the agent is given a *poke* action primitive. This primitive is similar to the *push* primitive described in section 4.4.3, to the exception of move forward which is replaced by a quick rotation of the wrist. Thus the robot transmits more kinetic energy to the object, which enables it to separate rolling from non-rolling ones.

The *poke* action consists in the following sequence of motor commands:

- **Approach:** move the end-effector 5 cm away from the target object. The end-effector is placed aligned to the axis formed the object's center and the robot base.
- **Poke:** rotate the wrist joint upward by 40 degrees (increased velocity compared to previous push action).
- **Retract:** return end-effector to the starting position.

Movement Detector Identical movement effect detector as in 4.4.3.

Initial Features Identical Initial Features as in 4.4.3.

CNN Architectures Identical CNN architectures as in 4.4.3.

4.4.4.2 Results

The results that follow have been produced with the same meta parameters as experiment 1, see 4.4.3.2 for details.

Effect Clustering Similarly to the first experiment (4.4.3),

Run	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Train Trigger	14	13	11	14	12	19	14	9	15	12
Val Trigger	9	11	7	8	8	10	6	7	10	

Table 4.2: Exp2 - Rollable: Ambiguity Detection Results

Ambiguity Detection

Ambiguity reduction The validation performance of the network during training are presented in the following figures 4.21, 4.23, 4.25, 4.27. Identically to 4.4.3.2, each figures contains 4 subfigures, each subfigure plots the results for a different dataset size. The results are the validation performances of the networks for each runs. The validation dataset being composed of the remaining objects unused for training. Thus they contains respectively $(37 - n)$ objects and 50%, 30% and 10% of the dataset.

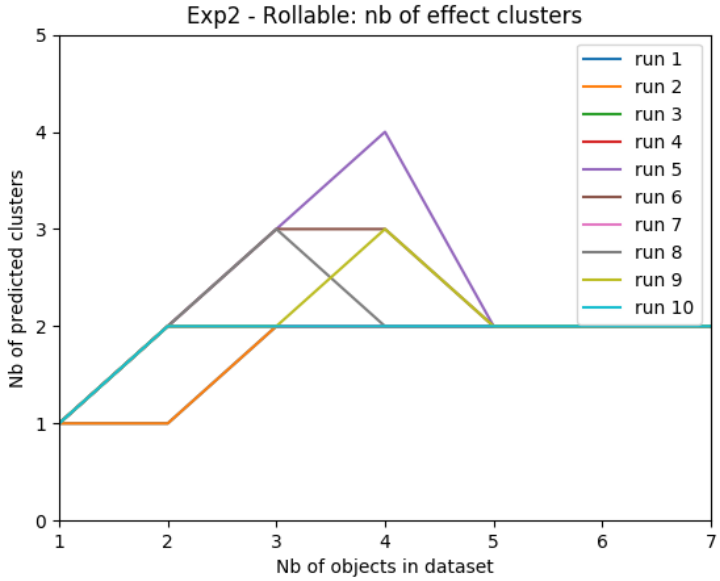


Figure 4.19: Exp2 - Rollable: results of the clustering process, in nb of clusters by nb of objects used for the clustering.

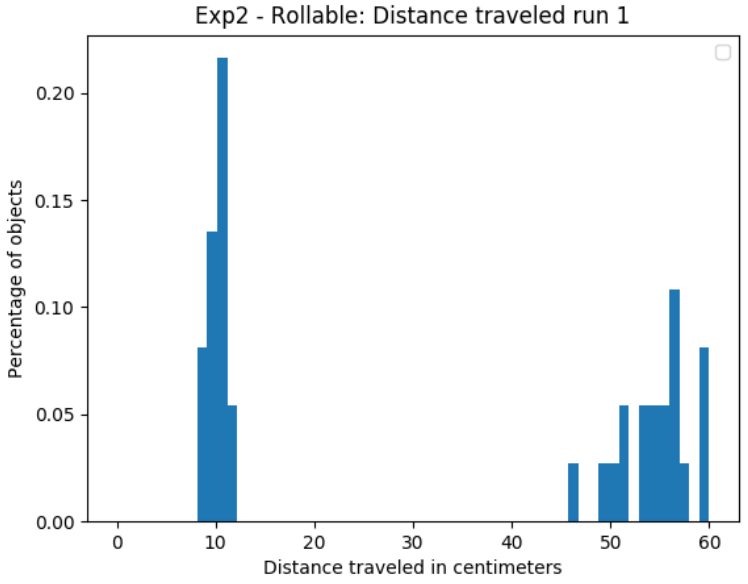


Figure 4.20: Exp2 - Rollable: Effects observed in run nb 1, the histogram plots the ratio of objects by distance traveled. We can observe the two distinct population, the non-rollable around 10cm and the rollable around 50 cm.

The final performances of the network are presented in the following figures 4.22, 4.24, 4.26, 4.28. Each figures contains 2 subfigures, the top one presents the validation accuracy while the bottom one presents the training accuracy. The histograms show on the x-axis the finale validation accuracy, and on the y-axis the percentage of networks that reached that performance.

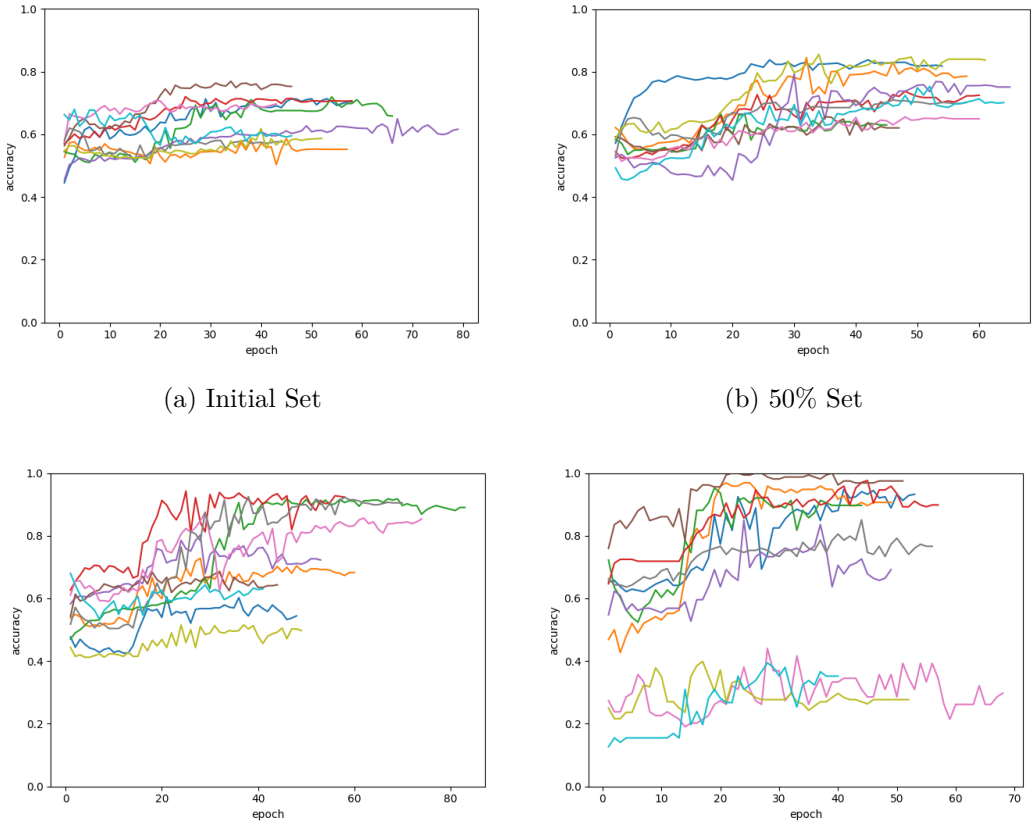
We can observe that the remarks regarding the first experiment (see section 4.4.3.2) are overall still relevant here. The VGG16 network outperforms the other architectures due to the better pretraining.

Therefore we will focus on the differences between both experiments.

Firstly we can observe that the average performances of all networks are lower than for experiment 1. We can hypothesize 2 reasons for this:

1. The features required to predict rollability are more complex than the 'texture' feature.

2. The dataset is composed of different kinds of 'rollable' objects (namely, sphere, toy cars and cylinders). Furthermore it is smaller (37 objects instead of 46). Which increases the probability for a biased training dataset. This is particularly clear in figure 4.21 where 3 runs have critically low performances.



(a) Initial Set

(b) 50% Set

(c) 70% Set

(d) 90% Set

Figure 4.21: Validation performance of the VGG16 architecture over the rollable dataset.

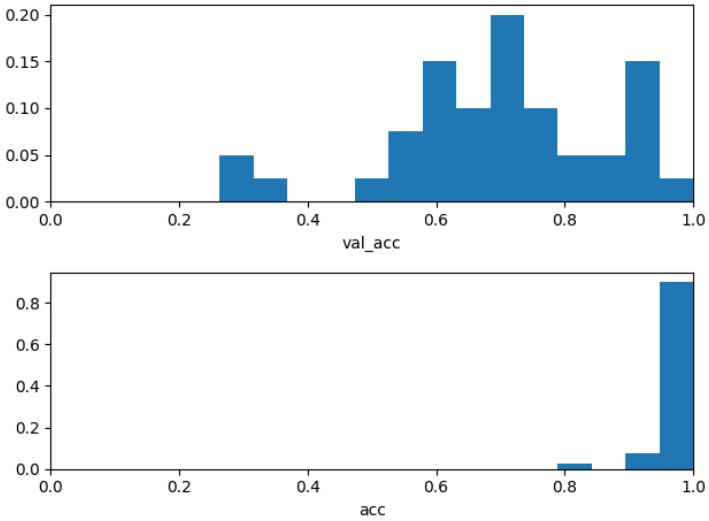
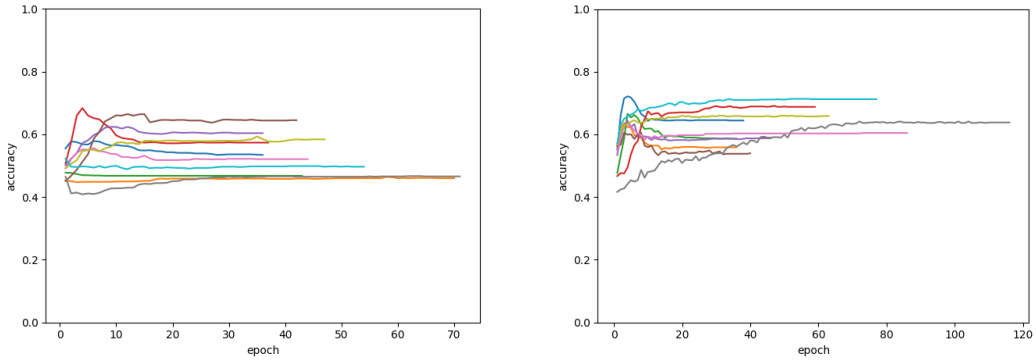
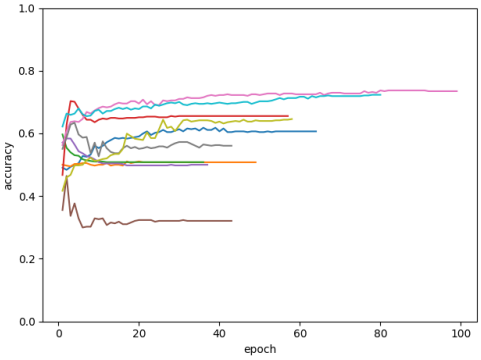


Figure 4.22: Final performance of vgg16 variants on Rollable dataset.

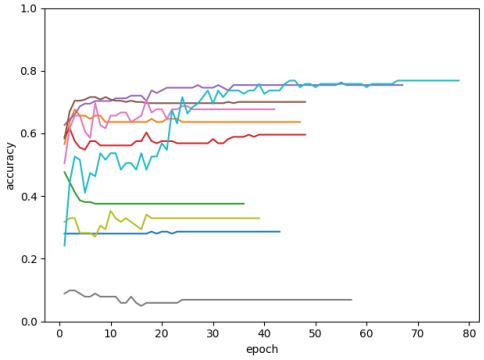


(a) Initial Set

(b) 50% Set



(c) 70% Set



(d) 90% Set

Figure 4.23: Validation performance of the 10conv3 architecture over the rollable dataset.

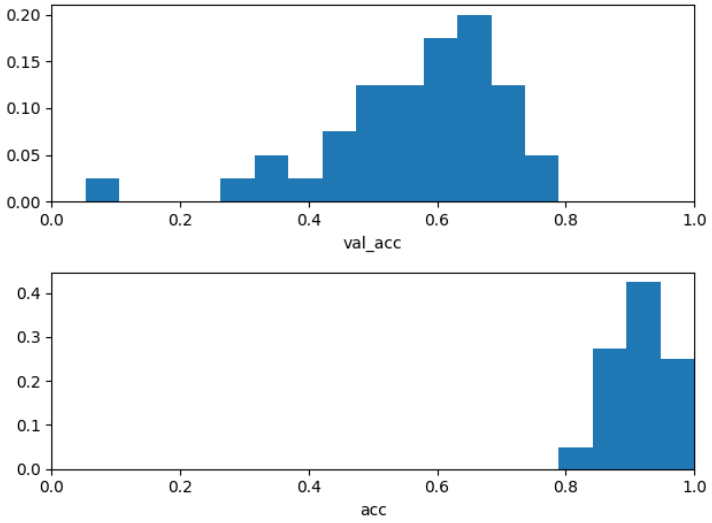
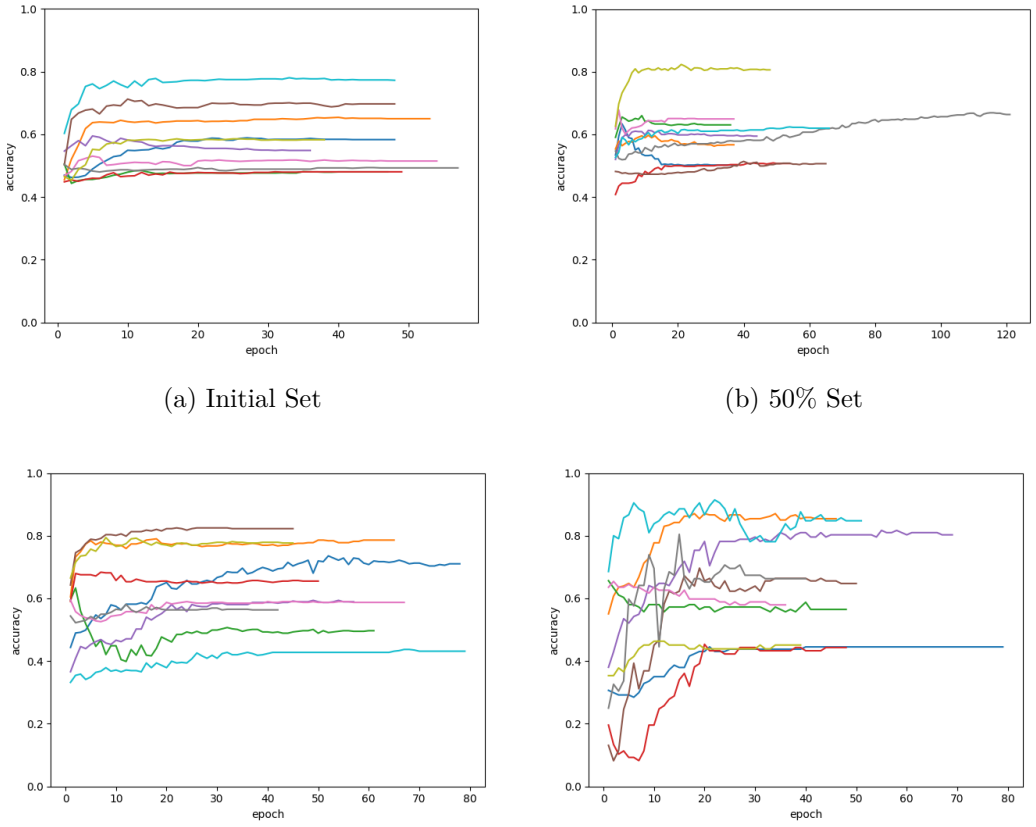


Figure 4.24: Final performance of 10conv3 variants on Rollable dataset.



(a) Initial Set

(b) 50% Set

(c) 70% Set

(d) 90% Set

Figure 4.25: Validation performance of the 8conv3 architecture over the rollable dataset.

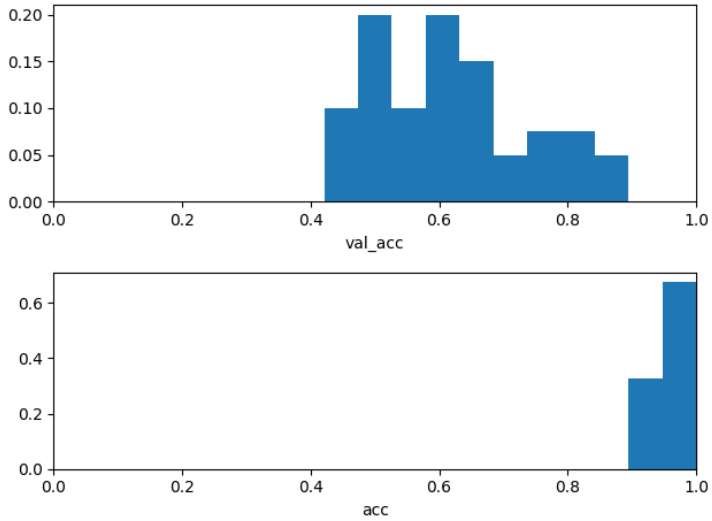
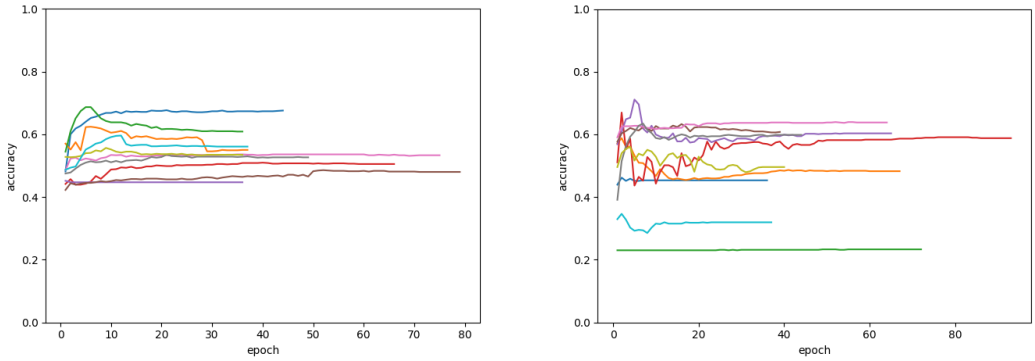
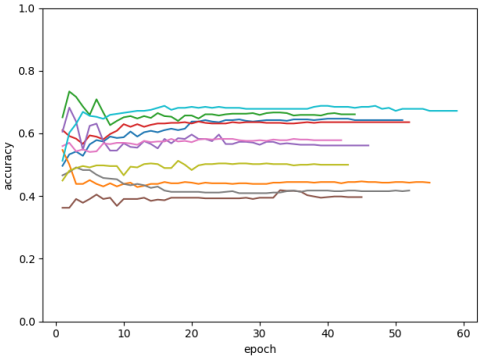


Figure 4.26: Final performance of 8conv3 variants on Rollable dataset.

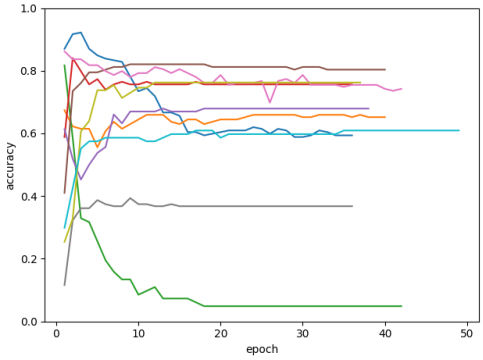


(a) Initial Set

(b) 50% Set



(c) 70% Set



(d) 90% Set

Figure 4.27: Validation performance of the 6conv3 architecture over the rollable dataset.

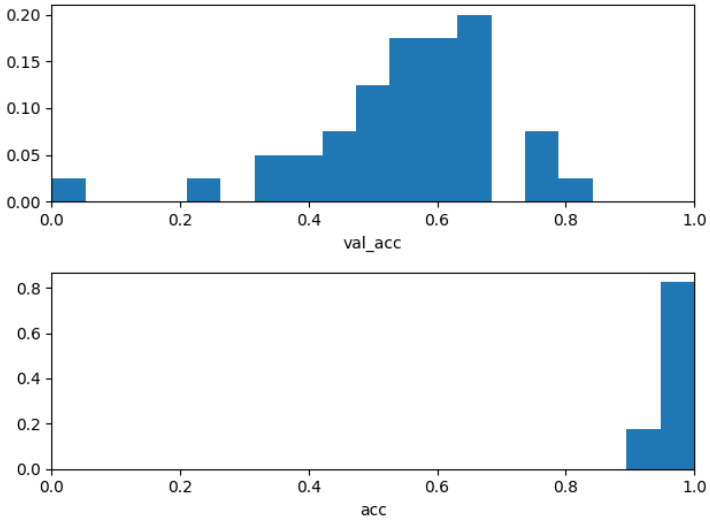


Figure 4.28: Final performance of 6conv3 variants on Rollable dataset.

4.4.5 Experiment 3: No Pretraining

4.4.5.1 Setup

In this third experiment we wanted to validate the importance of pretraining the network (as was hypothesized in both previous experiments 4.4.3.2 and 4.4.4.2). So we trained variants from the 6conv3, 8conv3 and 10conv3 networks with random initial weights. We did not test the VGG16 variant as it is identical to the 10conv3 variant. The exact grid search parameters are detailed in the annexes (A.1.2.2).

4.4.5.2 Results

The overall results are presented in the following figures: Figure 4.29 and 4.30 plot the final performance of each network respectively on the rollable dataset and on the movable dataset.

As expected, the performances are no better than random (2 classes, thus 50% correct prediction) for all architectures, confirming the importance of pretraining. Even the training accuracy is substantially lower, close to randomness while it was reaching over 90% with pretrained networks.

We can notice however that the performances with the rollable dataset is better, which can be explained by the smaller size of the dataset, and the overall greater diversity of appearances of objects. Thus it is easier for the network to over-fit and simply learn to distinguish objects.

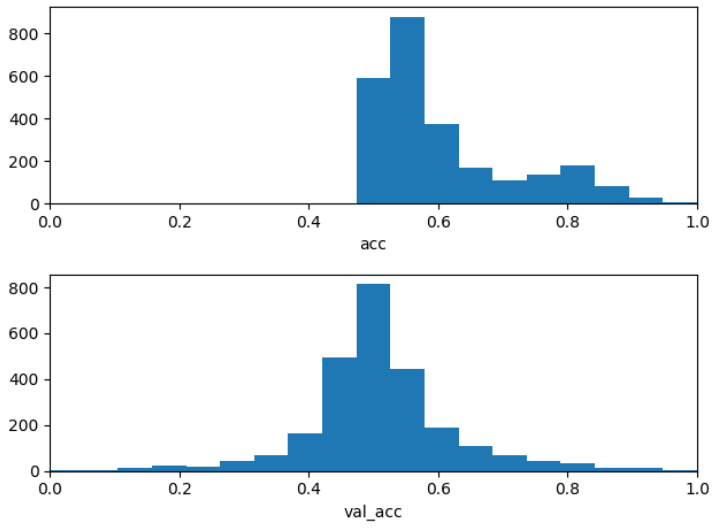


Figure 4.29: Final performance of all models on rollable dataset without pretraining. Top and bottom figure plot respectively the accuracy on training and validation sets.

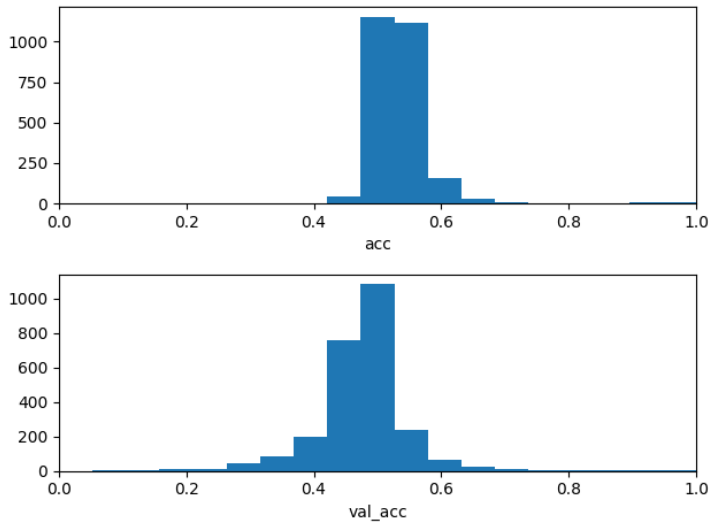


Figure 4.30: Final performance of all models on movable dataset without pretraining. Top and bottom figure plot respectively the accuracy on training and validation sets.

4.5 Perceiving Affordances

The main incentive of this work is to enable a robot to better comprehend its environment by using Affordance theory Gibson (1977). That is to enable a robot to perceive its environment in terms of action possibilities.

To do so we proposed to use *IP* Bohg et al. (2017) to collect sensorimotor data, then *CNN* and *BN* to learn the underlying representations (i.e. affordances). The presented results, although limited to simple tasks, hinted that our method can successfully enable a robot to learn the affordances of objects by re-purposing the convolutional layers of a pretrained CNN.

However learning an affordance and perceiving it are two different things. To learn an affordance is to discover the existing relations between action capabilities and sensory inputs. Which in the formalism from Sahin et al. (2007) consists in learning the relations between *objects*, *actions* and *effects*. While to perceive affordances is to infer the action possibilities offered by a scene. Which, respectively, consists in inferring the *effects* that the *actions* available to the agent could produce on the objects from the scene.

Therefore how can our method be used in that aspect ? In our approach this can be done in two ways:

- Segment the environment into objects hypothesis according to the method presented in 4.2.3. Then compute their features (both predefined and newly trained CNN). Finally infer the affordances through the BN.

- Convert a trained CNN into a fully convolutional network FCN, then directly

compute a 2D heatmap of the environment.

The first method is straight forward considering our overall model. Through our segmentation method the agent can create hypothesis of objects, extract 2d images of them, and then finally classify the hypothesis using the trained CNNs.

The second method however requires to firstly convert a trained network into a FCN. However it has the noticeable advantage of not requiring any a priori knowledge of the structure of the environment. Once the FCN is created, it can be used as it is, to compute a heatmap of the scene. Furthermore in our approach each CNN is partially specific to an action. Thus, without any prior information of the environment, the agent can use a FCN to detect the interesting parts of its environment in regard of the associated affordance.

4.6 Conclusion and Discussion

4.6.1 Contributions and Limitations

In this chapter we proposed a method to discover affordances while learning the corresponding pertinent visual features. Overall the presented method is able to correctly re-purpose a pretrained CNN to learn features relevant to an affordance with some promising properties:

- The architecture and meta-parameters of the CNNs are not crucial in these tasks of features extraction. Thus a single network architecture will fit different tasks, which is a required property of the model in order to allow for open-ended learning.

- An ambiguity can be reduced in a small number of objects. Hence in a real robotic setup the agent can learn an affordance without requiring to interact with thousands of objects. Which is necessary if we expect the agent to learn or at least adapt a pre-existing affordance in a real environment.

However as the results have shown, some strong limitations also remains:

- The quality of the pretraining is crucial. There is a significant gap in performance between the CNNs trained on our custom dataset (containing 22 classes), and the CNNs trained on a dataset containing a thousand classes. This means that this method cannot be done in a completely agnostic approach, and requires some pre-existing knowledge about the structure of the environment in the form of pretrained weights. However the honorable performance of the lesser network such as the 10conv3 iteration see figure 4.23 and 4.12, especially when training with 50% or 70% of the dataset indicates that this pretraining gap is reachable. By reachable we refer to the initial intention of this approach, which is to enable a life-long learning of visual features by incorporating this method in a developmental learning framework.

- The picking order is critical. Random sampling results in a high variance of the validation performance of the networks. This however can be dampened by pretraining, as the results of VGG16 variant network have shown. This means that a meta-learning controller must be added in order to evenly pick objects from the dataset, and furthermore, a motivational system to guide the agent interactions.

It is noticeable that the way the features are learned fit well with the paradigm of active perception and affordances. The features are directly linked to the affordance, therefore to the corresponding effect and action. Which mean that our method enable the system to build representation that will let it perceive the world in term of actions and motivations.

More precisely, when converting a trained CNN to a FCN, the agent can interpret images in terms of actions results. Thus it can be used by a motivation system to directly perceive interesting area of the environment for a specific interaction.

If the agent wishes to look for a hammer, it needs only using the corresponding FCN. Furthermore the processing of one image through the FCN is one degree of magnitude faster than the segmentation appraoch. Thus it is usable in real time without impeding the responsiveness of the agent.

In a way, the ability for the agent to 'directly' perceive the affordance through the FCN could be related to the original view of Gibson on affordances.

4.6.2 Future Work

As mentioned above, the proposed approach remains preliminary by some aspects, and the results need to be extended to more real robotic setup to better outline the contribution of this method.

We are considering several directions for future work:

- **FCN**: The perspective of using the trained CNNs to directly perceive affordances should be studied more. Combined with a motivational system to select which set of features to use at a given time, combining heatmap could enable the agent to directly perceive in its environment objects that fits its current goal. For instance combining graspable with heavy when looking for a hammer.

- **More Robotic Setup**: So far we only tested our methods in 2 scenario, a significant improvement would be to test it with several others, more complex ones.

- **Different Agents**: One fundamental aspect of affordances is that the relata learn by an agent are directly correlated to its perceptive and motor capabilities. Therefore, 2 different agents in an identical environment should not build the same affordances representations. We could explore that aspect by implementing our model on different robotic setup. This could be done for

instance simply by altering the sensor capabilities of the agent (e.g. by changing color images to grey images) or motor capabilities (smaller gripper, wider grip, etc). Or by using a whole different platform, for instance a turtle bot.

- **Removing Action Predefinition**: In this work we focus on removing the predefinition of object's features on the assumption that they limit the space of discoverable affordances. And we proposed the concept of ambiguity to do so. An interesting perspective could be to remove the predefinition of actions using the same concept. The agent creates sensorimotor data by exploring its motor space, at which point it can discover an action that results in an ambiguity, i.e., this action produces different effects on otherwise similar objects. In other words we could use the ambiguity detection to drive a motor control exploration.

Chapter 5

Conclusion and Discussion

Contents

5.1 Contributions

In this thesis we have proposed a framework for learning affordances while simultaneously learning the corresponding pertinent visual features. The main goal was to enable life-long learning by avoiding the predefinition of visual features.

The methods relies on a few assumptions:

1. The extraction of features is only visual, thus it limits the nature of the discoverable affordances.

2. The clustering is based on the ability of the agent to perceive the adequate environmental features.

3. The agent requires prior knowledge in order to act (action primitive) and to segment the environment (objects segmentation).

5.2 Discussion and Future Work

In the proposed approach, when encountering an ambiguity the proposed solution is to create a new visual feature to reduce it. However the actual required feature might not be visual, but haptic, or proprioceptive, an extension to our method could be to integrate a broader sensory spectrum.

However it could be related to a more global problem of perception. The inherent limits of sensors. Each agent is in fine limited by its own embodiment.

We do not use the time component to understand the environment. However many approaches proposes to integrate signals over time to construct better representations. For instance perceiving a round is the continuity of sensory input through movement. Rather than actually perceiving a "circle".

The main limitation of our method lies in the pretraining. In this study we proposed two different set of pretrained weights. And the results have demonstrated

the gap between the two. It was to be expected, and furthermore it was our intention to test the limits of the approach. In terms of applications it is not a problem as weights are now easily available, as well as large datasets. So the method is applicable in many contexts and with various input shape for the CNNs.

However it raises a question regarding our approach, can the features really be trained in such a small number of interactions? In other words, if the features were not already pre-existent in the network could they be built by our method? We argued in section 4.2 that we intend the framework to be used as life-long learning method. Thus each new ambiguity reduction contributes to building robust and more general features in the lower layers of the CNNs.

But remains the problem of the bootstrap. Our method relies on a few hypothesis, objects segmentation, effect detectors and primitives of actions.

As an extension and future work, we intend to include our method in a developmental learning framework.

An important advantage of our approach is that the built features are directly linked to goals trough actions-effects, therefore they can be easily selected with a focus or motivation system. In a broader aspect this question of whether an affordance should be linked to a motivation is very interesting. In our opinion a system only building affordances without motivation (i.e. using affordances as a hierarchical evaluation of the environment for decision making) would not be useful except for reactive agents. In the other hand a fully motivated system does not really require affordances. More precisely in a fully motivated system, affordances will most likely be limited to reinforcement learning. Or more precisely they will not distinguish themselves from it. Therefore we believe that a good approach could be to settle in the middle. Affordances bring a very good explanation of what 'intuitive' perception could be, and therefore enable the agent to perceive the environment in an uncostly manner. Add a touch of motivation or state mind to select which affordances to use at a current state and you get a good way of trading off between computer cost and a qualitative perception of the environment for predicting objects behavior.

Bibliography

Bajcsy, R. (1988). Active perception. (Cité en page 7.)

Bajcsy, R., Aloimonos, Y., and Tsotsos, J. K. (2018). Revisiting active perception. *Autonomous Robots*, 42(2):177–196. (Cité en pages 7, 8 et 36.)

Battaglia, P., Pascanu, R., Lai, M., Rezende, D. J., et al. (2016). Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pages 4502–4510. (Cité en page 8.)

Bohg, J., Hausman, K., Sankaran, B., Brock, O., Kragic, D., Schaal, S., and Sukhatme, G. S. (2017). Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291. (Cité en pages 3, 8, 9, 14, 54 et 74.)

Campbell, M., Hoane Jr, A. J., and Hsu, F.-h. (2002). Deep blue. *Artificial intelligence*, 134(1-2):57–83. (Cité en page 2.)

Canziani, A., Paszke, A., and Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*. (Cité en page 48.)

Chatila, R., Renaudo, E., Andries, M., Chavez-Garcia, R. O., Luce-Vayrac, P., Gottstein, R., Alami, R., Clodic, A., Devin, S., Girard, B., and Khamassi, M. (2018). Toward Self-Aware Robots. *Frontiers in Robotics and AI*, 5:88. (Cité en page 14.)

Chavez-Garcia, R. O., Andries, M., Luce-Vayrac, P., and Chatila, R. (2016a). Discovering and manipulating affordances. In *2016 International Symposium on Experimental Robotics (ISER)*. (Cité en pages 14 et 38.)

Chavez-Garcia, R. O., Luce-Vayrac, P., and Chatila, R. (2016b). Discovering affordances through perception and manipulation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3959–3964. IEEE. (Cité en pages 14, 38 et 42.)

Christoph Stein, S., Schoeler, M., Papon, J., and Worgotter, F. (2014). Object partitioning using local convexity. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. (Cité en pages 19 et 42.)

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619. (Cité en pages 19 et 42.)

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee. (Cité en page 50.)

Dotov, D. G., Nie, L., and De Wit, M. M. (2012). Understanding affordances: history and contemporary development of gibson's central concept. *Avant: the Journal of the Philosophical-Interdisciplinary Vanguard*. (Cité en page 11.)

Gibson, J. J. (1966). The senses considered as perceptual systems. (Cité en page 10.)

Gibson, J. J. (1977). Perceiving, acting, and knowing: Toward an ecological psychology. *The Theory of Affordances*, pages 67–82. (Cité en pages 3, 6, 8, 15, 36, 39 et 74.)

Gibson, J. J. (1979). The ecological approach to human perception. (Cité en pages 6, 10, 15 et 36.)

Hanson, A. (1978). *Computer vision systems*. Elsevier. (Cité en page 2.)

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*. (Cité en page 48.)

Högman, V., Björkman, M., Maki, A., and Kragic, D. (2016). A sensorimotor learning framework for object categorization. *IEEE Transactions on Cognitive and Developmental Systems*, 8(1):15–25. (Cité en page 8.)

Jones, K. S. (2003). What is an affordance? *Ecological psychology*, 15(2):107–114. (Cité en page 11.)

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*. (Cité en page 45.)

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. (Cité en pages 45 et 48.)

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436. (Cité en page 48.)

Li, K. and Meng, M. Q.-H. (2015). Learn like infants: A strategy for developmental learning of symbolic skills using humanoid robots. *International Journal of Social Robotics*, 7(4):439–450. (Cité en page 17.)

Lopes, M. and Santos-Victor, J. (2007). A developmental roadmap for learning by imitation in robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):308–321. (Cité en page 14.)

Lungarella, M., Metta, G., Pfeifer, R., and Sandini, G. (2003). Developmental robotics: a survey. *Connection science*, 15(4):151–190. (Cité en page 39.)

Mahler, J., Liang, J., Niyaz, S., Laskey, M., Doan, R., Liu, X., Ojea, J. A., and Goldberg, K. (2017a). Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*. (Cité en page 39.)

Mahler, J., Matl, M., Liu, X., Li, A., Gealy, D., and Goldberg, K. (2017b). Dex-net 3.0: Computing robust robot suction grasp targets in point clouds using a new analytic model and deep learning. *arXiv preprint arXiv:1709.06670*. (Cité en page 39.)

Nguyen, A., Kanoulas, D., Caldwell, D. G., and Tsagarakis, N. G. (2016). Detecting object affordances with convolutional neural networks. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 2765–2770. IEEE. (Cité en page 39.)

Nguyen, A., Kanoulas, D., Caldwell, D. G., and Tsagarakis, N. G. (2017). Object-based affordances detection with convolutional neural networks and dense conditional random fields. In *International Conference on Intelligent Robots and Systems (IROS)*. (Cité en page 39.)

Nilsson, H., Courtney, A., and Peterson, J. (2002). Functional reactive programming, continued. In *Proceedings of the 2002 ACM SIGPLAN workshop on Haskell*, pages 51–64. ACM. (Cité en page 6.)

Oliva, A. (2005). Gist of the scene. In *Neurobiology of attention*, pages 251–256. Elsevier. (Cité en page 2.)

O'Regan, J. K. and Noë, A. (2001). A sensorimotor account of vision and visual consciousness. *Behavioral and brain sciences*, 24(5):939–973. (Cité en page 15.)

Papon, J., Abramov, A., Schoeler, M., and Worgotter, F. (2013). Voxel cloud connectivity segmentation-supervoxels for point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2027–2034. (Cité en pages 18, 19 et 42.)

Pastore, N. (1974). Selective history of theories of visual perception: 1650-1950. (Cité en page 7.)

Pelleg, D., Moore, A. W., et al. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. In *Icml*, volume 1, pages 727–734. (Cité en pages 37 et 44.)

Rawat, W. and Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449. (Cité en page 45.)

Sahin, E., Cakmak, M., Dogar, M. R., Ugur, E., and Ucoluk, G. (2007). To afford or not to afford: A new formalization of affordances toward affordance-based robot

control. *Adaptive Behavior*, 15(4):447–472. (Cité en pages 15, 16, 17, 21, 27, 36, 41 et 74.)

Schwarz, G. et al. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464. (Cité en page 22.)

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423. (Cité en page 22.)

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer. (Cité en page 18.)

Simons, D. (2010). The monkey business illusion. *Retrieved June*, 12:2015. (Cité en page 7.)

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. (Cité en pages 46 et 48.)

Sucan, I. A. and Chitta, S. (2015). Moveit!(2015). (Cité en page 55.)

Sucan, I. A., Moll, M., and Kavraki, L. E. (2012). The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82. (Cité en page 55.)

Tenenbaum, J. M. (1970). Accommodation in computer vision. Technical report, Stanford Univ Ca Dept of Computer Science. (Cité en page 7.)

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78. (Cité en page 23.)

Ugur, E., Nagai, Y., Sahin, E., and Oztop, E. (2015). Staged development of robot skills: Behavior formation, affordance learning and imitation with motionese. *IEEE Transactions on Autonomous Mental Development*, 7(2):119–139. (Cité en page 14.)

Ugur, E., Oztop, E., and Sahin, E. (2011). Goal emulation and planning in perceptual space using learned affordances. *Robotics and Autonomous Systems*, 59(7-8):580–595. (Cité en page 39.)

Ugur, E. and Piater, J. (2016). Emergent structuring of interdependent affordance learning tasks using intrinsic motivation and empirical feature selection. *IEEE Transactions on Cognitive and Developmental Systems*. (Cité en page 39.)

Van Hoof, H., Kroemer, O., and Peters, J. (2014). Probabilistic segmentation and targeted exploration of objects in cluttered environments. *IEEE Transactions on Robotics*, 30(5):1198–1209. (Cité en page 18.)

Varadarajan, K. M. and Vincze, M. (2012). Afnet: The affordance network. In *Asian Conference on Computer Vision*, pages 512–523. Springer. (Cité en page 39.)

Wade, N. J. (2000). *A natural history of vision*. MIT press. (Cité en page 7.)

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328. (Cité en page 45.)

Zech, P., Haller, S., Lakani, S. R., Ridge, B., Ugur, E., and Piater, J. (2017). Computational models of affordance in robotics: a taxonomy and systematic classification. *Adaptive Behavior*, 25(5):235–271. (Cité en pages 11, 14 et 36.)

Appendix A

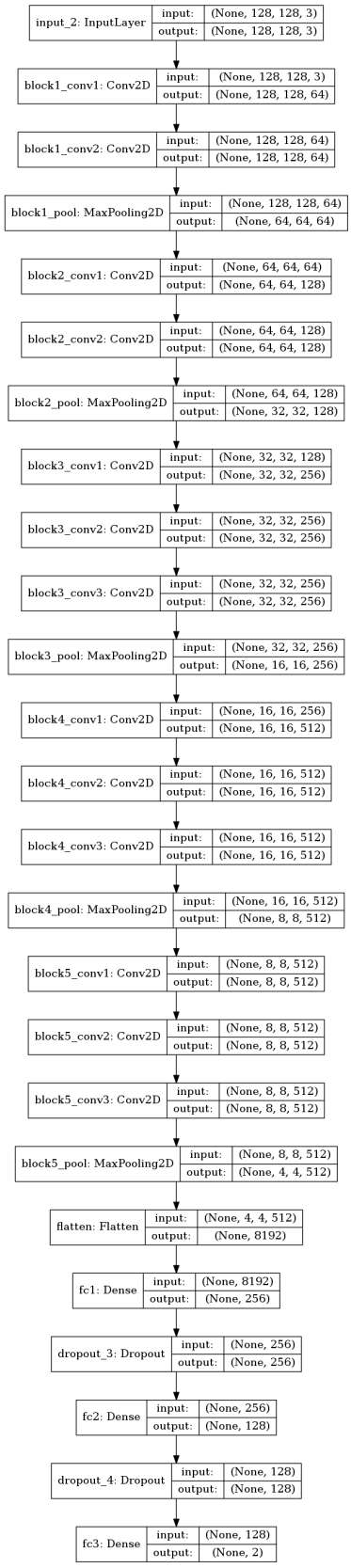
Annexes

In these annexes we present detailed versions of the methods that are used throughout the manuscript (see section A.1). Along with supplementary results (see section **??**) and experiments details (see section A.1.1 and A.1.2).

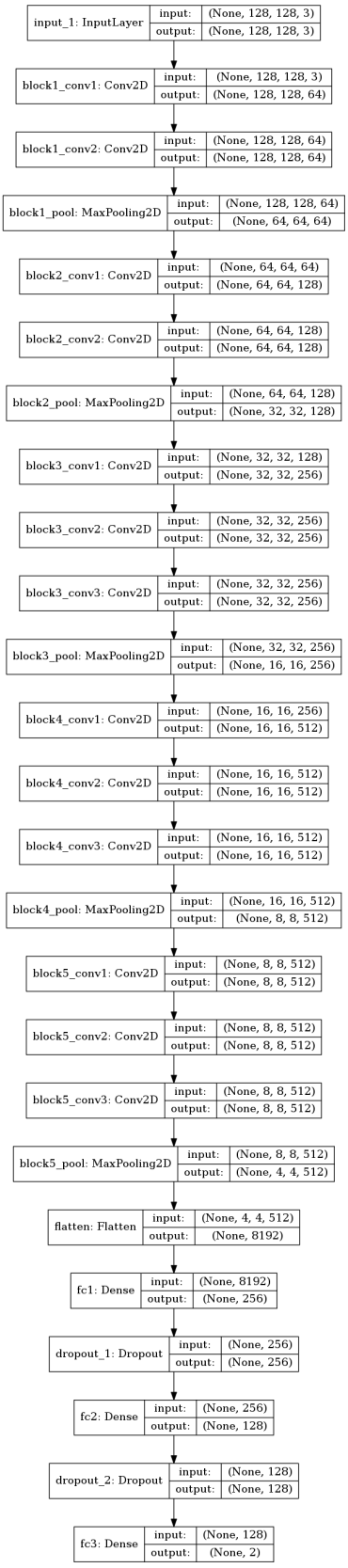
A.1 Methods

This section presents supplementary details regarding the methods that we proposed and their implementations.

A.1.1 CNNs Architectures

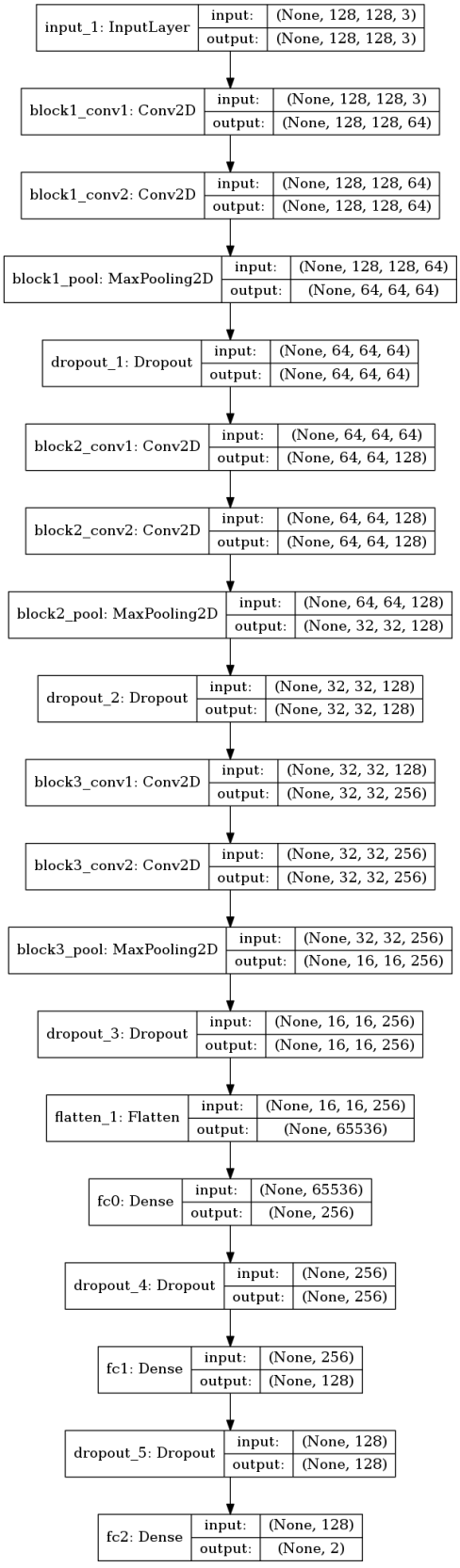


(a) **VGG16__K** Structure, 256x128 top block layers.



(b) **VGG16** Structure, 256x128 top block layers.

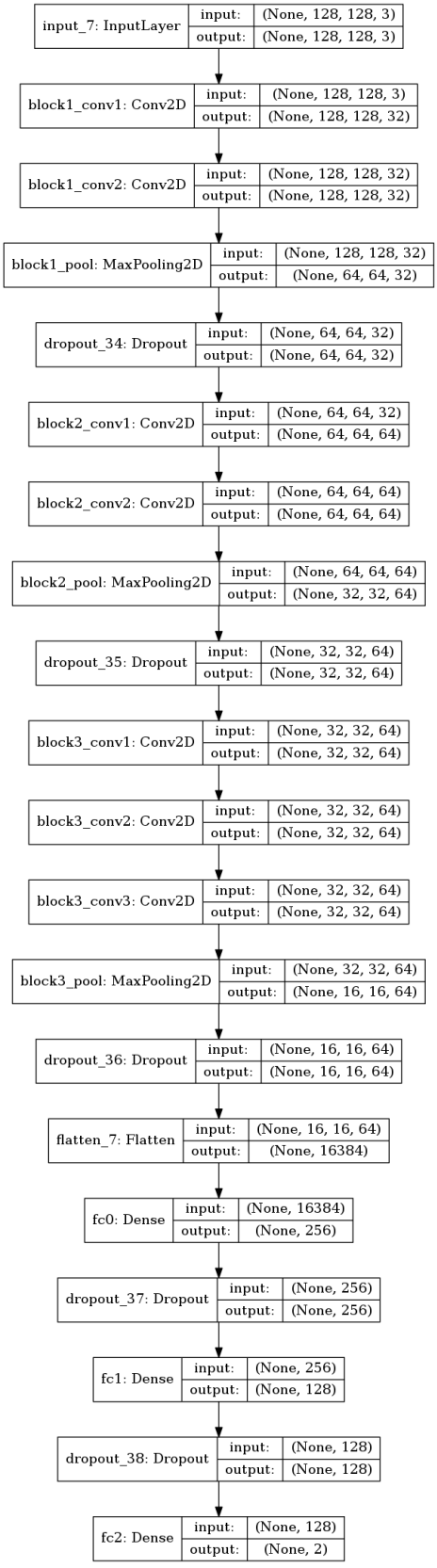
Figure A.1: **VGG16** Variants Architectures



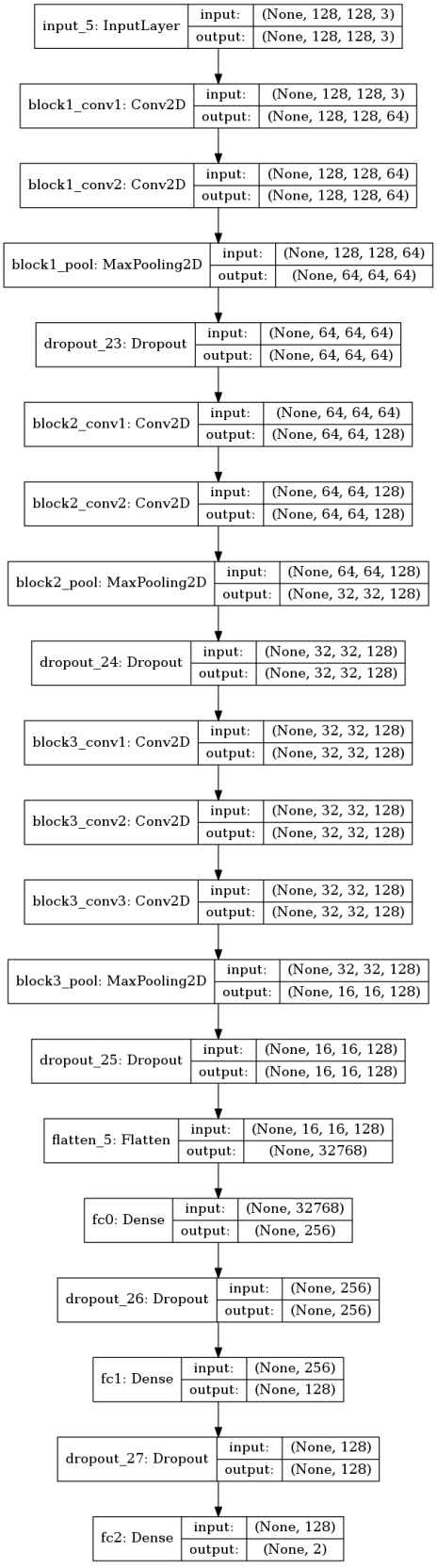
(a) **6conv** Structure, 256x128 top block layers.

(b) 6conv3 Structure, 256x128 top block layers.

Figure A.2: **6conv** and **6conv3** Variants Architectures

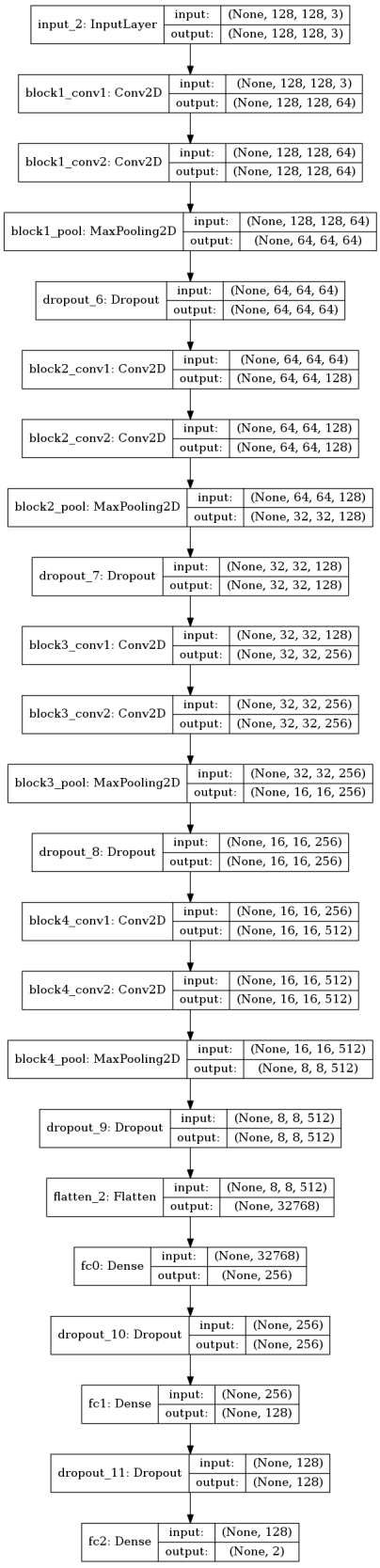


(a) 6conv3_s Structure, 256x128 top block layers.



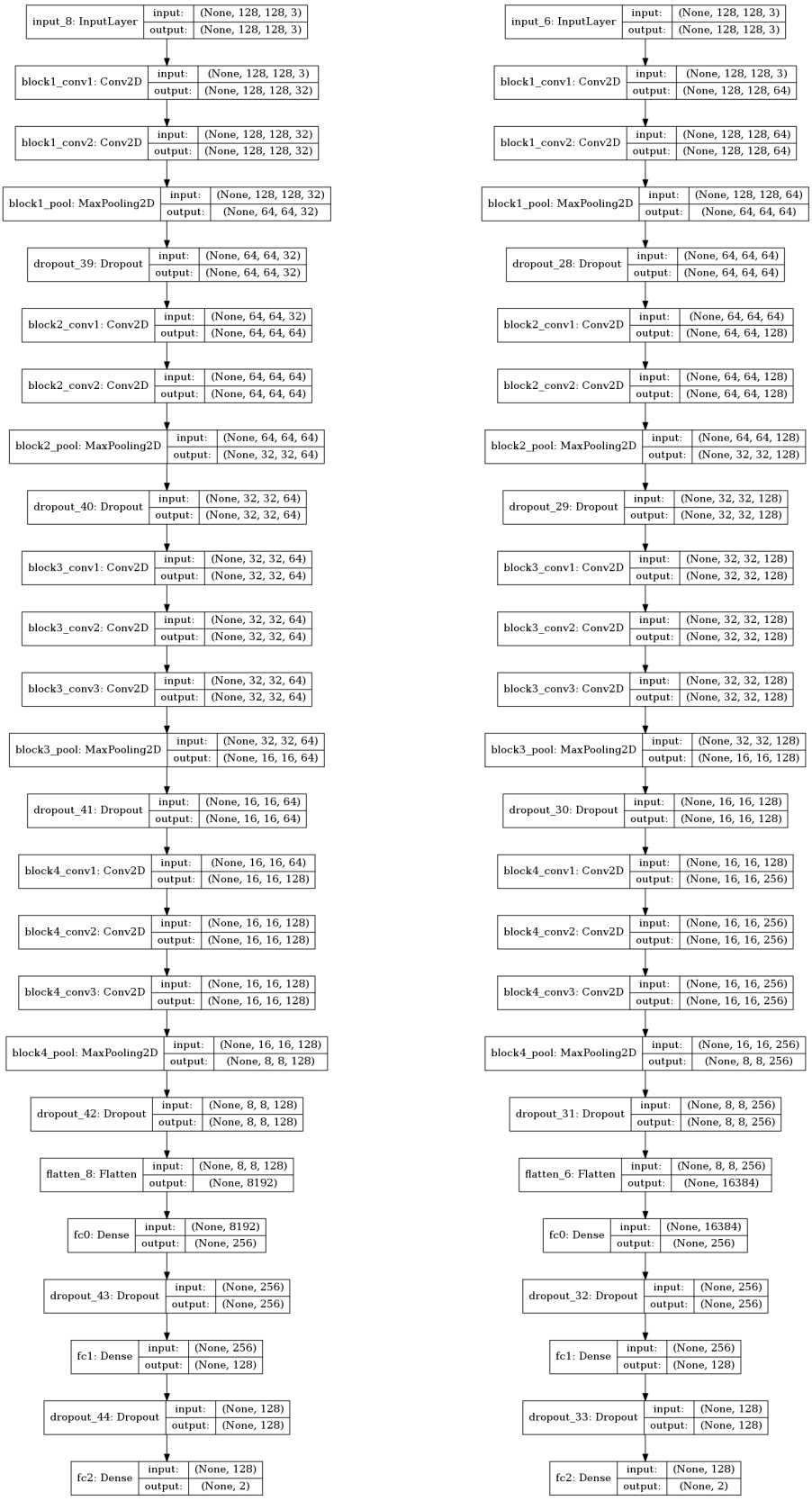
(b) 6conv3_l Structure, 256x128 top block layers.

Figure A.3: **6conv3_s** and **6conv3_l** Variants Architectures



(a) 8conv Structure, 256x128 top block layers. (b) 8conv3 Structure, 256x128 top block layers.

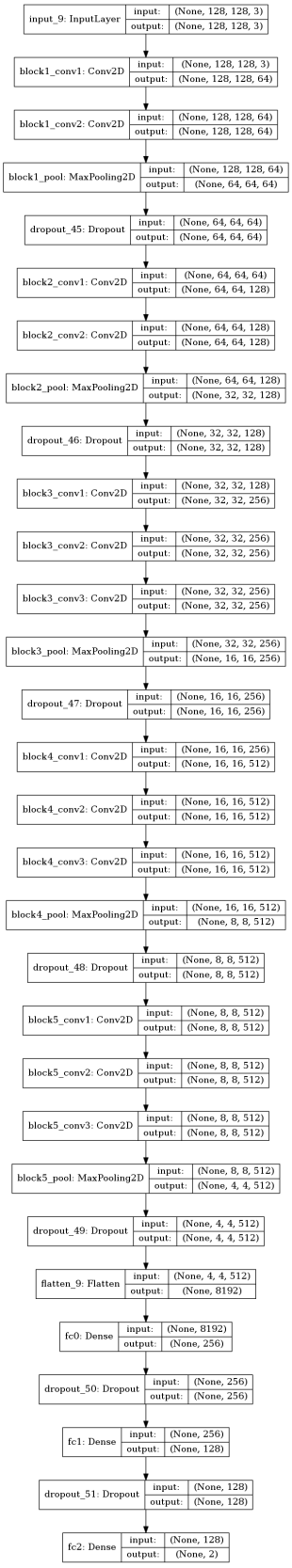
Figure A.4: **8conv** and **8conv3** Variants Architectures



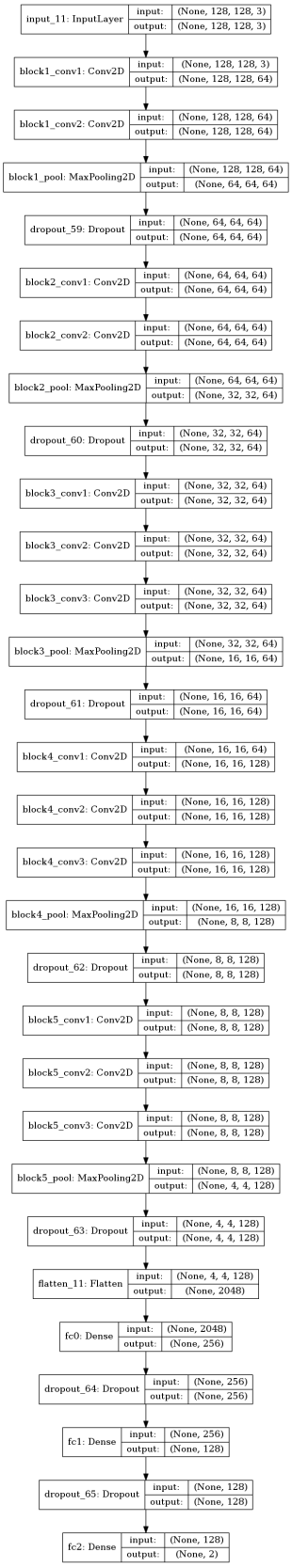
(a) 8conv3_s Structure, 256x128 top block layers.

(b) 8conv3_l Structure, 256x128 top block layers.

Figure A.5: **8conv3_s** and **8conv3_l** Variants Architectures



(a) 10conv3 Structure, 256x128 top block layers.



(b) 10conv3_s Structure, 256x128 top block layers.

Figure A.6: **10conv3** and **10conv3_s** Variants Architectures

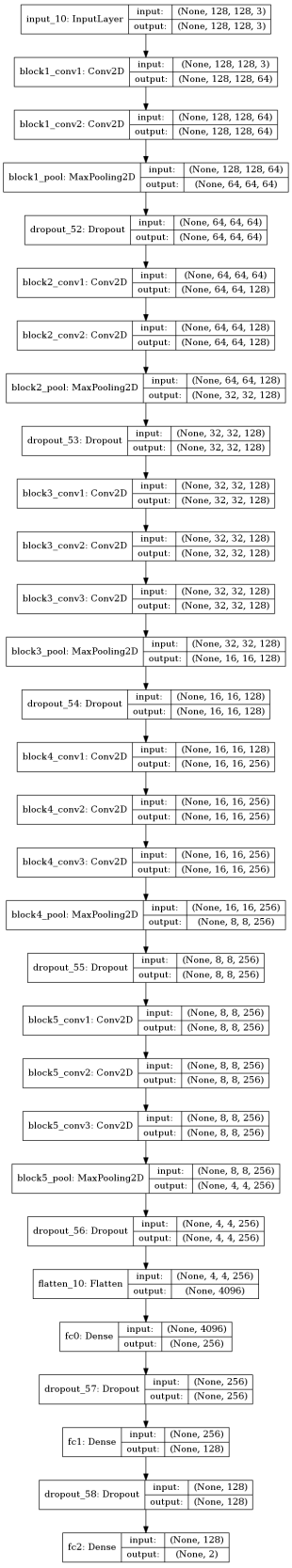


Figure A.7: 10conv3_l Structure, 256x128 top block layers.

A.1.2 Grid Search Parameters

A.1.2.1 Experiment 1 and 2: Movable and Rollable Objects

Grid search meta-parameters for experiment 1 and 2 :

```
# Define grid
grid_params['range_input_shape'] = [(128,128,3), (224,224,3)]
grid_params['range_lbl_shape'] = [(1,)]
grid_params['range_lbl_type'] = [int]
grid_params['range_last_layers_sizes'] = [[128, 32]]

grid_params['range_split_ratio'] = [n,0.5,0.7,0.9]
grid_params['range_nb_classes'] = [2]
grid_params['range_batch_size'] = [32]
grid_params['range_optimizers'] = [
    [
        {'type':'SGD', 'lr':0.001, 'decay':1e-6, 'nb_epoch':10},
        {'type':'SGD', 'lr':0.0005, 'decay':1e-6, 'nb_epoch':100},
        {'type':'SGD', 'lr':0.0001, 'decay':1e-6, 'nb_epoch':100}
    ]
]

grid_params['range_callbacks'] = [
    [
        None,
        [early_stopping, reduce_lr_on_plateau],
        [early_stopping, reduce_lr_on_plateau]
    ]
]

grid_params['range_fine_tuning'] = ['by_block']
grid_params['range_model_type'] = [
    dl_models.DL_MODEL_6CONV3, dl_models.DL_MODEL_6CONV3_S, dl_models.DL_MODEL_6CONV3_L,
    dl_models.DL_MODEL_8CONV3, dl_models.DL_MODEL_8CONV3_S, dl_models.DL_MODEL_8CONV3_L,
    dl_models.DL_MODEL_10CONV3, dl_models.DL_MODEL_10CONV3_S, dl_models.DL_MODEL_10CONV3_L,
    dl_models.DL_MODEL_VGG16_K
]

datagen = Image.ImageDataGenerator(
    rescale=1./255,
    rotation_range=90,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    vertical_flip=True)
grid_params['range_datagen'] = [datagen]

grid_params['nb_run'] = [5]
```

A.1.2.2 Experiment 4: No Pretraining

Grid search meta-parameters for experiment 4:

```
# Define grid
grid_params['range_input_shape'] = [(128, 128, 3), (224, 224, 3)]
grid_params['range_lbl_shape'] = [(1,)]
grid_params['range_lbl_type'] = [int]
grid_params['range_last_layers_sizes'] = [
    [64, 32], [128, 32], [128, 64], [256, 64], [256, 128], [512, 128], [512, 256],
    [256, 64, 32], [512, 128, 64]
]

grid_params['range_split_ratio'] = [0.3, 0.5, 0.7, 0.9]
grid_params['range_nb_classes'] = [2]
grid_params['range_batch_size'] = [32]
grid_params['range_optimizers'] = [
    [
        {'type': 'SGD', 'lr': 0.001, 'decay': 1e-6, 'nb_epoch': 30},
        {'type': 'SGD', 'lr': 0.0005, 'decay': 1e-6, 'nb_epoch': 100},
        {'type': 'SGD', 'lr': 0.0001, 'decay': 1e-6, 'nb_epoch': 100}
    ]
]

grid_params['range_callbacks'] = [
    [
        [early_stopping],
        [early_stopping, reduce_lr_on_plateau],
        [early_stopping, reduce_lr_on_plateau]
```

```
    ]
]

grid_params['range_pretrained_weights'] = [None]
grid_params['range_fine_tuning'] = ['by_block', 'whole']
grid_params['range_model_type'] = [
    # dl_models.DL_MODEL_VGG16,
    dl_models.DL_MODEL_6CONV3, dl_models.DL_MODEL_6CONV3_S, dl_models.DL_MODEL_6CONV3_L,
    dl_models.DL_MODEL_8CONV3, dl_models.DL_MODEL_8CONV3_S, dl_models.DL_MODEL_8CONV3_L,
    dl_models.DL_MODEL_10CONV3, dl_models.DL_MODEL_10CONV3_S, dl_models.DL_MODEL_10CONV3_L,
]

datagen = Image.ImageDataGenerator(
    rescale=1. / 255,
    rotation_range=90,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    vertical_flip=True)
grid_params['range_datagen'] = [datagen]

grid_params['nb_run'] = [2]
```