

ÉCOLE DOCTORALE SCIENCES ET MÉTIERS DE L'INGÉNIEUR
Laboratoire d'Ingénierie des Systèmes Physiques et Numériques -
Campus de Lille

THÈSE

présentée par : **François Hélénon**
soutenue le : **18 janvier 2022**

pour obtenir le grade de : **Docteur d'HESAM Université**

préparée au : **École Nationale Supérieure d'Arts et Métiers**
Spécialité : **Génie informatique, automatique et traitement du signal**

Architecture robotique et cognitive pour l'apprentissage de tâches en interaction avec l'humain. Une application pour la collaboration homme/robot dans l'Industrie 4.0.

THÈSE dirigée par :
M. Gibaru Olivier

et co-encadrée par :
M. Thiery Stéphane et M. Nyiri Éric

Jury

M. Rachid ALAMI, Directeur de recherche, LAAS, ANITI, Toulouse
M. Pedro NETO, Maître de conférences, HDR, CoRLuc, Université de Coimbra, Portugal
M. Gang ZHENG, Chargé de recherche, HDR, DEFROST, INRIA, Lille
Mme. Hélène CHANAL, Maître de conférences, HDR, SIGMA, Université de Clermont-Ferrand
M. Martin RIEDMILLER, Professeur des universités, Google DeepMind, Londres
M. Olivier GIBARU, Professeur des universités, LISPEN, Arts et Métiers, Lille
M. Éric NYIRI, Maître de conférences, LISPEN, Arts et Métiers, Lille
M. Stéphane THIERY, Maître de conférences, LISPEN, Arts et Métiers, Lille

Président
Rapporteur
Rapporteur
Examinatrice
Examinateur
Examinateur
Examinateur
Examinateur

Remerciements

Je tiens tout d'abord à remercier ceux qui m'ont permis d'intégrer cette thèse : mon directeur **Olivier Gibaru**, et mes co-encadrants **Stéphane Thiery** et **Éric Nyiri**. Merci pour la confiance que vous m'avez accordé, pour les apports matériels et pour vos retours avisés, parfois le week-end, dans la direction et la valorisation de ce travail de recherche. Vous m'avez aidé au fil du temps à prendre du recul et de la hauteur sur mon travail.

Par ailleurs, je tiens à remercier la région **Hauts-de-France** et **AMValor** qui ont cofinancé cette thèse et m'ont ainsi permis de m'y consacrer à plein temps.

Je remercie l'ensemble des membres de mon jury pour m'avoir fait l'honneur de s'être intéressé à ce projet. Merci à **Pedro Neto** pour m'avoir accueilli au CoRLuc à Coimbra, où j'ai pu être introduit aux robots collaboratifs, puis pour avoir accepté de rapporter ce travail de thèse avec **Gang Zheng**. Vos relectures attentives, vos retours précis et synthétiques dans vos rapports et au cours de ma soutenance montrent l'intérêt porté à l'ensemble des travaux. Je tiens à remercier **Rachid Alami** pour avoir accepté de présider mon jury, ainsi que **Martin Riedmiller** et **Hélène Chanal** pour avoir accepté d'examiner mon travail. Merci à tous pour vos retours bienveillants, curieux et encourageants.

J'ai une pensée pour les nombreuses personnes dont j'ai eu le plaisir de partager un petit bout de chemin. Sans exhaustivité, merci à : **Arthur** pour sa gentillesse et m'avoir accompagné avec succès pour danser le rock ; **Eddy** pour sa maîtrise du café et son aide, aussi bien dans l'accueil des nouveaux doctorants que pour le départ des futurs docteurs ; **Zein** pour son flegme et son sérieux ; **Michel** pour sa bienveillance et son abnégation ; **Vincent** et **Tiphaine** pour leurs chaleureux accueils dans leur appartement ; **Mathieu** que j'ai eu le plaisir de retrouver en thèse et pour de longues discussions ; **Marielle** pour son organisation, ses encouragements et pour m'avoir accompagné pour de nombreux footings ; **Marguerite** pour sa bonne humeur et son franc-parler ; **Martin** pour sa curiosité vis-à-vis

REMERCIEMENTS

de l'IA ; **Mohammad** pour son courage et sa force de caractère ; **Modhi** pour sa bienveillance et m'avoir montré que l'on peut être thésard et (très) sportif ; **Amélie** avec laquelle j'ai partagé de beaux instants musicaux à l'OUL ; **Sebastian** et ses talents de danseurs ; **Maxime** pour sa relecture de certains chapitres et qui garde la pêche malgré les caprices des robots et des réseaux de neurones ; **Floriane** pour ces courts, mais sympathiques moments passés au laboratoire ; **Dorian** pour sa gentillesse et sa rapide intégration dans l'équipe ; **Anthony** avec qui j'ai partagé la plateforme robotique et les mois de rédaction ; Merci à **Laurent** pour son aide et collaboration sur une partie des travaux de recherche présentés dans ce manuscrit ; Merci à **Lei** pour son ouverture d'esprit et sa volonté de partage avec l'ensemble du laboratoire (merci encore pour les dumplings !) ; Merci à **Olivier** (également un petit *fu-tum* pour **Grecia**) pour les ateliers cuisines, les cours d'investissements, et sa relecture de chapitres ; Merci à **Lamine** pour les couscous, sa positivité et son soutien sans faille aussi bien au cours de la thèse qu'en dehors ; ...

Je remercie les personnels de l'ENSAM et d'AMValor pour leur accueil et leur accompagnement divers au cours de ma thèse : **Sabine, Ludovic, Richard, Olivier T., Vincent D., Vincent I., Adrien, Estelle, Éloïse, ...**

Merci aux chefs, musiciens et aux chanteurs de l'OUL et du COUL, qui m'ont permis de m'évader musicalement pendant la thèse.

Une pensée à tous mes amis de l'I.N.D., de l'ENSAM et de Supélec et en particulier pour toute ma famille proche et éloignée, qui m'ont accompagnée dans cette complexe aventure. Merci à mes parents, **Fily et Alain**, à ma sœur et meilleure amie **Oriane**, à mes oncles, tantes, marraine, grands-parents, et toute ma famille, qui, où qu'ils soient aujourd'hui, m'ont appris que la vie est un combat et ont participé à façonner la personne que je suis.

Enfin, j'aimerais remercier l'ensemble des contributeurs et contributrices de l'Open Source. La réalisation technique de ce travail de thèse n'aurait pas été possible sans l'exploitation et l'intégration de ressources issues de programmes libres et ouverts.

Résumé

Des capacités d'interaction flexible et centrée sur l'humain, en robotique collaborative, est un aspect essentiel de l'industrie 4.0/5.0. Les robots collaboratifs peuvent désormais fournir une assistance dans de nombreuses tâches, contribuant ainsi à réduire les risques de troubles musculo-squelettiques pour les travailleurs humains. Cependant, le niveau de collaboration reste encore loin du niveau naturel entre deux collègues humains. En effet, la reconfiguration des robots collaboratifs manque encore de flexibilité et est souvent hors de portée du travailleur du quotidien, qui n'est ni un programmeur ni un expert en robotique. Un robot collaboratif idéal devrait devenir un Assistant Robotique Intelligent (SRA) capable d'adapter dynamiquement son comportement à la diversité de chaque situation, y compris les tâches, les changements d'environnement, les caractéristiques des travailleurs et leurs préférences. De telles exigences conduisent à un changement de paradigme dans la façon dont les robots collaboratifs sont programmés.

Tout au long de cette thèse, pour répondre aux spécifications d'un SRA, nous avons exploré la conception d'un prototype d'architecture cognitive autour de la notion d'Enseignement Robotisé en Interaction (IRL). L'agent robotique peut apprendre, en s'appuyant sur des connaissances antérieures, comment représenter et exécuter des tâches inconnues avec des capacités de généralisation, selon les préférences et les caractéristiques des travailleurs. L'apprentissage se fait tout au long de l'interaction, dans un cadre d'initiative mixte, de manière incrémentale, rapide et naturelle, par des personnes non expertes en programmation.

En nous inspirant d'approches complémentaires de la littérature en Intelligence Artificielle (IA) et en ITL, nous avons mis en évidence les avantages d'une architecture hybride, entrelaçant les approches symbolique et connexionniste en IA. Suivant les spécifications du SRA, nous avons choisi de développer un nouveau système cognitif basé sur des modèles de représentations relationnelles et l'intégration de modules d'apprentissage spécifiques basés sur l'apprentissage profond. En particulier, nous nous

sommes concentrés sur l'exploitation de représentations modulaires pour les comportements du SRA, intervenant pour le processus d'apprentissage délibératif et incrémental de l'agent. Cela a conduit à considérer les arbres de comportements réactifs au coeur du modèle de comportement de l'architecture. Cela permet d'apprendre des niveaux hiérarchisés de représentations, de la perception motrice du monde réel aux représentations symboliques abstraites.

Des validations expérimentales, avec de vrais robots collaboratifs, ont été effectuées tout au long de la thèse pour évaluer le comportement de l'actuel prototype d'architecture par rapport aux spécifications du SRA. Comme les tâches de manipulation sont courantes dans de nombreuses applications industrielles, nous avons choisi de concentrer ces validations expérimentales sur des scénarios de préhension planaires, orientés vers la tâche. Ceci a motivé le développement et l'intégration de modules d'apprentissage en IA basés sur des démonstrations humaines pour l'apprentissage de la préhension. À partir de quelques démonstrations, un humain peut enseigner rapidement et naturellement les emplacements autorisés et interdits, en fonction de la tâche et/ou de leurs propres préférences.

En outre, et en tant que perspectives d'intégration futures, nous discutons de la façon dont les techniques d'incertitude et d'estimation pour l'apprentissage profond pourraient être exploitées au coeur de l'architecture, pour les prédictions d'échec et pour l'apprentissage actif.

Abstract

Human-centric and flexible interaction in collaborative robotics is a key aspect of industry 4.0/5.0. Collaborative robots can now assist in many tasks, helping to reduce musculoskeletal disorders risks for human workers. However, the level of collaboration remains far from the natural one between two human coworkers. Indeed, reconfiguration of collaborative robots still lacks flexibility and is often out of reach of the everyday worker, who is neither a programmer nor a robotics expert. An ideal collaborative robot should become a Smart Robotic Assistant (SRA) that can adapt dynamically its behavior to the diversity of each situation, including tasks, environment changes, workers characteristics and their preferences. Such SRA requirements lead to a paradigm shift in the way collaborative robots are programmed.

Throughout this thesis, to fulfill SRA specifications, we have explored the design of a prototype of cognitive architecture around the notion of Interactive Robot Learning (IRL). The robotic agent can be taught, by leveraging prior knowledge, how to represent and carry out unknown tasks with generalization abilities, according to workers preferences and characteristics. Teaching is done throughout interactions, in a mixed-initiative setting, incrementally, and in a fast and natural way by non programming experts.

Taking inspiration from complementary AI and IRL paradigms found in the literature, we have highlighted the benefits of a hybrid architecture, interleaving symbolic and connectionist approaches. With SRA specifications in mind, we chose to develop a new cognitive system based on relational representations models and integration of specific learning modules based on deep learning. In particular, we have focused on exploiting modularity of behaviors representations for the agent deliberative and incremental learning process, which led to consider Behaviors Trees (BT) at the core of the behavior model. It helps to learn a hierarchical level of representations, from real world moto-perception to symbolic abstract representations.

ABSTRACT

Experimental validations, with real collaborative robots, were made throughout the thesis to assess the behavior of the current architecture prototype with respect to our SRA specifications. As manipulation tasks are common in many industrial applications, we chose to focus these experimental validations on planar, task-oriented grasping scenarios. This has motivated the development and integration of specific based AI learning modules, leveraging humans demonstrations for learning grasping. From a few demonstrations, workers can teach quickly and naturally authorized and prohibited locations concerning the task and/or their own preferences.

In addition and as future integration perspectives, we discuss how uncertainty and estimation techniques for deep learning could be leveraged in the core of the architecture, for failure predictions and active learning.

Contents

Remerciements	i
Résumé	iii
Abstract	v
List of tables	xvi
List of figures	xxiii
Acronyms	xxv
1 Introduction	1
1.1 Motivations: challenges in fully reconfigurable robotics for industry	2
1.1.1 Classical industrial robots: one task specialist for robots programmer experts .	2
1.1.2 Collaborative industrial robots at the dawn of a new social industrial robotic context	3
1.2 Positioning in the broader cognitive robotics field	4
1.2.1 What is a collaborative artificial intelligence ?	5
1.2.2 Human in the loop for flexible interactive robot learning	7
1.3 Towards a smart robotic assistant	9
1.3.1 Objectives	9

CONTENTS

1.3.2	Contributions and thesis organization	10
2	State of the art on cognitive systems	11
2.1	Connectionist and symbolic approaches	12
2.1.1	System understandability	12
2.1.2	Connectionism	13
2.1.3	Symbolic	15
2.1.4	Hybrid	16
2.2	Main building blocks of a cognitive architecture	17
2.2.1	Ontology as an explainable structure for components interoperability	17
2.2.2	Behavior model	17
2.2.3	Teaching complex behaviors to robots : Interactive robot learning	23
2.3	Conclusion	25
3	Design of a cognitive architecture for Industry 4.0	29
3.1	Base ontology for our ITL/IRL	30
3.1.1	Robotic Agent as a goal driven agent	31
3.1.2	Environment representation	31
3.1.3	Utterances	33
3.1.4	Skills and actions primitive	33
3.2	Hierarchical, modular representations	35
3.2.1	Semantic declarative memory	36
3.2.2	Working memory	36
3.2.3	Episodic memory	37
3.2.4	Procedural memory	37
3.2.5	Perceptual memory	38
3.3	IRL process with preferences	39

3.4	Conclusion	41
4	Complementary ML approaches for IRL on planar grasping use cases	43
4.1	Learning planar grasping	45
4.1.1	Theoretical general formulation	45
4.1.2	Some common techniques	46
4.1.3	Planar grasping formulation	49
4.1.4	Base approaches in deep learning	49
4.2	Learning autonomously bin picking	51
4.2.1	Bin picking module	51
4.2.2	Methodology	53
4.2.3	Experimental results	55
4.2.4	Module conclusion	55
4.3	Learning grasping location affordance from demonstration	55
4.3.1	Task oriented grasping	56
4.3.2	Methodology	58
4.3.3	Experiments results	61
4.3.4	Module conclusion	69
4.4	Conclusion	69
5	Learning under uncertainty	71
5.1	What does uncertainty means ?	73
5.1.1	Stochasticity and data shift	73
5.1.2	Aleatoric uncertainty τ_a	79
5.1.3	Epistemic uncertainty τ_e	80
5.1.4	Total uncertainty τ	80
5.2	Uncertainty estimation methods	81

CONTENTS

5.2.1	Bayesian and variational inference methods	82
5.2.2	Deep ensemble methods	83
5.2.3	Distance aware uncertainty model	85
5.2.4	External measure	88
5.2.5	Conclusion on uncertainty review in deep learning	90
5.3	Uncertainty for decision-making	94
5.3.1	Calibration and sharpness	94
5.3.2	Performance metrics of a system	97
5.4	Active learning setting	100
5.4.1	Uncertainty for active learning setting	100
5.4.2	General active learning process for ITL	101
5.4.3	Uncertainty aware behavior model	103
5.5	Conclusion	106
6	Implementation and validation on a planar pick and place learning task	107
6.1	Choices of sensors for IRL perception and interaction	108
6.1.1	Non-verbal communication	109
6.1.2	Verbal communication	113
6.2	Perceptual and acting modules integration	117
6.2.1	Speech recognition and understanding	117
6.2.2	Prior object segmentation and tracking	118
6.2.3	Joints pose estimation for non verbal interaction	118
6.2.4	Location affordance learning	118
6.2.5	Acting	120
6.3	Validation scenario	120
6.4	Conclusion	124

7	Conclusion and perspectives	127
7.1	Main points	128
7.2	Integration of an uncertainty aware grasping module	130
7.2.1	Grasping under uncertainty	130
7.2.2	Trustnet	131
7.2.3	Extension for active learning by demonstration	133
7.3	Long terms perspectives	136
7.3.1	Continuous active learning setting	136
7.3.2	Reasoning about specific uncertainties	136
7.3.3	Multimodal fusion	137
7.3.4	Improve preferences generalization	139
7.3.5	Improve behavior models in terms of learning and representations	140
7.4	General conclusion	144
A	Evaluation of grasping quality and uncertainty	147
A.1	Jacquard metric	147
A.2	GraspNet/Trustnet experimental details	147
A.2.1	Implémentation details	147
A.2.2	TrustNet improvement study	151
B	Résumé étendu en français	155
B.1	Introduction: vers un Assistant Robotique Intelligent	155
B.1.1	Motivations	155
B.1.2	Positionnement	156
B.1.3	Objectifs	157
B.1.4	Contributions et organisation	158
B.2	Conception d'une architecture cognitive inspirée de l'état de l'art	159

CONTENTS

B.2.1	Approches connexionnistes et symboliques	160
B.2.2	Choix du modèle de comportements	161
B.2.3	Processus décisionnel du SRA prenant en compte les préférences	165
B.2.4	Conclusion	165
B.3	Approche ML pour la préhension planaire	166
B.3.1	Apprentissage autonome du devracage	166
B.3.2	Apprendre l'affordance de la position de saisie à partir de quelques démonstrations	169
B.4	Mise en oeuvre et validation sur une tâche d'apprentissage de type "prendre et placer"	173
B.4.1	Choix des capteurs pour la perception et l'interaction SRA	174
B.4.2	Intégration des modules de perception et d'action	175
B.4.3	Scénario de validation	175
B.4.4	Conclusion	178
B.5	Vers plus de flexibilités	178
B.5.1	Obtenir un niveau d'incertitude	179
B.5.2	Incertitude dans la prise de décision	180
B.5.3	Apprentissage actif	182
B.5.4	Conclusion	184
B.6	Conclusion générale	185
References		187
Chapter 1:	Introduction	187
Chapter 2:	State of the art on cognitive systems	189
Chapter 3:	Design of a cognitive architecture for Industry 4.0	196
Chapter 4:	Complementary ML approaches for IRL on planar grasping use cases	198
Chapter 5:	Learning under uncertainty	202
Chapter 6 :	Implementation and validation on a planar pick and place learning task	206

CONTENTS

Chapter 7: Conclusion and perspectives	209
Résumé étendu en français	211

CONTENTS

List of Tables

1.1	Paradigm shift in industrial robotics	4
2.1	Comparison of properties exhibited in cognitive and interactive robotics learning systems.	26
3.1	Control flow and action nodes in the standard behavior tree framework	35
4.1	Summary of existing task oriented grasping works	57
4.2	Results of our grasping test, percentage (number) of good grasps over the 36 unseen positions	63
4.3	Comparing grasping results with only one authorised area and with both authorised/prohibited areas (in parenthesis)	65
4.4	Influence of the weighted L_2 loss. Training was made with a set of 3 demonstrations.	66
4.5	Ability of the Network to generalize to an <i>unreferenced</i> similar object. The network was trained after demonstrations on a <i>referenced</i> object and the test was performed on <i>unreferenced</i> similar objects. In parenthesis, results obtained in section 4.3.3-II are recalled.	67
4.6	Grasping accuracy results for groups of similar objects	69
5.1	Contingency table with uncertainty measure τ . Upper a threshold uncertainty, an alarm is triggered, for instance in an active learning framework.	97
6.1	Summary table of sensors and perception modalities with qualitative comparison. Green sensors were chosen for the validation and the implementation of the architecture	116

LIST OF TABLES

6.2 Overview of prior knowledge 122

6.3 Synthesis of what will be learned during the incremental interactive learning process of the unknown task to give 122

6.4 Detail of the learning process during the incremental interactive learning process of the unknown task to give 123

7.1 Pearl’s Causal Hierarchy, adapted from Table 1.1 [29]. Appearing correspondences within the IRL architecture design. 144

A.1 Accuracy for various *GraspNet* architectures and learning scheme 149

A.2 Uncertainty metric performance for grasping cornell|jacquard datasets 151

B.1 Flux de contrôle et noeuds d’exécution dans le cadre standard des arbres de comportement 164

B.2 Aperçu des connaissances *a priori* 176

B.3 Synthèse de ce qui sera appris au cours du processus d’apprentissage interactif incrémentale de la tâche inconnue donner 176

B.4 Détail du processus d’apprentissage pendant le processus d’apprentissage interactif incrémentiel de la tâche inconnue ”donner” 177

B.5 Tableau de contingence avec mesure d’incertitude τ . Au-delà d’un seuil d’incertitude τ_{thresh} , une alarme est déclenchée, par exemple dans un cadre d’apprentissage actif. . . 180

List of Figures

1.1	Evolution of industrial robotics.	2
1.2	This figure was reproduced based on Figure 1 from [5] (Creative Commons Attribution 4.0 License). Based on psychological findings, it depicts general landmarks of how infants incrementally learn to build a world model during their early cognitive development.	6
1.3	Image taken at "Citadelle" park in Lille (59000), France.	8
2.1	General hybrid architecture design	13
2.2	Robotics architecture can be represented through three layers of control.	18
3.1	High-level overview of the architecture. The architecture consists of perceptual modules based on connectionist approaches, symbolic relational representations, and a deliberative process for interactive robot learning with human. The IRL process consists of two interleaved paths (a plain path with circled number and a dashed path with boxed number) which are described in section 3.3.	30
3.2	Base ontology example overview.	31
3.3	Base skill model	35
3.4	Example of a semantic memory database derived from the ontology	36
3.5	Example of the working memory derived from the ontology	37
3.6	Transfer learning in deep neural networks is leveraged to allow fast reconfiguration and resource management. To simplify the figure, we represent here a dense network but any kind of neural networks layers can be used if relevant for the sensory modality.	38

LIST OF FIGURES

3.7	It is the same figure as figure 3.1 and is reproduced here for reading convenience. . . .	40
3.8	The failure handling process triggers interactive learning of symbolic or perceptual representations.	41
4.1	Reinforcement learning base description	48
4.2	Grasping parameters	49
4.3	<i>GraspNet</i> architecture	50
4.4	Example of synergy between pushing and gripping. A pile of objects is presented none of which can be retrieved by direct grasping (a). The robot will first push the pile (b) and then separate the objects (b) and then grab an isolated object (c).	52
4.5	Hardware pipeline of the algorithm	53
4.6	Pipeline of the algorithm	54
4.7	First, for a new object, the system learns from operator’s demonstration. After few minutes of training, the system will be able to retrieve the demonstrated area on a depthmap.	56
4.8	Data capture and data augmentation pipeline.	59
4.9	Overview of our CNN pipeline.	60
4.10	Objects used for our experiment, with the name of grasping areas. Green colour (resp red colour) denotes authorised (resp prohibited) grasping area. The yellow colour is used to illustrate authorised/prohibited area and vice versa, depending on the task (for example the accessibility of the screwing operation). For the pliers, two different authorised areas are tested separately	63
4.11	Different segmentation quality results. The top images (cups) were trained showing one authorised grasping area (handle). The bottom images (socket wrench) were trained showing one authorised (handle) and one prohibited (head) grasping area. The left column shows bad segmentation resulting in a bad grasping decision, the middle one shows average segmentation, and the right one shows good segmentation of the object. For the purpose of illustration, outputs were reoriented and resized. Those outputs were obtained using networks trained on 3 demonstrations.	64

LIST OF FIGURES

4.12 Illustration of grasping results with respect to the number of demonstrations 65

4.13 Qualitative comparison between pipeline output. Networks were trained using 3 demonstrations 66

4.14 Similar objects used to test our algorithm generalization abilities 67

4.15 Bulbs grasping affordance pixel wise representations. From left to right : *referenced* bulb on which the training was done, bulb 1, and bulb 2. 67

4.16 The three groups of similar objects used in our test. 68

5.1 Behaviors of deep learning classification models given the two moons distribution dataset with an additional OOD cluster, with and without uncertainty aware leveraged techniques. 78

5.2 Two moons classification with an uncertainty aware model (here SNGP). We illustrate the aleatoric uncertainty τ_a and the learned epistemic uncertainty τ_e given the two moons dataset. 79

5.3 Two moons classification: evolution of prediction (top) and uncertainty (bottom) with time (learning epochs) 81

5.4 Bayesian methods for uncertainty estimation of deep neural networks 84

5.5 Ensemble methods for uncertainty estimation of deep neural networks 86

5.6 Illustration of the general principle of distance aware based model. SNGP is specific case. Plotting does not come from real data and serve just as an illustration. 88

5.7 Illustration of temperature scaling against standard classification 89

5.8 External measures of uncertainty estimation of deep neural networks 90

5.9 Classification probability with uncertainty for different uncertainty metrics τ 92

5.10 Regression task with Monte Carlo Dropout 93

5.11 Predicted uncertainty inference at the last layer 93

5.12 Estimating uncertainty using predictive uncertainty of the ensemble 94

5.13 Confidence histogram illustrating ECE metrics applied to the predictive uncertainty of an uncalibrated classification network based on EfficientNetb0 [36], pretrained on ImageNet and fine-tuned on cifar100 with coarse label (20 different classes). 96

LIST OF FIGURES

5.14 ROC curve. 98

5.15 Illustration of τ_{thresh} usage to limit wrong actions 99

5.16 Based on the ROC curve, one can choose a threshold to balance the trade-off between the amount of true and false positive. 100

5.17 General pipeline for active learning 103

5.18 General extension of belief behavior trees for condition and action nodes for uncertainty propagation. Dotted arrow represent what is returned by the node after its termination. 105

5.19 The IRL agent can exploit specific BBTs structure to handle uncertainty provided by lower level modules and the belief state in working memory. 106

6.1 Details view of the architecture in terms of integrated modules. 108

6.2 Learning authorised and prohibited grasping location : one demonstration (couple input I, label L) and data augmentation pipeline 119

6.3 Convolutional Neural Network (CNN) pipeline and prediction illustration. The approach used is a regression one, where CNN outputs a pixel-wise affordance map (pixels values are between -1 and 1). 119

6.4 The IRL agent can dynamically adapt its affordance prediction according to the interacting human as shown here on a wrench example. On the left, H_1 has taught the IRL agent to grasp the wrench by the head. On the right, H_2 has taught the IRL agent to grasp the wrench by the tail. The agent can then dynamically choose the preference according to the interacting human. 124

7.1 *TrustNet* architecture 132

7.2 Histogram of τ for good (green) and bad (red) predictions for different uncertainty methods on Cornell grasping datasets. Bad predictions are represented by negative occurrences 133

7.3 Graspnet and TrustNet integration can be used in a task oriented setting for active learning by demonstration. 134

LIST OF FIGURES

7.4 Possible integration of the module with BBTs. Uncertainty can be propagated backward and t leveraged by the IRL agent for active learning. 135

7.5 Integration and learning pipeline of the TrustNet/GraspNet module with the higher level processes of the IRL architecture. 135

7.6 Illustration of τ_{thresh} usage to prevent wrong actions, given the uncertainty τ 137

7.7 Different Bayesian Networks graphical models can represent the same data. But not all graphs represent *causal* relations. Gray variables are observed variables, white one is a hidden variable that could influence object visibility. 144

A.1 ROC and S-FPR curves for different uncertainty metrics τ 150

A.2 Histogram of τ for good (green) and bad (red) predictions for different uncertainty methods on Cornell grasping datasets. Bad predictions are represented by negative occurrences 151

A.3 FPR-95%-TPR evolution for different γ . For some γ , we also plot similar histograms as presented in the qualitative analysis. 153

A.4 Performances comparaison between *Ensemble Weighted Trustnet* and the original *Trust-Net* (Cornell dataset) 153

B.1 Evolution des robots industriels. 156

B.2 Aperçu de haut niveau de l’architecture. L’architecture se compose de modules perceptifs basés sur des approches connexionnistes, de représentations relationnelles symboliques et d’un processus délibératif pour l’apprentissage en interaction avec l’humain. . . 159

B.3 Conception générale de l’architecture hybride 160

B.4 Exemple haut niveau d’une ontologie 161

B.5 L’architecture robotique cognitive peut être représentée par trois couches de contrôle. 162

B.6 Modèle de base pour une compétence 164

B.7 Le processus de résolution des échecs déclenche l’apprentissage de représentations symboliques ou perceptuelles grâce à l’interaction avec l’humain. 165

LIST OF FIGURES

B.8 Exemple de synergie entre actions de poussée et de préhension. On présente une pile d'objets dont aucun ne peut être récupéré par préhension directe (a). Le robot va d'abord pousser la pile pour séparer des objets (b) afin de saisir un premier objet isolé (c).	167
B.9 Dispositif expérimental	168
B.10 Architecture de l'algorithme	169
B.11 Tout d'abord, pour un nouvel objet, le système apprend à partir de la démonstration de l'opérateur. Après quelques minutes d'entraînement, le système est capable de retrouver la zone désignée sur une carte de profondeur.	170
B.12 Capture de données et augmentation	171
B.13 Aperçu de notre pipeline CNN.	171
B.14 Objets utilisés pour notre expérience, avec le nom des zones de préhension. La couleur verte (resp. la couleur rouge) indique la zone de préhension autorisée (resp. interdite). La couleur jaune est utilisée pour illustrer la zone autorisée/interdite et vice versa, en fonction de la tâche. Pour les pinces, deux zones autorisées différentes sont testées séparément	172
B.15 Illustration des résultats de préhension suivant le nombre de démonstrations	173
B.16 Vue détaillée de l'architecture en termes de modules intégrés.	174
B.17 L'agent SRA peut adapter dynamiquement sa prédiction d'affordance en fonction de l'humain avec lequel il interagit, comme le montre l'exemple de la clé. À gauche, H_1 a appris à l'agent SRA à saisir la clé par la tête. À droite, H_2 a appris à l'agent à saisir la clé par le manche. En fonction de l'opérateur avec lequel le SRA interagit, le robot apporte alors la clé dans sa main dominante et selon sa préférence.	178
B.18 Illustration de l'utilisation de τ_{thresh} pour limiter les mauvaises actions	181
B.19 Processus général pour l'apprentissage actif	182
B.20 L'agent SRA peut exploiter la structure spécifique des BBTs pour gérer l'incertitude fournie par les modules de niveau inférieur	183

LIST OF FIGURES

B.21 Illustration de l'utilisation de τ_{thresh} pour éviter les mauvaises actions, compte tenu de l'incertitude τ 184

LIST OF FIGURES

Acronyms

ACC Accuracy.

AGI Artificial General Intelligence.

AI Artificial Intelligence.

AS Action Script.

AUROC Area under the ROC Curve.

BAN Bayesian Networks.

BBTs Belief Behavior Trees.

BN Behavior Networks.

BT Behavior Trees.

CAD Computer Aided Design.

CNN Convolutional Neural Networks.

CP Cram Plan.

DQN Deep Q-Network.

ECE Expected Calibration Error.

FN False Negative.

FP False Positive.

FPR False Positive Rate.

FPR- α %-TPR False Positive Rate for a given percentage α % of the True Positive Rate.

FSM Finite State Machine.

GP Gaussian Process.

HFSM Hierarchical Finite State Machine.

HRI Human Robot Interaction.

HRL Hierarchical Reinforcement Learning.

HTN Hierarchical Task Network.

IID Independent and Identically Distributed.

IRL Interactive Robot Learning.

ITL Interactive Task Learner.

KPI Key Point Indicators.

Lfd Learning From Demonstrations.

MCE Maximum Calibration Error.

MDP Markov Decision Process.

ML Machine Learning.

MSE Mean Square Error.

NLP Natural Language Processing.

NPC Non Playable Character.

NPV Negative Predictive Value.

OOD Out Of Distribution.

PCA Principal Component Analysis.

PNP Petri Net Plan.

POS Part of Speech.

PR Percept-Response.

PRS Procedural Reasoning System.

RAE Refinement Acting Engine.

RAP Relational Activity Processes.

RL Reinforcement Learning.

ROC Receiver Operating Characteristic.

ROS Robotic Operating System.

S- β %-FPR Compute the success rate β % of the False Positive Rate.

SCM Structural Causal Model.

SGD Stochastic Gradient Descent.

SHP Shared Plans.

SNGP Spectral-normalized Neural Gaussian Process.

SPR Soar Production Rules.

SRA Smart Robotic Agent.

STT Speech To Text.

SWA Stochastic Weight Averaging.

SWAG Stochastic Weight Averaging-Gaussian.

Acronyms

TDL Task Description Language.

TN True Negative.

TP True Positive.

TPR True Positive Rate or sensitivity.

Chapter 1

Introduction

Contents

1.1 Motivations: challenges in fully reconfigurable robotics for industry	2
1.1.1 Classical industrial robots: one task specialist for robots programmer experts .	2
1.1.2 Collaborative industrial robots at the dawn of a new social industrial robotic context	3
1.2 Positioning in the broader cognitive robotics field	4
1.2.1 What is a collaborative artificial intelligence ?	5
1.2.2 Human in the loop for flexible interactive robot learning	7
1.3 Towards a smart robotic assistant	9
1.3.1 Objectives	9
1.3.2 Contributions and thesis organization	10

In recent years, industrial robots have left their cages to become more collaborative thanks to better sensors and higher level programming libraries. Yet, in real world scenarios, flexibility and interaction abilities of robots remains far from the natural interaction expected between two human co-workers. This new paradigm requires both better hardware and software. Particularly for the latter, artificial intelligence is playing an increasing role to cope with environment variability and complexity of human interactions. This chapter introduces the global context of the thesis. Section 1.1 first motivates this work by drawing through an industrial collaborative perspective, the required paradigm shift to build a Smart Robotic Assistant (SRA). Then, we introduce in section 1.2 the field of cognitive robots. We emphasise the need of a cognitive architecture and the integration of several AI paradigms as the basis of an SRA that learns in interaction with humans. This path has led to contributions to Learning From Humans Demonstrations and Interactive Robot Learning, described in section 1.3.

1.1 Motivations: challenges in fully reconfigurable robotics for industry

In the ideal Industry 4.0, robots are expected to work hand to hand with humans. Collaborative robots will have a predominant place, but humans will be at the center. It is therefore up to the robot to adapt to diversity: each person, each task, each situation. This will be the generation of Smart Robot Assistant (SRA). The road is still long and requires a paradigm shift and the development of solutions allowing the robot to acquire all these adaptive capabilities in a way that is both generalizable and explainable.

1.1.1 Classical industrial robots: one task specialist for robots programmer experts

The idea of building complex mechanical systems that can mimic animals or humans capabilities, inspired humans centuries before the first computer. The first automata, which come from the ancient Greek mythological word *automaton*, were purely mechanical animated devices. For instance in 1734, Jacques de Vaucanson built one of the first documented biomechanical inspired automaton: *the Digesting Duck*. It was a purely mechanical device, able to quack, to flap its wing, to process food and that was programmed by a set of well-designed cams. These mechanical devices of course were too limited to be more than curiosities, but they can be seen as precursors to modern robotics.

According to a brief historical overview in [1], industrial robotics evolved through four generations which have been summarized in the timeline Figure 1.1.

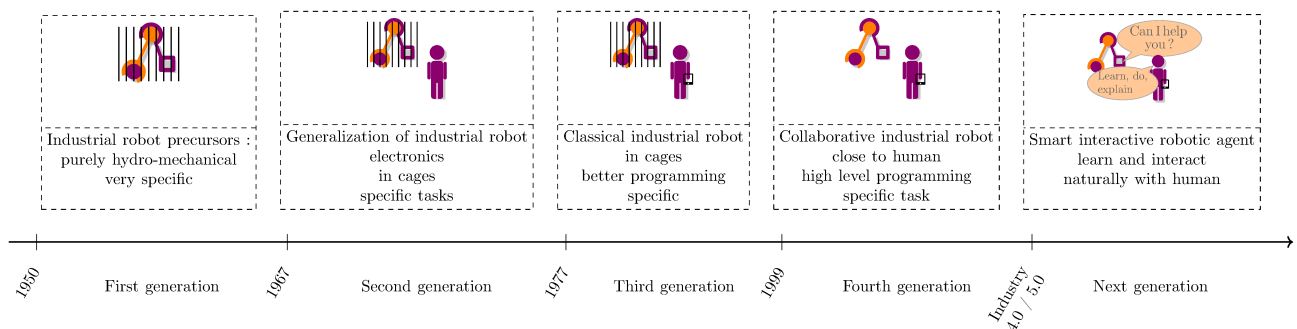


Figure 1.1: Evolution of industrial robotics.

A few decades after the Industrial Revolution, in the 1950s, the idea of automatizing industrial processes with mechanical manipulators emerged and gave birth to the first generation of industrial robots. In 1961, the first *true* commercialized industrial robot, *Unimate*, was introduced in *General*

1.1. MOTIVATIONS: CHALLENGES IN FULLY RECONFIGURABLE ROBOTICS FOR INDUSTRY

Motors factories. Many competitors followed but as purely hydro-mechanical devices, they were specifically designed for a task. Therefore, despite many successes, their use was defined at the time of integration as welding, pick and place or painting, and this was extremely difficult to modify.

With the development of electronics and computer science, the second generation of industrial robots appeared (1968-1977). They consisted of basic programmable machines leveraging the computing power of microprocessors and programmable logic controllers to accomplish more complex tasks. However, here again, because of robots diversity and low-level programming, changing from a task to another was requiring a high-level of expertise, with vendor-specific language to update the controller. Several improvements occur in these decades with the development of 6DoF manipulators and the use of embedded sensors to measure joints positions and velocities. Nevertheless, because of their lack of versatility, robots early successes were mostly focused on highly specific and redundant tasks.

During 1978-1999, the third generation of robots began to leverage human interaction interfaces such as pre-programmed vision or voice commands. High-level command libraries were also developed allowing more high-level control, such as point to point motion planning, in an offline or online context with a computer. The use of more complex sensors such as cameras allowed to bring more adaptability in well-controlled environment for mass production or where high-precision was needed such as in automotive or spatial industry. These robots had in common that they were dedicated to specific tasks, not very adaptable to changes and dangerous, therefore unsuited to interactions with humans.

A new evolution has led to the fourth generation of industrial robots started from the year 2000 and was marked by the release of a new kind of robot: collaborative industrial robots also called cobots.

1.1.2 Collaborative industrial robots at the dawn of a new social industrial robotic context

With the development of industrial collaborative robots has emerged a new paradigm shift in the way workers use and interact with them. While classical robots are used to be isolated from humans and tailored to very narrow and repetitive tasks in a predictable environment, collaborative robots evolve close or even in contact with humans. Their commercialization is a first step towards effective human/robot interaction as they are now able to go outside of their cages, thanks to better sensors and more accessible high-level programming libraries. For instance, force sensors at the different joints of the robot can help operators to program a desired trajectory with hand guiding [2] and make them

safe for human interactions. This makes it easier to program or reprogram robots in a simple industrial setting. It also provides assistance in hard and tedious tasks that were previously beyond the reach of robots. Hence, collaborative robots have social benefits, like reducing physical work-load, and thus preventing many musculoskeletal disorders (MSDs) risks among human workers in a much wider set of tasks. However, to collaborate with human workers in everyday tasks and to become smart robotics assistants, collaborative robots have to be endowed with much higher abilities. The authors of [3] surveyed industry on the general requirements for cobots. Although limited to Finnish industry, their questions and analysis are relevant to industry in general. We compare in table 1.1 some of the main asked requirements of cobots against classical robots.

Overall these results shows that cobots are likely to be used in companies of all kinds. Their integration is still difficult because the environment, tasks and human agents are not taken into account and are difficult to predict. Moreover, as the tasks can change, the robots need to be reconfigured while the resources of expert robotics programmers remain scarce. In that context, robots manufacturers and integrators have developed user interfaces with high-level libraries that non-experts can quickly learn in order to program specific tasks. Yet these interfaces are far from natural communications between two human coworkers and programmed tasks are often too specialized and not transferable to other tasks.

Table 1.1: Paradigm shift in industrial robotics

Classical industrial robots	Cobots requirements from [3]
highly repetitive, usually only programmed for one specific task	multiple reconfigurable tasks by non-expert users
well structured and predictable environment	unstructured environment
works on cages far from humans or other unpredictable agents	works with or even in contact with other agents such as humans : need of natural HRI (vision, speech,...) and better allocation procedures
robots are unsafe	safer but can still be dangerous if bad behaviors (need of explainable and interpretable behaviors)

1.2 Positioning in the broader cognitive robotics field

This substantial paradigm shift requires a cognitive system defining a Smart Robotic Assistant (SRA) [4]. It should be able to interpret and react to human natural interactions for incremental learning. This incremental learning should improve and leverage a knowledge base of modular skills

that can be used, composed and transferred to a broad set of tasks, with adaptation to individual preferences and characteristics. These abilities should be integrated in a decision-making process which should be made as explainable as possible for the non programming experts, with high-level, trustful explanations. Such a SRA could have a great impact for the next generation of collaborative industrial robots.

To handle such complexity in a meaningful and understandable way, we need to implement an artificial agent by leveraging several artificial intelligence and human robot interaction paradigms.

1.2.1 What is a collaborative artificial intelligence ?

Broad understanding of cognitive abilities: Defining Artificial Intelligence (AI) is a complex task with many scientific, technical, philosophical and ethical ramifications, as there is actually no consensus on what "Intelligence" is. Yet, progresses has been made in different sub-fields of AI and expectations are higher and higher. This is especially true with robots, which are embodied agents as they act in the same real-world environment as us, humans. In this setting, AI field usually distinguishes several levels of autonomy: narrow AI, broad AI, animal and human level AI and General AI (AGI). Narrow AI agents learn to solve tasks and generalize only in a very narrow setting close to training. Broad AI agents are able to leverage prior knowledge in tasks far from training but still with domain specialization. Animals level AI and especially human level AI can generalize and adapt quickly across domains with very few data examples. General AI (or strong AI) is sometimes referred to the human level AI or superhuman level AI in all tasks that could be done by humans. AGI is one of the ultimate dream goal of some AI research. It has fueled several fantasies since birth of AI, such as the concept of *singularity*, where an AGI could achieve consciousness, continuously improving until creating knowledge and technologies beyond human understanding.

A measure of intelligence can be seen as the ability to learn how to accomplish tasks and to leverage what was learned to new target tasks. Another interesting property of human intelligence is that knowledge is built incrementally throughout their lives. Generalization and adaptation could take root in this ability of autonomous continuous learning capabilities and from exchanging information with others. It seems that this is done by leveraging a certain level of prior knowledge and common sense knowledge built at the early stage of life. For instance, cognitive sciences and neurosciences have shown that during their cognitive development, infants progressively build more and more complex

1.2. POSITIONING IN THE BROADER COGNITIVE ROBOTICS FIELD

knowledge of the world. Figure 1.2 taken from [5], illustrates the emergence of physical concepts that babies learn, such as visual properties and acting abilities in the real-world. However, the amount of required prior knowledge in this process is still unknown. This interaction takes place in a social context which allows infants and other agents to exchange information to build a shared, structured and abstract representation that is usually referred as common sense knowledge. While human level AI is probably still far from reach, this encourages the development of broad AI agents which learn throughout their lives, in an online, incremental and interactive way as we do. The implementation of such robotic agents could be facilitated by the design of cognitive robotic architectures.

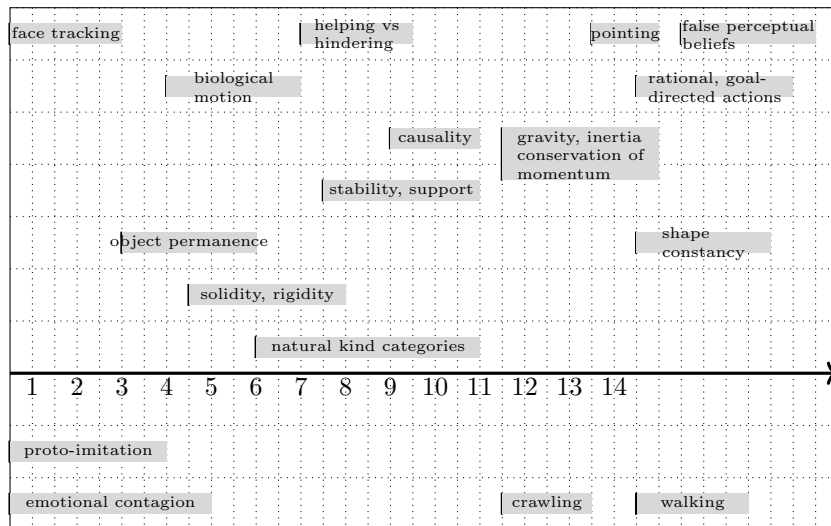


Figure 1.2: This figure was reproduced based on Figure 1 from [5] (Creative Commons Attribution 4.0 License). Based on psychological findings, it depicts general landmarks of how infants **incrementally learn** to build a **world model** during their early cognitive development.

Cognitive architecture: Cognitive science is concerned with understanding the mechanisms of action and thought, including the notions of perception, learning, knowledge, reasoning, decision-making through deliberation. There is still no real consensus on human cognition, but for some cognitive specialists such as Newell [6], it has appeared necessary to integrate different theories and hypotheses into a unified theory of cognition. Several visions, based on symbolic and connectionist views have been conceptualized and developed, giving birth to a large number of software architectures over the last few decades. Among cognitive systems, called cognitive architectures some focus on psychologically (ACT-R [7]) or biologically (SPAUN [8, 9]) plausible features in order to study living being cognition. Some are more pragmatic and focus on exhibiting cognitive abilities for practical applications (such as

SOAR [10], SIGMA [11], DIARC [12]) without biological plausibility concerns. These approaches are of particular interest, when one wants to build reliable and flexible agents, as they aim at explaining how cognition allows us to learn representations and use them to better adapt in the world.

Cognitive robots as embodied cognitive systems: Robots are specific agents are they are *embodied*. This means that in contrast to virtual agents such as conversational agent, they have a physical body that interacts with the real-world and other physical agents. While embodiment is not a subject of this thesis, keeping in mind the notion in the broader field is important as it has several implications. Cognition is closely related to body capabilities. A striking example comes from passive dynamic robot walker [13], which can mimic human walk without involving complex control and planning. Thus, the kind of high-level representations and reasoning that an embodied agent can build, are likely to be related to the complex interaction between its body, extension of its body (such as tools) and the environment including other agents. For instance, a rigid bi-arm industrial manipulator is likely to solve a storing task, in a different manner than a single rigid arm manipulator or than a compliant soft robot. Different bodies can lead to different representations. Therefore a SRA needs to have a good representation of its body and abilities, what is called proprioceptive perception. In the case of industrial collaborative robotics, we want a SRA to share a certain level of common ground with human knowledge and representations. This motivates the integration of humans in the embodied learning process of robots.

1.2.2 Human in the loop for flexible interactive robot learning

Integrating human in the loop, is also of particular interest as it can lead to system with much more flexibility and potential acceptability. For illustration, we can take the viewpoint of a human interface designer. Human centered interfaces, including collaborative robots, must often rely on careful human and task specific requirements. In traditional design, the designer has to think about a great number of possible interactions situations between humans and the designed object. Of course, in practice this assumption is unrealistic and limits are quickly reached. One striking visual example, coming from user experience field and often used in design courses is illustrated in Figure 1.3.

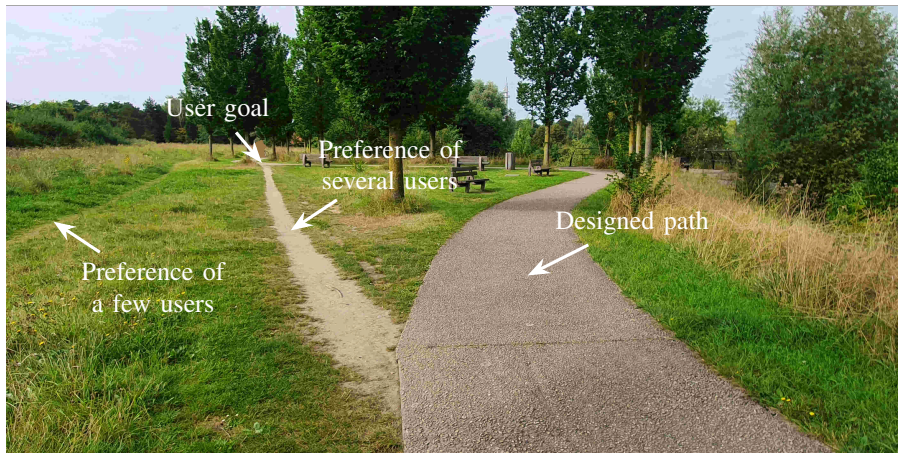


Figure 1.3: Image taken at "Citadelle" park in Lille (59000), France.

An architect has defined a certain path walk in a park that people should follow. However, as each person has preferences, they cross in different ways. Some follow the designed path but it can be seen by grass wear that, actually, a lot of people just go straight to the shortest path while a few choose a more isolated path on the left. This simple example can be generalized to every interactive system and bring to light the need of adaptive systems that take people into account. For instance, Global Positioning System devices (GPS) can be considered as such adaptive systems. For a same goal destination but different people, a GPS can be reconfigured to favour high-speed highway or touristic paths with several points of interest, while avoiding toll roads. It can also dynamically adapt its answer to minimize travel time, with respect to environmental changes, such as traffic jams or a driver which has not followed the suggested path. In industrial collaborative robotics, task achievements are also likely to depend on specific tasks and people preferences. The design of an interactive collaborative robot must be able to take them into account. But in contrast with previous systems, the complexity of human-robot interactions prevents full *a priori* specifications, as stated in [14] (chapter 1.1).

Ideally, most of the robot capabilities should be naturally reconfigurable and extendable by end-users. The designer builds a general task learner, whose learning abilities are directly leveraged by the end-user for task adaptation. Such systems can be studied from complementary viewpoints in artificial intelligence for human-robot interaction: learning from demonstrations (LfD) [14], Interactive Task Learning (ITL)[15], Interactive Machine Learning, or the in our case the more specific Interactive Robot Learning (IRL) [16].

1.3 Towards a smart robotic assistant

1.3.1 Objectives

Overall the design of a SRA requires a multidisciplinary approach. In this thesis, we aimed at building a core prototype of IRL for Industry 4.0 setting. We have fixed several specifications that should be fulfilled. An IRL agent should be able to:

- reason and to have at least partial explanations abilities. An industrial collaborative robot must be able to provide some insights to its predictions and its behaviors.
- interpret and react to human interactions in real-time. A robotic system should be able to perceive and interpret quickly to humans.
- interact intuitively with non-programmers. The IRL agent should specifically understand human natural communication means such as vision, speech, gaze, touch. Its explanations should be understandable by non-programmers. This could help build more acceptable cobots and help non-expert users to reconfigure the system in an intuitive way.
- learn quickly and incrementally a new task from low level to high-level abstractions. Carrying out a task requires both knowledges at high-level for general understanding and at low level for perception and execution in the real-world. This can be done by transferring knowledge and it needs representations and processes that foster modularity throughout the system.
- leverage a prior knowledge base for tasks execution and learning online. We do not want to teach everything from scratch to a robot. Therefore, an IRL agent should be able to leverage some prior knowledge while doing and learning modular skills to solve tasks.
- adapt to preferences and specificities such as disabilities. While the IRL agent learns new tasks, it must be able to adapt with a certain automation level its behaviors according to each individual preferences and characteristics.
- handle uncertainty in moto-perception and its inner knowledge. We want the IRL agent to know what it does not know. For that, the notion of uncertainty is important. As a measure of confidence in its own actions, it can give the IRL agent, the ability to reason about its own

predictions in order to decide to act or not to act. As a measure of curiosity, it can be a drive for learning.

1.3.2 Contributions and thesis organization

This thesis has aimed at developing and integrating the main building blocks to create a cognitive robotic architecture for collaborative robotics that is likely to get closer to the aforementioned specifications. Since pick and place related tasks are common in many industrial applications, we decided to choose planar grasping as use-case for validation on real robots. A main part of the literature review is detailed in chapter 2 which presents the main principles in the design of a cognitive architecture and a state of the art on ITL and IRL. Specific state of the art is then enriched throughout the thesis chapters. Chapter 3 details the current architecture we proposed, in terms of main building blocks, interactive learning processes and modules organization at a high-level overview. We then further detail in chapter 4, different skills learning paradigms that were investigated during the thesis. Learning grasping with real robots were used as a validation of specifications integration. Specifically, a contribution to learning from demonstration and task oriented grasping was made by the development of a specific module, developed in section B.3.2. Chapter 5, reviews the specific problems of learning uncertainty with deep neural networks and introduces how it can be used in an active learning setting. Chapter 6 describes the implementation of the architecture, integrated modules and validates the overall thesis approach with a real robot. Finally, chapter 7 introduces ongoing perspectives and future works.

Work done during this thesis was valorized through international publications:

- Contribution in learning by demonstration and task oriented grasping: [17]
- Contribution in architecture approach for interactive robot learning in industrial collaborative robotics: [18]
- An international journal article gathering, and updating the contributions with the last developments and validations for our ITL architecture, is close to be submitted.

Chapter 2

State of the art on cognitive systems

Contents

2.1	Connectionist and symbolic approaches	12
2.1.1	System understandability	12
2.1.2	Connectionism	13
2.1.3	Symbolic	15
2.1.4	Hybrid	16
2.2	Main building blocks of a cognitive architecture	17
2.2.1	Ontology as an explainable structure for components interoperability	17
2.2.2	Behavior model	17
2.2.3	Teaching complex behaviors to robots : Interactive robot learning	23
2.3	Conclusion	25

This chapter describes the main required components to build a cognitive architecture. An emphasis was made on various trade-offs between the connectionist and the symbolic view in artificial intelligence in order to justify the development of an hybrid architecture for skill learning. Works on interactive robot learning with human in the loop embrace many different research topics in learning, communications modalities, decision-making and acting in situated interaction. While there is no consensus on the ideal cognitive architecture, several decades of research work led to core design principles. Managing reasoning, planning and acting abilities at several temporal and abstract scales is determinant for smart behaviors whereas modularity is key to knowledge reuse and for architecture long term evolution. We position our work in the extensive taxonomy of cognitive architectures developed in [1] and compared it to existing ITL/IRL against our specifications, motivating the development of our own IRL architecture. Section B.2.1 introduces the two approaches one can adopt on a cognitive systems: a connectionist and a symbolic point of view. We motivate the use of a hybrid architecture to develop an IRL agent. Section 2.2 then focus on the main building blocks required to develop a cognitive system and how such systems have been used for IRL.

2.1 Connectionist and symbolic approaches

2.1.1 System understandability

Practical deployment of AI techniques in the industrial collaborative setting needs to have a certain level of understandability to be trust. In [2], authors draw a comprehensive overview of eXplainable AI (XAI) field. They emphasize the need of different levels of explainability depending on the target *audience*, as an engineer or a non expert, and the importance of a conceptual taxonomy of understandability. In particular, they distinguish interpretability and explainability of a model. Interpretability is a passive characteristic of the model related to the ability to extract meaning from the model in understandable terms for human. Explainability is related to an active characteristic of the model, where the model itself acts to clarify its decision according to a specific audience. In an IRL setting, ideally, we would like models that are both explainable and interpretable. In other words, after making a prediction, if asked, the model should be able to give some insights on its decision process (explainability) in a human understandable way (interpretability). In practice, there are two main research directions in XAI literature [2]:

- Post-hoc explanations of fully black box models, where technical methods are developed to analyze model predictions *after training*.
- Inherently interpretable or transparent models such as decision trees or hierarchical symbolic models, which thanks to their structure, help managing complexity in an interpretable way.

In the IRL setting, we specifically want *non experts* to trust the robotic agent. To be accepted by non technical users, we want the IRL agent to present in a common sense manner its prediction and decision process. As human, we trust each other because we share some common world representations and we are able to explain at high level our behaviors in an interpretable way. Understanding the lower level brain processes is not mandatory. For instance, explanations requirements is likely to vary between non user experts of the system and an engineer. Non user experts of our IRL would likely expect qualitative information, in an everyday language form or as images about robot behaviors and decision processes. In that direction, authors in [3] highlight that in the context of social robotics, robots are likely to be trust by non technical users if they are given the ability to share their intents, goals and beliefs. Allowing sharing information at that level is important, if we want a non expert

2.1. CONNECTIONIST AND SYMBOLIC APPROACHES

being able to provide valuable feedback to teach the IRL agent. On the other hand, an engineer could be interested in more quantitative information and a finer grained analysis of inner decision processes and algorithms. This requires some hierarchies in the explanation abilities and therefore in the IRL architecture.

AI agent systems can roughly be approached given two points of view: a top-down view where a complex model starts from high level and relational abstract knowledge, where most of reasoning and planing occurs, (sometimes refers as system 2 in the literature) to the sensory motor capabilities of the agent. It has usually been the territory of symbolic AI which exploits symbols for internal representations. At the opposite, bottom-up view is related to connectionist AI and aims at leveraging the interaction of several simple models from which complex behaviors emerge (at system 1 level). Both approaches have their upsides and downsides for building a robotics cognitive architecture. We present some of them to justify a hybrid approach in our architecture. We illustrate in Figure 2.1 the trade-off, in term of ease of implementation and representations, between symbolic and connectionist architectures with respect to abstract representations, data efficiency for learning, and explainability of the system.

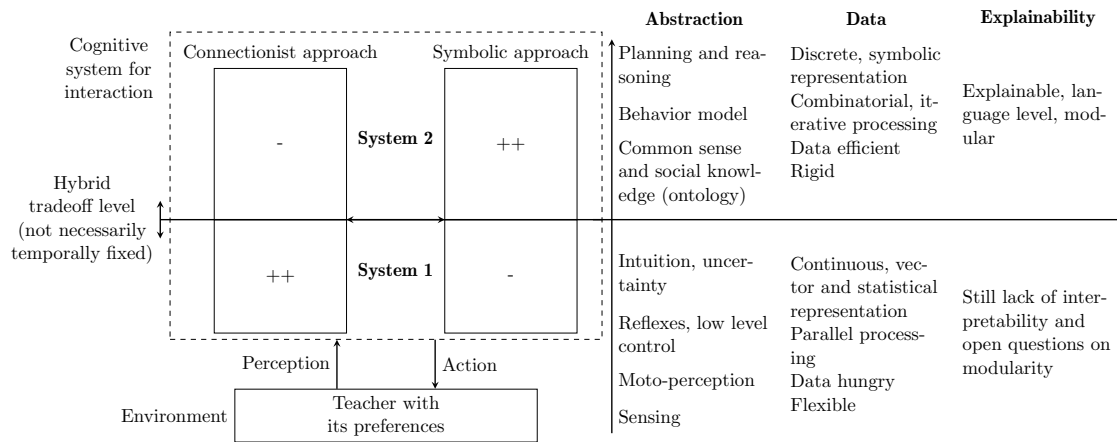


Figure 2.1: General hybrid architecture design

2.1.2 Connectionism

One of the most powerful tools used in connectionist approach are currently deep learning techniques which are now state of the art in many domains [4]. They consist in building end to end deep neural networks architectures which learn from data in a bottom-up, parallel process. Architectural

design and stacking layers have shown improved performances in learning by leveraging higher level of abstractions. Moreover, these networks are robust to noise provided that inputs are close enough to the training data. Processing of rare cases, however, as a sample outside of the distribution of training data is still an open problem. Thus, deep neural networks systems hardly generalize outside of narrow AI tasks. These techniques are very data-hungry: learning from scratch often requires much more data than what would be required for a human. Data efficiency is a serious issue when it comes to online interactive learning in robotics as data is scarce with only one or a few available data examples. Nevertheless, several techniques such as transfer learning and data augmentation can help mitigate the amount of data. Finally, high-level learned abstractions are also different from those we learn as human [4, 5]. This leads for instance to failure modes very different from those of humans, as proved adversarial examples. This lack of interpretability hinders understandability and the ability to enforce high-level prior knowledge in the system.

In deep reinforcement learning, planning and reasoning seem hard. Indeed, in most of the current deep learning models, the agent is not enforced to learn a causal model of the world. Thus, it is difficult for the agent to explain *why* it did something, to reuse behaviors across tasks or to correct biases. In the Natural Language Processing (NLP) literature for instance, GPT-3 [6], is one of the largest and most powerful model. It has shown impressive results on standard benchmarks and even few shots learning but they are also strong limitations when it comes to understanding with negative biases in language generation tasks [5]. Such system do not learn as we do in a real-world, they only learn from a big corpora of unimodal text data. Therefore it might not be enough to get a good understandability of the world, which is multimodal.

Otherwise, in order to be acceptable and trustable, interactive robots have to explain their behaviors as their actions decisions can have annoying consequences. In a real industrial and collaborative world setting, wrong robot actions could indeed be unsafe for humans, or damage goods including the cobot itself.

In that context, several promising methods are currently investigated. Most XAI techniques explore post-hoc explainability. For instance, some techniques try to produce examples to explain predictions abilities. Given a test input, we can try to find the closest train input example to explain the predictions [7]. Some works approximate locally a model prediction with more simple model such as linear models [8]. Finally many techniques rely on features based on features visualization or attribution or by gener-

ating input example based on layer activations [9]. While these techniques improve interpretability of models predictions, inner decision process of the model can still be hardly understandable, as learned representations can be far from the one we learned. To learn more efficient, inherently interpretable and transferable model representations, this require building architecture with more inner constraints during training. For instance, in vision, authors in [10] build a specific deep neural network architectures for classification, where the model has to learn images classification based on image prototypes. Prototypes are patches examples, sampled from images of the train dataset, *learned* during the training phase and that explain well the network predictions. In deep RL, learning an embedding space for skill representations [11] has shown improvements in data efficiency and for transfer learning. Other works aim at learning more modular representation such as in meta-learning of distangled features [12–14] or in bridging causal learning with machine learning [15] which could bring more abilities for planing and reasoning to deep networks architectures. It is believed that these networks and learning paradigms, given enough time and data, should be able to learn those high-level AI functions.

2.1.3 Symbolic

In contrast to connectionist approaches, many early advances have relied on the notion of symbolic programming by modeling relationships and using meaningful symbols to create smart AI systems. As they use symbols close to human language, their decision-making process is usually more concrete and understandable (section 5.3 of [2]).

Symbols are also very practical for logical reasoning and to express causality at a high-level [16]. As explicit relations and hierarchical modeling are the bases of symbolic representations, this approach allows building modular systems, which can generalize quickly and with much fewer data than current connectionist systems. They can indeed exploit objectness and functional principles. This is especially useful in the setting of IRL in industrial settings where robots are expected to be reconfigured quickly. Nevertheless, whereas connectionist approaches can start from almost *tabula rasa*, symbolic ones require the system designer to build prior common sense knowledge from scratch. There is no consensus of what common sense knowledge is and how to build it. This means that the system could embed biases and misunderstanding about the world because of erroneous designed prior knowledge. When it comes to low level moto-perception learning, symbolic approach quickly reaches limits as this type of knowledge is often not verbally explainable even for humans or because the number of rules to describe

simple behaviors can potentially explode due to low level variations in the task. Fundamentally, a unified symbolic representation referring to the outside world through motor-perception is difficult to construct. This issue is often viewed in cognitive literature as the "Symbol grounding problem" [17].

Finally in terms of understandability, a symbolic model that becomes too complex can also produce hardly interpretable explanations for humans, such as very deep and wide ensembles of decision trees (section 4.2.1 of [2]). That means that we might need some trade-off between a model complexity and its accuracy in the context of understandable IRL agent. These limitations hamper the symbolic system's ability to scale and take into account all the data variability and noise that a robotic agent faces in the real-world.

2.1.4 Hybrid

A purely connectionist or symbolic approaches do not seem to be able to take account for all the capabilities required by a truly interacting SRA. However, pros and cons of connectionist and symbolic approaches are complementary. Therefore, more and more works exploit best of both world paradigms in *hybrid systems*. For instance early symbolic architectures such as SOAR [18], ACT-R [19] have progressively integrated or exploited connectionist components to handle more diverse situations while recent one such as SIGMA [20] are built from start as hybrid. We can refer to [1] for a deep overview of hybrid cognitive architectures. Most of ITL/IRL systems discussed in section 2.2.3 fall under hybrid cognitive systems definition. There is no consensus, however, on how such hybridization should be done [21]. Overall, it is essentially a matter of trade-off between the different views. Usually, hybrid approaches use connectionist methods to process raw sensory data, while symbolic methods provides reasoning and planning abilities at higher levels.

Hybrid models allow to build architectures that can leverage modular explainable and interpretable models integrating more black-box models. Developing a fully explainable agent with good performance might be impossible, as humans themselves after all, are not fully explainable agent. However, decomposing knowledge in a hierarchical and modular way, even with specialized black box modules, is likely to improve the overall understandability of the system.

The architecture we propose can be classified as hybrid. We have chosen to exploit symbolic structures for high-level representations and learning of tasks structure whereas a combination connectionist techniques based on classical methods and deep learning are used for low perceptual learning. Both

approaches are integrated in an interactive decisional process to learn and carry out tasks. Next section provides the main building blocks for a general IRL architecture in terms of representations and behavior models.

2.2 Main building blocks of a cognitive architecture

2.2.1 Ontology as an explainable structure for components interoperability

Many architectures relies on more or less complex ontologies. An ontology is informally "an explicit specification of a conceptualization"[22] or, in other words, it is an object oriented conceptual representation build around classes, attributes/properties, and relations between these concepts. Cognitive systems often come with an ontology which provides a base symbolic structure that eases compatibility between the different modules and sub-systems, or even across different independent systems (such as other robots) [23]. An ontology allows to model and integrate expert knowledge over a domain. Composed of explicit symbols, it helps in building transparent knowledge as required for our IRL. However, building a whole ontology from scratch can be a hard task, as it requires programming abilities to build and update the ontology on complex system such as robots. Therefore, learning the ontology in an IRL setting by leveraging interaction principles with domain experts is a convenient way to build this complex ontology for real-world use cases. Conversely, ontologies have been used as bases to build coherently different kind of semantic memories, that the IRL agent can leverage to learn, reason and act more efficiently [24]. Practically ontology can be seen as a graph knowledge base that can be queried and updated depending on the current situation faced by the agent.

2.2.2 Behavior model

As robots are acting agents, we also need a behavior model that can exploit the prior and learned ontology. To act in the real-world, an IRL agent has to develop the ability to generate relevant complex behaviors. For this, it needs to be controlled to simultaneously plan and react while learning in an online way.

Robotic cognitive architecture have progressively identified three main layers of interest [25, 26] (see Figure 2.2): a functional layer tailored to action, perception and learning; a decision layer tailored to planning or supervision; and an execution layer where the behavior model intervenes, for interfacing

2.2. MAIN BUILDING BLOCKS OF A COGNITIVE ARCHITECTURE

and coordinating other layers of the system according to the current task requirements.

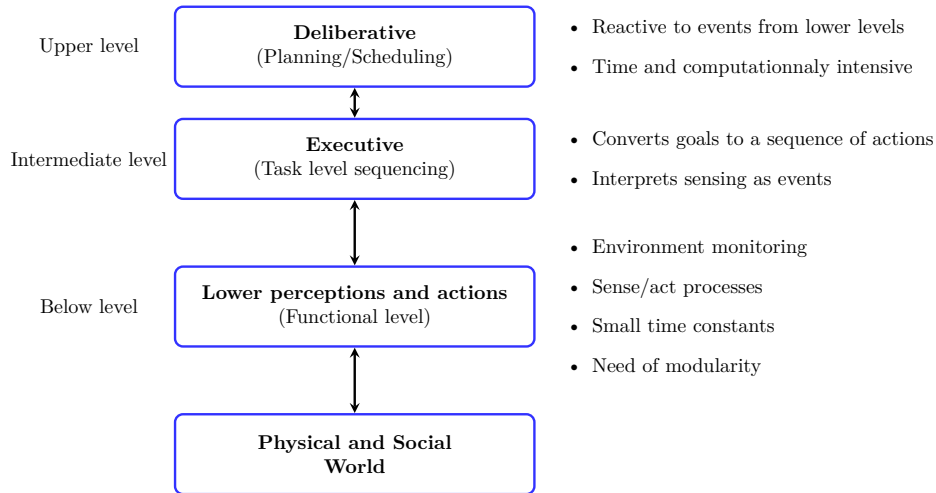


Figure 2.2: Robotics architecture can be represented through three layers of control.

Planning is used to predict actions effects and to search for the best sequence of actions in order to reach a given goal, while acting consists in the executive parts. As the environment dynamically changes during execution, due to agent own actions or external effect, it might need to re-plan and refine its plan, given new and past information (see for instance section 2.6 of [27]). In order to account for the tight integration between planning, acting and learning in an architecture, behavior models are needed to encompass the descriptive part (the what) and the executive part (the how) of the skill while being open and modular enough to have adaptation capabilities.

Adaptation capabilities should also be extended to preference learning and handling. Indeed, cobots could interact with different human workers, having, even for the same task, their own characteristics and preferences during interaction. It has been shown in [28] that preferences learning is associated with higher confidence in robots motivating this integration in the architecture. In the literature several methods has been used for specific tasks, such as the use of Markov Decision Process (MDP) in closed scenarios [29], , implicit discovery based on user defined constraints [30] or on bayesian networks (BAN) [31] for words to actions learning.

Finally, we can determine the following requirements for an interactive robot. The behavior model should:

- be explainable. With respect with our specifications and section 2.1.1, the behavior of the robot

2.2. MAIN BUILDING BLOCKS OF A COGNITIVE ARCHITECTURE

should be at least partly understandable. Therefore, the behavior should be complex enough to allow the agent to execute tasks but also simple and interpretable enough to be understood by non-expert users.

- describe both the descriptive and executive parts of behaviors. The IRL agent must especially be able to describe the *what*, *why* and the *how* of its actions.
- interface with databases of priors knowledge (such as users preferences). The behavior model should allow integration of prior knowledge of different types such as rule based knowledge like users specificities and preferences. For instance, this can be the dominant hand, or bio-metrics information for security access in industrial restricted area.
- be interoperable with world model built from sensing and acting modalities (such as speech, gestures, touch). As the agent builds a world model, the behavior model must be able to use it with respect to the ontology.
- interface low level and high-level skills in a multimodal way. Learning skills need both information at high and at low level and therefore the behavior model acts as a bridge between them.
- interface with learning techniques such as deep neural networks. As deep learning techniques has emerged as very powerful tools for learning, the behavior model must easily interface with those systems
- allow fast learning and strong generalization thanks to behavior reuse with composable and parameterizable representations. In the IRL setting, learning is done online when interacting with a human teacher. Therefore, we want learning to be fast while keeping good generalization abilities. This can be done by exploiting parameterized modular behaviors, as actions template. It allows, indeed, to reuse a learned behavior in several related task with minimal updates.
- allow refining actions in a reactive way. As environment is dynamic and can change under the IRL actions or other agent actions, the robot must permanently alternate between perception coming from sensory streams, learning, planing and acting in a deliberative loop.

In order to choose the paradigm to implement, we have reviewed the literature on ITL/IRL and the behavior models they used. We based our comparison by updating the state of the art in [32]

(part 2.) with regards to our requirements. We can distinguish procedural models that explicit the temporal structure of behaviors and those that only map state to actions. For instance, in [33], use of probabilistic models in order to learn a mapping between the best action command given the current utterance. In [29, 34] authors extend the MDPs framework with relational activity processes (RAPs) [35], giving them more relational representation power to model concurrent actions between the robot and the instructor. They use these models to learn a RAP where instructor preferences are learned as specific paths in the overall process. Communication modality is limited to touch screen interface to send utterances to the robot. These techniques are interesting to produce efficient behaviors. However, they roughly maps a state to the best action according to the learned policy without modelization and understanding of the effects of actions. Moreover, because they rely purely on data driven with gradient based learning technique, learning a new policy could take several trials even for simple tasks. These limitations hamper the symbolic system’s ability to scale and take into account all the data variability and data noise that a robotic agent will face the real-world.

Many more ITL use classical behavior models based on symbolic procedural models [31, 32, 36–61]. This is interesting because the symbolic nature of these models make them explainable, and it allows them to learn fast (typically in a one shot manner) by leveraging the use of high-level abstractions. In the simplest models, procedural knowledge can be represented as a mere sequence of primitive actions (SEP) [44, 48, 56]. It eases the implementation of behaviors but it limits the modularity of the system and the ability for action branching. Complex reasoning such as changing actions procedure based on preferences could not be handle in this framework. Most ITL/IRL however rely on more complex representations with skills than can be modeled in terms of *preconditions* which are conditions that must hold true before carrying out the action, *postconditions* (or effects) which will be true action and operating conditions which must hold true during the action. In contrast to simple action sequence it allows more modularity and the integration with standard planning techniques. There are several technical frameworks that were used in the literature. There is no widespread framework for IRL architecture, but most of them share common characteristics, with slight nuances:

- Finite State Machine (FSM) and Hierarchical Finite State Machine (HFSM) [62]. A Finite state machine defines a list of states and an explicit set of transitions between states. When complexity of behaviors grows, FSM can become unmanageable due to state-space explosion. HFSM is an evolution of FSM as one can now define one transition between set of states (superstates), rather

than individual transitions for all sub-states. Thus, they are easier to design and implement, as they reduce the state explosion problem in complex scenario. They were used in [60]

- Action Script (AS) [63] (section Action Representation) specifically used in DIARC cognitive architecture [59, 63, 64], is described as a compact way of specifying hierarchical robot behavior. An AS is an expression $\alpha(p_1 : t_1, p_2 : t_2, \dots, p_m : t_m)$ where α is an action symbol and $p_i : t_i$ a parameter p_i of type t_i (such as a reference to a graspable object). Types are used as abstract classes for generalization. Each AS contains a sequence of action $\alpha_1, \alpha_2, \dots, \alpha_n$ and is associated with a set of pre-conditions, post-conditions and operating conditions. Each action α_i can be an action script or an action primitive which contains a single action.
- Operator with production rules used in the Soar cognitive architecture (SPR) and with pre and post conditions in Rosie ITL agent [46, 57, 65]. To achieve a goal, production rules conditions are matched to the SOAR working memory and trigger other operators acting on inner memories or on external modules for action.
- Percept-Response (PR) which is essentially event-driven behavior. Used in [40], perceived events directly associated to a sequence of actions. Modularity and refinement abilities seem limited.
- Shared Plans (SHP) with preconditions and postconditions in [41, 42, 45] which integrates representation of other agents to deal with behavior synchronization for collaboration.
- Task Description Language (TDL) used in [66]. TDL is a language used to describe tasks as sequence of actions or conditionals and are represented as a Petri Net Plan (PNP) [67].
- CRAM Plan (CP) used in [68] with KnowRob [51, 52]. CRAM uses a custom language called CPL (for CRAM plan language) based on Common Lisp for both planning and reasoning at task level. It exploits KnowRob, a web knowledge base of skills and facts with reasoning capabilities based on Prolog [69].
- Some models do not seem to have been used in interactive learning setting but interesting properties in terms of action refinement such as Procedural Reasoning System (PRS) [70, 71] and Refinement Acting Engine (RAE) [27] (chapitre 3.2) in [72].

- Hierarchical Task networks (HTN) [73, 74] used in [32, 47, 49, 54, 75]. HTN is a tree that consists of primitive task nodes that can be executed directly and non primitive nodes (called compound tasks) that can be decomposed and refined before execution. Different decompositions are allowed and depend of specific methods.
- Behavior Networks (BN) used in [31, 38, 39]. A behavior maps a set of inputs such as sensor information to a set of actions. A BN is a graph of behavior where each edge represent a transition between different behaviors. Internally, every behavior is defined as a finite state machine with an explicit start state (preconditions) and termination states (postconditions), depending on whether the behavior reach or does not reach the goal.
- Network Abstract Behavior (NAB) [76] used in [37]. A NAB is a hierarchical representation of Abstract Behaviours (AB). An abstract behavior is a process composed of several input ports and output ports. Input ports consists of action status of the behavior, its preconditions, sensory inputs, activation and inhibition levels by other behaviors. Output ports consists of primitive action activation/deactivation, and postconditions status. There are different types of preconditions depending on wether conditions must be valid during the whole action (permanent preconditions) or can change during action (enabling conditions).
- Behavior Trees (BT) [77] used in [50, 60] and in our architecture [61]. BTs are introduced here after and detail in section 3.1.4 of chapter 3.

In our architecture, we chose Behavior Trees (BTs) as the behavior model. BTs are tree based models which allows a clear separation between the tree structure (the descriptive part as a control flow of behaviors) and the implementation of the nodes (the executive part). They are heavily used in the game industry over FSM that are prone to state explosion as behaviors become more complex. The use of explicit parallel nodes also ease the execution of parallel processes as required in a multimodal interactive setting. Failure handling is much easier and is at the core of the learning process in our architecture (see section 3.3 in next chapter for more information). The hierarchical nature of BTs eases the implementation of refinement methods: given high-level actions, it is possible according to environment changes, to branch in a reactive way through different and more concrete sub actions. This appealing properties in terms of behavior modularity make them a relevant alternative to HFSM.

Furthermore, subtrees can be added or removed anywhere in the BT without modifying other components, while in FSM such modifications implies to redefine all transitions leading to or starting from the state. Finally, the flexibility of these models allows to extend standard BTs with preconditions and postconditions nodes [78], which helps build representations for planning.

PR on its own does not seem to be able to propose refinement methods as it only maps one perception to one behavior. In contrast, refinement methods are also at the core of the following models: TDL, AS, CRAM Plan, PRS, RAE, HTN, BN. In Soar, SPR, thanks to sub-goaling is also capable to provide re-planning ability. One drawback is that those models are either tightly integrated within the underlying architecture, which make them hard to transfer to another one or because they use specific language that are not easily integrable and interoperable with python and deep learning frameworks (Common Lisp for CRAM, C/C++ for openPRS and SPR, java for RAE).

HTN and BTs are close in terms of representation as they both leverage a graph structure with refinement abilities. HTN has traditionally been more focused on long term planning, while BTs, are specifically designed for execution of behaviors with reactivity concerns. This of particular interest for the online interactive learning setting of the IRL. Therefore, BTs might be not the most suitable tool for **long** term planning. However, thanks to pre/post conditions extension, it has been shown that they can be combined with proven traditional planners such as HTN planners [78, 79], bringing best of both frameworks.

2.2.3 Teaching complex behaviors to robots : Interactive robot learning

Once we have an ontology and a behavior model we can organize an architecture for IRL. In the literature, most IRL have particularly focused on learning high-level procedural knowledge through the chosen behavior model. Usually low level perception such as object recognition and low level motor abilities are given *a priori* and are a fixed knowledge. Only a few works in IRL have tackled and demonstrated interactive learning of **both** high-level and low level skills requirements to solve a given task, we focus on these works. In [54], the IRL agent learns action primitives by observing the human during the interaction. In [46], author teaches pick and place tasks to a simulated tabletop arm. The agent can learn online, through KNN classifiers, simple perceptual cues such as color, size and shape. Similarly in [80], authors also teach online to a real-world manipulator, color, size and shape of unknown objects. Visual perception is based on clustering objects based on color and

2.2. MAIN BUILDING BLOCKS OF A COGNITIVE ARCHITECTURE

depth. Nevertheless, learning online more complex perceptual features in an IRL setting is rare. Such features can be location affordance and complex visual features learned by deep neural networks. Our architecture aims at learning both high-level procedural knowledge about the tasks but also complex low perceptual features necessary to solve a tasks. Another point of discrepancies concerns the way behaviors are taught to the agent. A behavior can be taught in a one sided way where the teacher explain sequentially all the tasks to the agent. This method is not always adapted as it puts a lot of cognitive burden on the teacher. It is indeed not easy for the teacher to know what the agent does not know. IRL can use a mixed initiative approach [81] with an emphasis on language as suggested in collaborative discourse theory [82] or with active learning by demonstration [83]. In these settings, during the interaction, the IRL asks the teacher for the missing knowledge it needs to carry out the task. This enables a flexible, natural and incremental way to teach new behaviors to the IRL agent. Finally, to our knowledge, preferences learning to adapt the behavior according to the human in ITL architecture has rarely been demonstrated, such as in [29, 34]. As stated in ours specifications, a robotic agent in an industrial setting is likely to be used by several different humans with specificities or disabilities. Being able to quickly reconfigure learned behavior according to an identified human is therefore an import feature we added in our architecture. As learning knowledge happens both at high and low levels, preferences learning has also to take into account both of them.















We synthesize in table 2.1 a general comparison of different ITL/IRL agents and some cognitive agent in interaction with humans and we show that to our knowledge, our IRL is the only one to handle all our specific requirements. We point out interaction modalities leveraged by the agent, the modularity and reusability of the learned behaviors, the ability of the agent to adapt behaviors to preferences, the type of used robotics platforms, exhibition of perceptual and procedural learning in a mixed initiative and incremental way.

We list here the legends used in the table:

- ✓ : implemented
- × : not implemented/no real robot
- 🔊¹ : speech modality

¹These icons were used from open source repositories at <https://fonts.google.com/about> and <https://fontawesome.com/>

2.3. CONCLUSION

- ¹ : gesture demonstration
- ¹: GUI, tactile or mouse
- ¹: kinesthetic demonstration, imitation
- ¹: ease of behaviors composition,  behaviors composition is limited or not possible
- ¹: preferences learning,
- ¹: demonstration of speaker location and/or skeleton
- ¹: written words (on keyboard)
- ¹: touch screen,
-  : mobile robot
- ¹: manipulator
- ,  : mobile robotic manipulator
- ¹: humanoid torso, AIBO humanoid robot, multiple humanoid social robot such as iCub, Baxter

2.3 Conclusion

In this chapter, we have developed a state of the art of IRL agents and cognitive systems within a symbolic/connectionist view and how these approaches exhibit complementary upsides and downsides. Exploiting existing ITL was difficult, as most ITL/IRL were either platform specific, not open source or seem not maintained. Moreover, most IRL agents could not validate all our specifications in the same and unique approach, in particular learning both high-level and complex low level features, related to human preferences and with ease of integration of deep learning modules. This has motivated exploring the development of our own hybrid cognitive architecture for our collaborative industrial use cases. Eventually, our main objective and contribution is to integrate several complementary ideas from the literature in a single architecture, to build an IRL exhibiting interactive and incremental learning capabilities, not only at high-level but also at low level, with deep learning modules, while being able

2.3. CONCLUSION

Table 2.1: Comparison of properties exhibited in cognitive and interactive robotics learning systems.

Publications	Modalities	Behavior Modularity	Preferences learning	Use case platform	Perceptual learning	Procedural learning	Incremental learning
[84]			×	×	×	×	×
[85]		SPR	×	×	×	✓	✓
[36]		SA _w PP	×		×	✓	×
[37]		NAB	×		×	✓	✓
[38]		BN	×		×	✓	×
[39]		BN	×		×	✓	✓
[40]		PR	×		×	✓	✓
[41]		SHP	×		✓	✓	✓
[42]		SHP	×		×	✓	✓
[31]		BAN	×		✓	✓	×
[86]		MDP	×		×	×	✓
[44]		SA	×		×	✓	✓
[33]		MDP & SA	×		×	✓	✓
[45]		SHP	×		×	✓	✓
[46, 65]		PR	×		✓	✓	✓
[47]		HTN	×	×	×	✓	✓
[48]		SA	×		×	✓	✓
[49]		HTN	×		×	✓	✓
[50]		BT	×		×	✓	×
[51, 52]		CP	×		×	✓	×
[29, 34]		RAPs			×	✓	✓
[53]		SA _w PP	×		✓	✓	✓
[80]		HT-N/MDP	×		✓	✓	✓
[32]		HTN	×		×	✓	×
[54, 75]		HTN	×		✓	✓	✓
[55]		SPR	×	×	✓	✓	×
[56]		SA	×		×	✓	×
[57, 87]		SPR	×		×	✓	✓
[63, 64]		AS	×		✓	✓	✓
[60]		HFSM,BT	×		×	×	×
[61]		BT	×		✓	✓	✓
chapitre 6		BT			✓	✓	✓

2.3. CONCLUSION

to be quickly reconfigured according to human preferences. The next chapter (chapter 3) focuses on the design of this hybrid architecture.

2.3. CONCLUSION

Chapter 3

Design of a cognitive architecture for Industry 4.0

Contents

3.1	Base ontology for our ITL/IRL	30
3.1.1	Robotic Agent as a goal driven agent	31
3.1.2	Environment representation	31
3.1.3	Utterances	33
3.1.4	Skills and actions primitive	33
3.2	Hierarchical, modular representations	35
3.2.1	Semantic declarative memory	36
3.2.2	Working memory	36
3.2.3	Episodic memory	37
3.2.4	Procedural memory	37
3.2.5	Perceptual memory	38
3.3	IRL process with preferences	39
3.4	Conclusion	41

By taking inspiration from the state of the art in IRL/ITL, cognitive architecture, and recent advances in deep learning architectures, several design choices were made to build a base architecture that could handle our specifications. This chapter illustrates and details the architecture organization in terms of representation and decision processes at the symbolic level. We first focus in section 3.1 on the base ontology used in the system, for knowledge interoperability and high-level abstractions for IRL. We further detail in section 3.2, the complementary inner symbolic memories of the system that are used as high-level relational representations, grounded by connectionist learning components to real-world data. Finally, we explain in section 3.3, the main deliberation processes used by the IRL agent for learning a task structure and related skills by leveraging an incremental and mixed initiative interaction process. This process is based on the concept of *failure* and *success* of goal-driven behaviors while representations are leveraged to learn and take into account human specificities and preferences during interaction.

3.1. BASE ONTOLOGY FOR OUR ITL/IRL

We detail in this chapter the main organization of our architecture, its representations and its learning processes. Figure 3.1 provides a high-level overview of the architecture. It presents the different representations and how they interact in order to build complex behaviors. Each block is explained in this section.

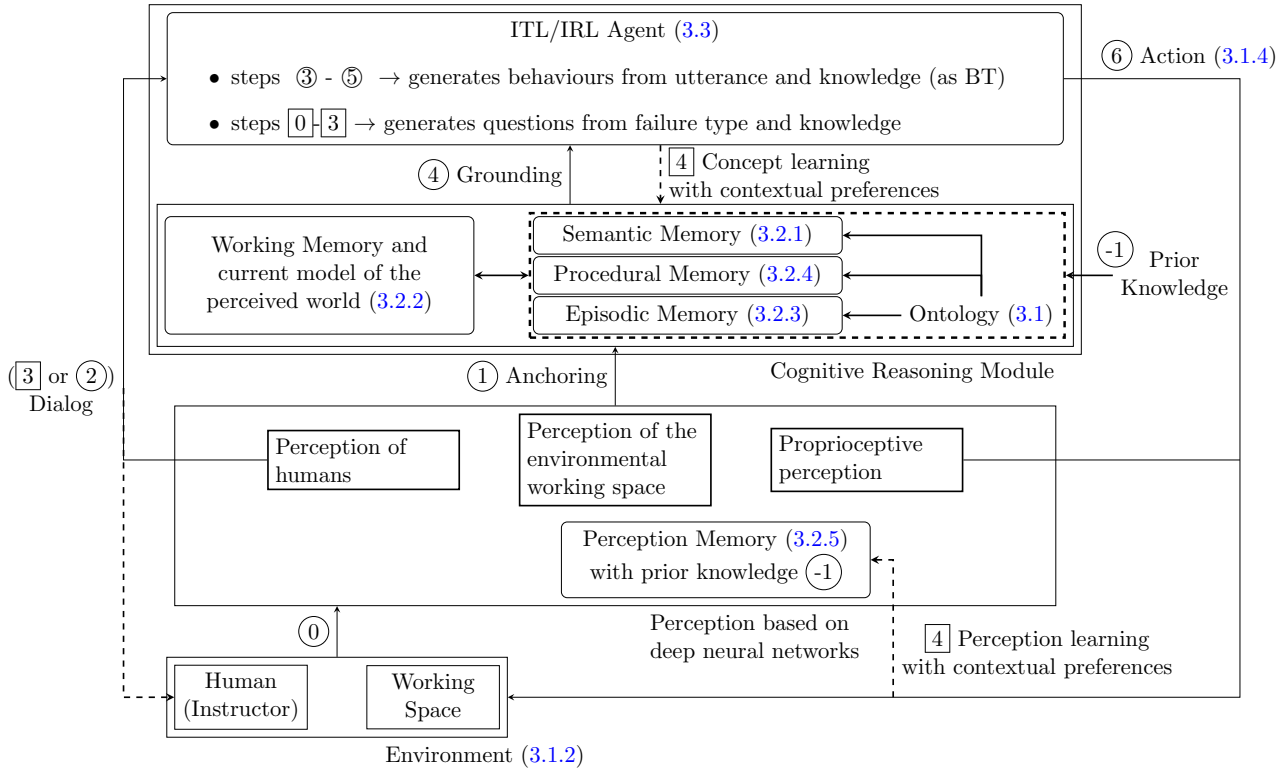


Figure 3.1: High-level overview of the architecture. The architecture consists of perceptual modules based on connectionist approaches, symbolic relational representations, and a deliberative process for interactive robot learning with human. The IRL process consists of two interleaved paths (a plain path with circled number and a dashed path with boxed number) which are described in section 3.3.

3.1 Base ontology for our ITL/IRL

As stated in section 2.2.1, an IRL agent needs an ontology for the interoperability of its components. We describe in this section the base ontology that our IRL agent leverages during the interaction to express and to learn new behaviors, or relevant perceptual features with preferences. For now, the ontology is quite standard as our current goal is more focused on the global architecture foundations and validation. More complex ontology could serve as a basis, such as the DOLCE ontology [1] leveraged in the IRL from [2] (chapter 4.2) or KnowRob [3] used in CRAM architecture [4]. Figure

3.2 illustrates a simple ontology.

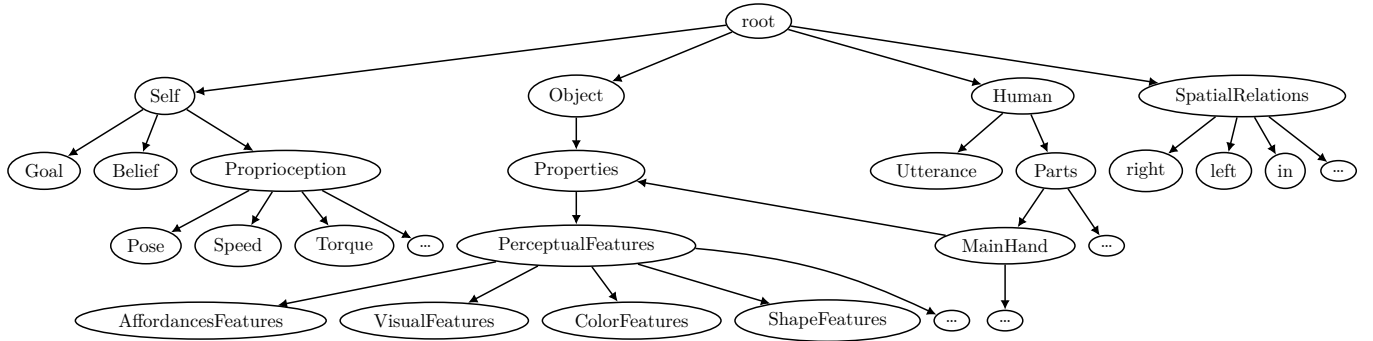


Figure 3.2: Base ontology example overview.

3.1.1 Robotic Agent as a goal driven agent

The IRL agent, represented as *Self*, has to solve tasks that are driven by *Goals*. A *Goal* can be described as a first-order logic statement of predicates over the environment, that the IRL agent must satisfy to validate the goal (i.e each predicate is considered True). For that, it builds a plan based on its *Skills*. Goals are built from *Utterances* of humans and from agent inner representations in terms of post-conditions requirements. The IRL agent has also proprioceptive abilities such as its *Location*, joints or Cartesian state in space, that are leveraged to carry out actions. The IRL agent is also able to *Focus* its attention on an *Object*.

3.1.2 Environment representation

The environment is seen as a continuous 3D space which is composed of entities. Among entities, we specifically distinguish *Human* and *Physical Object*. Entities are given a *Location* (a 3D vector coordinate) and a 2D surface *Area*. Moreover, it is possible to describe some spatial relations between entities such as right of, in, left of. Those symbolic representations can be grounded to language and real-world data by specialized connectionist modules for scene and human detection and understanding.

Human: The human agent is represented in the ontology as *Human*. *Human* is a complex, structured class that consists of specificities such as *Name* or sub-parts like its dominant *Hand*, which help represent the human characteristics.

A *Human* can communicate *Utterances* to the robot, equivalent to a set of rules, that should be

interpreted and executed by the IRL agent to learn how to solve a task. Currently we make the limited assumption that the human is an oracle. Therefore, it always provides unambiguous, trustful information to the robot and there is no implemented corrective feedbacks [5] of previously learned rules.

The *Focus* of the IRL agent can be triggered by a *Human*. Hence, it allows a shared and explicit representation of which *Object* of the working space the human wants to work on.

Example of technical integration of connectionist modules to ground *Human* to the real-world are given in the experimental validation in chapter 6, section 6.2.3.

Physical Objects: The world is assumed to be composed of salient physical objects. They consist of a set of perceptual properties which are built from a stream of data provided by sensors. It assumes that the IRL agent has prior segmentation capabilities that are used to discover proto-objects [6]. These proto-objects are given by a *Location*, an *Area*, and can be tracked according to perceptual features, and are used as object precursors.

For instance, by equipping an industrial manipulator with an RGBD camera, an *Object* can be categorised from a detected proto-object according to different perceptual properties such as its *Color*, *Visual Patterns*, *Shape*, *Affordances*, *Locations*, *Areas*.

More specifically, for an affordance, we use the classical definition with contextualization [7, 8]: an affordance *aff* is a triplet $aff = (o; ca; e)$ where *o* is an object, *ca* a contextualized action, and *e* the effect of the action on the world. A contextualized action *ca* is an action accounting for a context that can be for instance preference learning. A contextualized action will be validated if the effect *e* are in the relevant postconditions.

Learning connectionist components can be leveraged to ground those symbolic representations into data. The IRL agent can then learn objects perceptual properties such as visual features and affordance, given the context of the task and Human preferences and characteristics. An example of technical integration of such connectionist modules will be given in the experimental validation chapter (chapter 6), section 6.2.2 and 6.2.4.

3.1.3 Utterances

An *Utterance* can be built from a verbal or a non verbal perceived interactions act such as speech, pose, gesture. It is either interpreted as a *Goal* to achieve or as information for learning events. This is done through the use of a communication protocol and semantic analysis. Connectionist components are used to ground perceived interaction into a symbolic *Utterance* of words which are then further mapped into *Human's* intents. Each word is given a type called a Part Of Speech (POS) tags. Verbs in sentences are related to the tasks and actions to carry out. Nouns are related to objects on which to accomplish the task. Adjectives can refer to object attributes. Prepositions refer to temporal or spatial relations between several objects. Each word in the *Utterance* has to be grounded to the physical world, giving the IRL agent a better understanding. An integration example of connectionist components is provided in the experimental validation chapter (chapter 6), section 6.2.1.

3.1.4 Skills and actions primitive

With respect to the comparison in section 2, we chose Behavior Trees (BTs) [9] as the behavior model of our architecture. BTs were originally developed in the video games industry [10] for virtual agents, commonly known as Non Playable Characters (NPC). While NPC evolve in known environments, robots usually evolve in partially known or unknown ones.

Yet, BTs have several interesting properties which explain their growing use in control architecture for robotics. They have been proven to generalize several well-known control architectures such as the one based on finite state machines or decision trees [9] (chapter 2). Their graphical nature fosters *modularity* and *explainability*, as each individual BT can be run independently or can be *combined* with other trees. Furthermore, they allow to design *reactive* behaviors to unexpected events.

As such, they have been used and extended in various robotics contexts such as learning from demonstrations [11, 12], mobile robotics [13], unmanned aerial vehicle [14] and more general robotics architecture [15, 16]. In terms of BTs technical development, there is a growing amount of libraries available in various languages [17, 18]. They can leverage different programming paradigms for their practical implementations, such as multithreading with preemption, or asynchronous programming [19]

Behavior trees (BTs) are composed of several (usually six) kinds of nodes illustrated in Table 3.1:

a set of *control* nodes that helps manage the decision flow, a set of *executive* nodes that carry out actions, a *decorator* node that helps build more complex control nodes such as retrying an action or a subtree until success. Each node can return a status, usually *success* or *failure*. Control nodes return *success* or *failure* according to the return status of their children and the rules defined in table 3.1.

In order to define complex modular behaviors, compatible with planning and reasoning purposes, we have designed skills with BTs using the traditional precondition, execution, postconditions (also called effects) model (see Figure 3.3). In [9], authors provide a detailed formal overview of BTs and their use in robotics. The *fallback* node executes children in order (from left to right). If the first child fails, the execution continues to the following child, which act as a fallback. If a child succeeds, the fallback returns success without visiting the following child. The *sequence* node tries to execute all its children in order. If any child fails, the sequence stops and propagates the failure back.

In our architecture, we specifically exploit BTs failures mechanisms as a high-level signal for learning purposes by leveraging the *fallback* node. This allows to incrementally refine or expand the tree during the interaction of the IRL agent similarly to [20]. Transparency, modularity, and efficiency of BTs appear naturally as behaviors reuse, update and composition can be done by leveraging graphs. Finally, the *parallel* node is used for simultaneous multimodal sensing. For now, modalities are assumed to have orthogonal effect on memory, so that there is no need for low level and complex concurrency management. Postconditions help in checking if the skill execution is a success while preconditions determine if the agent has the knowledge to carry out the skill. Another interesting property of skills modularization with conditions is the fact that it helps the agent in doing active perception: the agent only checks conditions that are relevant to the task, according to the previously learned skills.

A *primitive action* is a leaf in the overall behavior model. Therefore, it is directly performed without further refinement by the robot. Examples of primitive actions are the opening or closing of a gripper, point to point motion, sending questions to the human.

3.2. HIERARCHICAL, MODULAR REPRESENTATIONS

Table 3.1: Control flow and action nodes in the standard behavior tree framework

Execution nodes	Symbol	Success	Failure
Action	□	Execution is carried out	Exception during execution
Condition	○	Condition is true	Condition is false
Control nodes			
Sequence	→	All children must succeeds	One child fails
Parallel	⇒	More than $M \in \mathbb{N}^*$ children succeeds	More than $N \in \mathbb{N}^*$ children fail
Fallback (or Selector)	?	One child succeeds	All child fail
Decorator	◇	User defined	User defined

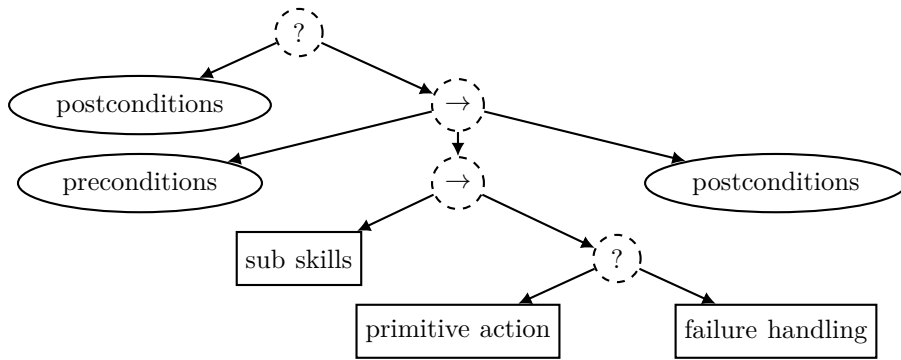


Figure 3.3: Base skill model

3.2 Hierarchical, modular representations

Given our ontology, we exploit different kind of memories, useful for different aspect of task learning:

- a Semantic Declarative Memory
- a Working Memory
- an Episodic Memory
- a Procedural Memory

We leverage relational graph representations for modularity, better explainability and learning with respect to our specifications.

3.2.1 Semantic declarative memory

A semantic declarative memory is a kind of database of semantic expressions. It stores terms in a way that is grounded to language. This allows to represent the high-level knowledge base on concepts and facts. By querying the semantic memory, one can then leverage language to get access to the agent knowledge. Figure 3.4 illustrates an example of semantic memory. Practically, this is implemented as a semantic graph network, whose nodes point to the conceptual instances provided by the ontology. This memory can be linked to other databases of facts such as humans specificities for preferences handling.

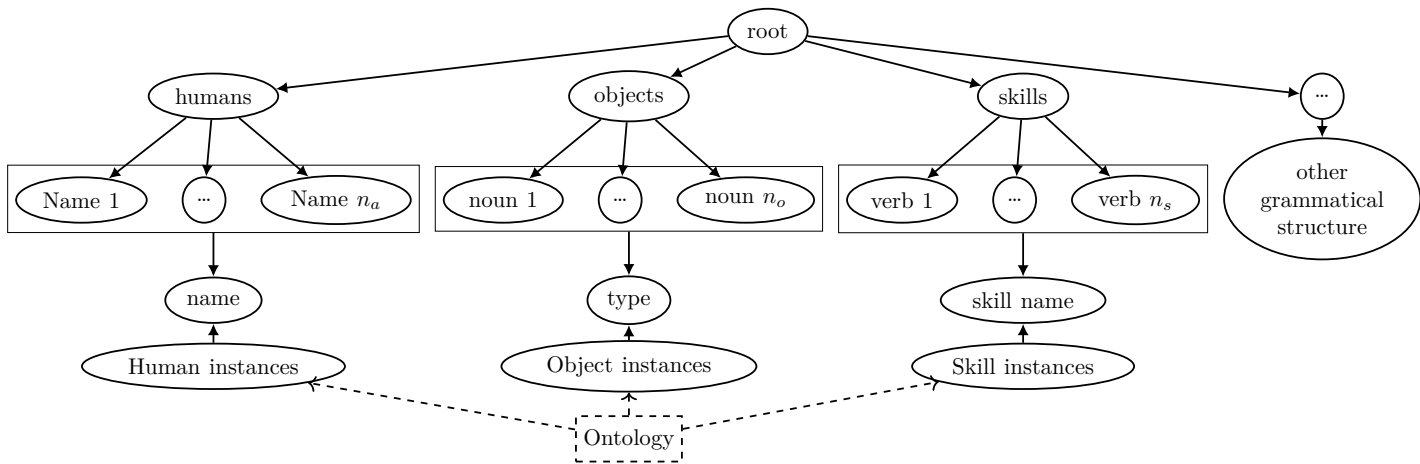


Figure 3.4: Example of a semantic memory database derived from the ontology

3.2.2 Working memory

The working memory is a short-term memory that provides an explainable representation tool for learning and reasoning on the current situation. This is where links between the low-level perception and the high-level symbolic representations take place. This enables to instantiate objects of the ontology and to ground symbols with the current perception and belief of the agent. It is represented and implemented as a semantic, relational graph network. This memory contains:

- entities that are categorized for instance as objects or humans and the robot (Self).
- predicates which are relations and properties over entities, for instances: spatial relations, colors, affordance, neural networks features.

3.2. HIERARCHICAL, MODULAR REPRESENTATIONS

During a session, each entity and predicate is given a unique integer Id when it is first created. This Id allows to have a general cross-referencing system in memories. It is used as a standard querying cue in the different inner representations.

Figure 3.5 present an example of the working memory representation when an object is detected by the IRL agent.

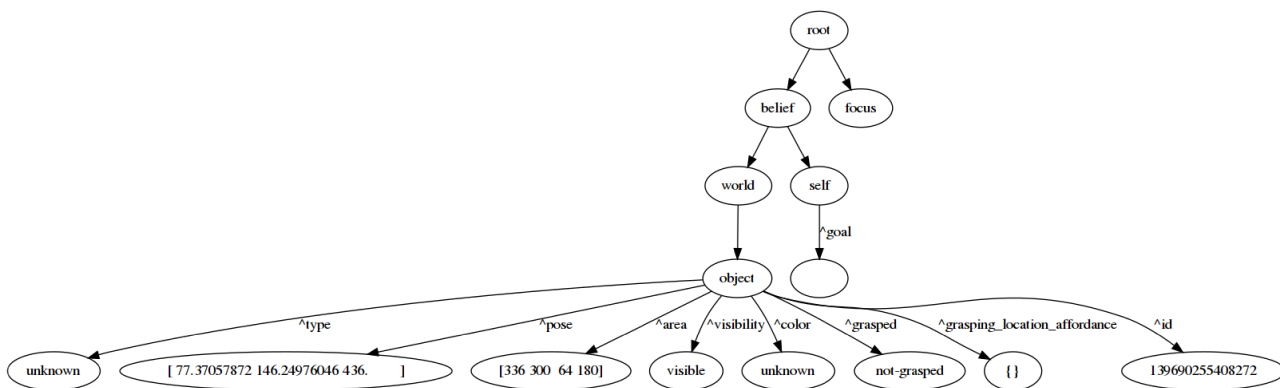


Figure 3.5: Example of the working memory derived from the ontology

3.2.3 Episodic memory

The episodic memory is a set of working memory episodes that are stored across the time according to specific rules. Thanks to this memory, an agent can remember previous encountered situations and decisions. It can also track objects properties within time. Defining the right rule to decide when to store an episode is not a straightforward task as change in the environment can be the fact of observable or partially observable external events. Ideally the agent should be able to detect these changes and build hypothesis about the causal nature of these changes. In the current architecture state, we store episodes between two actions of the robot. This rule is enough to limit the memory requirements of the system while allowing some interesting temporal reasoning: bringing back an object which was first taken by querying its location property. The memory is session dependent and therefore it is erased when the agent is shut down.

3.2.4 Procedural memory

The procedural memory is a set of skills that are represented as specific BTs. As describe in section 3.1.4, skills can be related thanks to the modular properties of BTs.

3.2.5 Perceptual memory

The perceptual memory is set of isolated or interacting modules based on connectionist approaches, mostly deep learning, and traditional signal processing approaches specific to each modality such as vision, speech, torque sensing. Humans are likely to expect that an IRL agent can quickly learn to recognize and interact with various objects. Reaching these needs in terms of reconfigurability is hard with classical deep learning modules because of their data requirements.

In that context, transfer learning [21, 22] is used as a way to leverage in a hierarchical manner features learning, where prior deep networks are pre-trained on the sensory modality, in a supervised or unsupervised manner. This first training is usually long, data-hungry and is done offline. Once the network learns features, these can be leveraged by transfer learning to define new, specialized networks that are more quickly optimized, online, with less data. This neural modularity has also a positive side effect as it allows to better manage memory resources which can be limiting in some settings. Figure 3.6 illustrates the main architectural principles in the perceptual memory. For instance for vision modality, popular variations of pretrained neural networks can be used such as Densenet [23] or MobileNet [24].

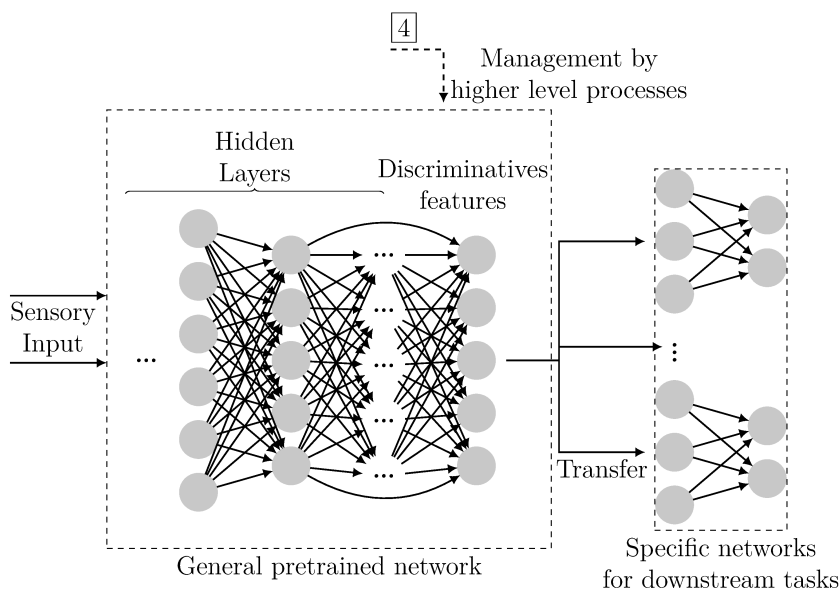


Figure 3.6: Transfer learning in deep neural networks is leveraged to allow fast reconfiguration and resource management. To simplify the figure, we represent here a dense network but any kind of neural networks layers can be used if relevant for the sensory modality.

3.3 IRL process with preferences

The interactive learning process is based on error handling called *failure* or *impasse* during the program execution flow. It is built in the architecture by leveraging the tree structure and the ontological representations in the behavior trees framework. Figure B.7 illustrates more specifically the decisional process that occurs during a skill execution. Here we mainly focus on the deliberative interaction process. More details on the architecture implementation and the choice of sensors and interaction modalities are given in chapter 6. Figure 3.7 describes two paths, one in plain line and one in dashed line. Plain line path represents what happens when the agent has all the knowledge to act, the steps ordering are represented by surrounded numbers. Dashed path represents what happens when a failure occurs, the steps ordering are represented by boxed number. The interleaved execution of these two paths is at the core of the mixed initiative interaction cycle during which the robotic agent acts according to human instructions or learns from failure and interaction.

For the plain line path, the typical interactive cycle is the following: we suppose the agent has some prior knowledge ((-1)). The agent proprioceptive state, the working space and the instructor interaction means are perceived by various sensors and deep learning perceptual modules ((0)). The agent build a symbolic representation of its environment by anchoring perceived information to symbolic representations and processes ((1)) according to its prior knowledge. It asks request for a task ((2)). Once it has a determined human's intents through semantic analysis ((3)), goal formulation and grounding to memories ((4)) and behaviors selection ((5)), the agent tries to carry out the task according to its skills ((6)). When executing its skills, the core decision-making IRL process browses the agent's knowledge by checking sequentially or in parallel the conditions c_i (figure B.7) that are grounded to the current perception and world belief. The agent determines if it knows how to solve the task and exploits the corresponding BT before executing it.

When it deals with a lack of information ($\boxed{0}$), however, a failure is generated ($\boxed{1}$), leading to build a request from the interaction state ($\boxed{2}$). The IRL agent then make a request to the human ($\boxed{3}$): learning the missing parts in the skill knowledge ($\boxed{4}$). A failure can be a lack of perceptual (the what) or procedural (the how) knowledge. BT control flow lets us easily design failures handling as they necessary happen in conditions nodes during the execution. We can thus easily and automatically define a new branch in the BT that leverages the known information about the task, the current state

3.3. IRL PROCESS WITH PREFERENCES

of the world and the failure type, in order to have an explainable description of the failure properties. Currently in our architecture, we rely on the mixed initiative teacher/learner setting to overcome failures and incrementally learn perceptual or procedural features to complete new tasks (4).

Humans' identity and preferences knowledge are managed in by the use of a specific preconditions before executing skills. In the current implementation, the IRL agent interacts only with one human at a time and a human teacher must be identified by the system in order to link the teacher id to his preferences. If the system does not know the human, while checking the precondition, the IRL agent asks for some basic information such as the name of the human. A new branch is built in the BTs skill representation. Branch selection is then governed by the current human identifier. The first branch represents a default skill that is used to drive preferences learning. Each branch represents a preference for a different person. Hence, while learning a skill, learned perceptual and procedural information are personalized in function of the interacting human. This is done by leveraging specific preconditions in the procedural skill structure. Therefore, once a skill has been learned, these preconditions allow the agent to branch to the personalized behaviors.

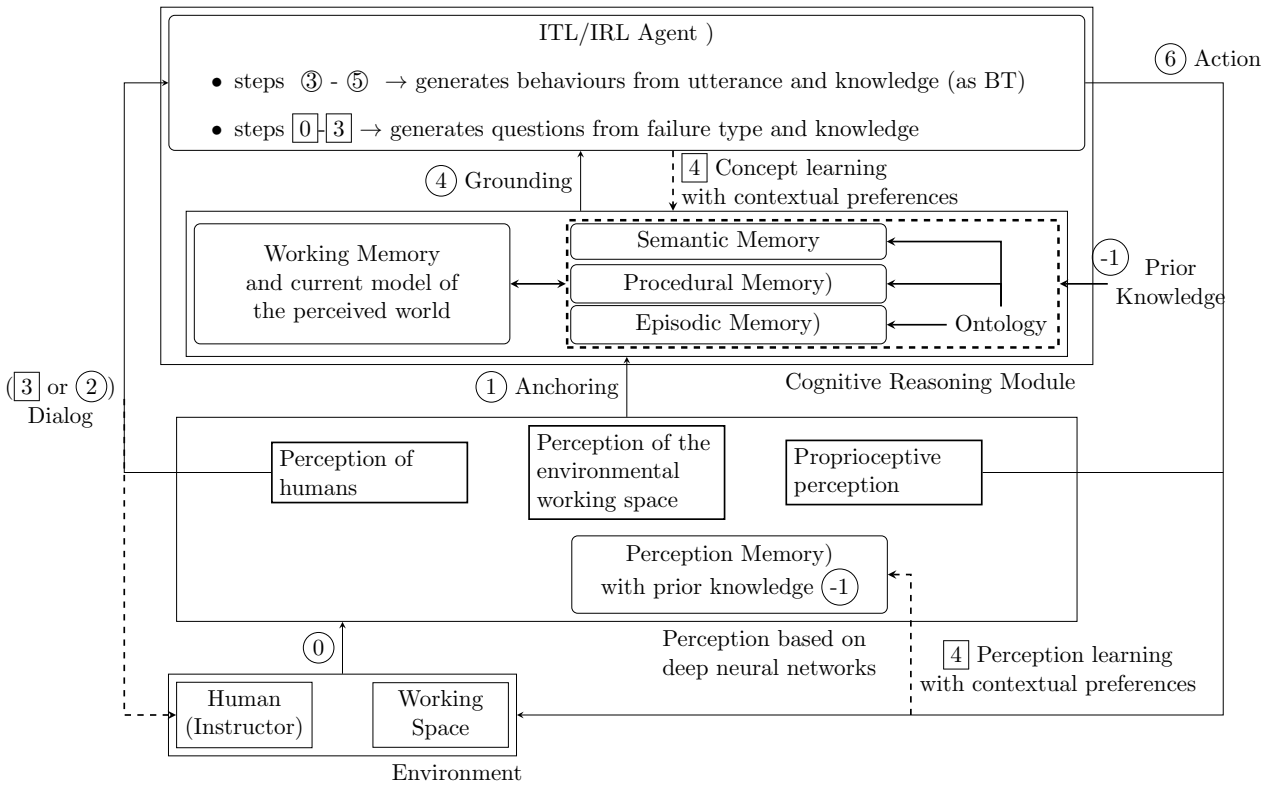


Figure 3.7: It is the same figure as figure 3.1 and is reproduced here for reading convenience.

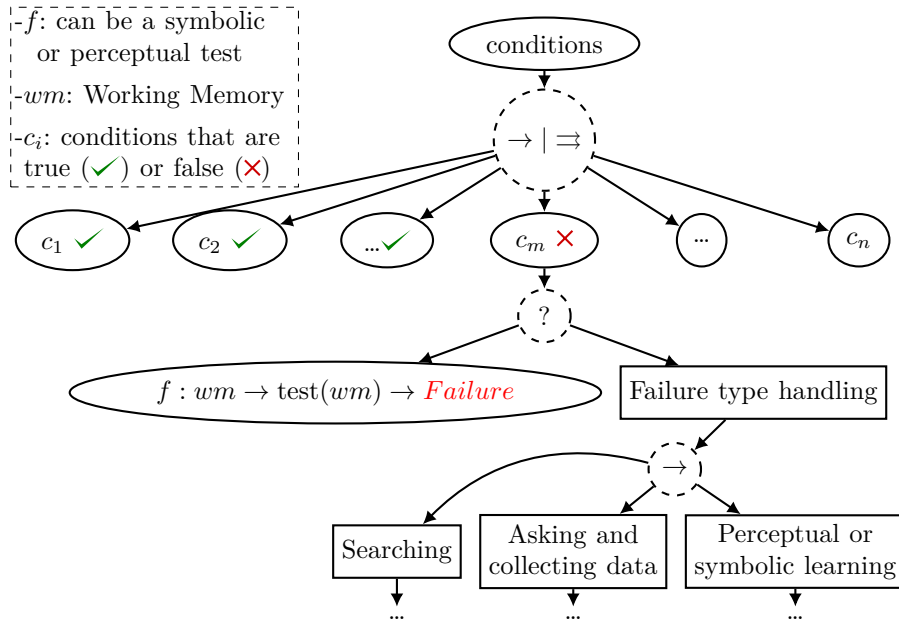


Figure 3.8: The failure handling process triggers interactive learning of symbolic or perceptual representations.

3.4 Conclusion

In this chapter, we introduced the core design of our cognitive architecture for collaborative industrial robotics in the context of IRL. We have described its organization in terms of ontology and hierarchical relational memories. This has led to semantically meaningful, high-level symbolic representations for complexity management and better explainability of the systems behavior. The IRL deliberative process leverages these representations to drive connectionist learning modules, through a goal-driven and mixed initiative human/robot interactive process. Reciprocally, these connectionist modules help anchor lower level data to the higher symbolic representations. By this way, the IRL agent can incrementally learn new task and related skills both at high-level and low level representations. The next chapter (chapter 4) focuses on complementary machine learning approaches that have been used during the thesis in order to develop, with co-authors, connectionist modules for the IRL setting.

3.4. CONCLUSION

Chapter 4

Complementary ML approaches for IRL on planar grasping use cases

Contents

4.1	Learning planar grasping	45
4.1.1	Theoretical general formulation	45
4.1.2	Some common techniques	46
4.1.3	Planar grasping formulation	49
4.1.4	Base approaches in deep learning	49
4.2	Learning autonomously bin picking	51
4.2.1	Bin picking module	51
4.2.2	Methodology	53
4.2.3	Experimental results	55
4.2.4	Module conclusion	55
4.3	Learning grasping location affordance from demonstration	55
4.3.1	Task oriented grasping	56
4.3.2	Methodology	58
4.3.3	Experiments results	61
4.3.4	Module conclusion	69
4.4	Conclusion	69

As the IRL agent learns interactively a task and related skill structure, we saw that it needs to ground its representations by learning from real-world data. In this chapter, we further detail for that purpose, how the IRL agent can exploit complementary Machine Learning (ML) paradigms in a connectionist approach. Following our ITL specifications, we want to exploit modules that allow a fast online learning, from datasets built on the fly, during interaction. Because of the importance of pick and place related tasks in many industrial applications, we focus our use-case on planar grasping related tasks. We first present in section 4.1 the general ML approaches that we leveraged for learning planar grasping relevant parameters. Then, we present two learning modules tailored to grasping related skills. The first one, presented in section B.3.1, leverages a deep reinforcement learning

approach from [1] for autonomous learning of bin picking. We adapted their work for an industrial context. The second and most important contribution, developed in section B.3.2, presents a module for learning task oriented grasping affordance from a few human demonstrations, with respect to our IRL specifications. Individual modules were developed in collaboration with a co-author, Laurent Bimont.

4.1 Learning planar grasping

4.1.1 Theoretical general formulation

We first define some important notions in machine learning to better understand what we mean by learning from data and how these notions are used in the learning modules for planar grasping.

Dataset: Machine Learning (ML) techniques rely on the use of data, and require building datasets. We usually consider an input domain X and a target domain Y . In the IRL setting, we aim to build online predictive models based on a finite dataset $D_{train}(X_{train} \subset X, Y_{train} \subset Y)$, which represent the associative nature of the problem the robot is facing. These datasets can be a mix of prior data and data collected, online, during the interaction.

Risk minimization: A ML model f , aims to learn a map from the input domain X to the target domain Y , given datasets D_{train} . From a very general point of view, this is expressed as a minimization optimization problem.

We would like to be able to predict from $x \in X$ the value $y \in Y$. For that we have to introduce a *loss* (or *cost*) function \mathcal{L} , such as the Mean Square Error (MSE), which measures a notion of distance between predictions $f(x)$ and the real target data y . Then, we can compute a theoretical quantity, the *risk* $R(f)$, which is the expectation of the *loss* function, evaluated for the model, given an infinite amount of data:

$$R(f) = \mathbb{E}_D(\mathcal{L}(f(x), y)) \quad (4.1)$$

A learning problem then consists in finding a model \hat{f} which minimizes $R(f)$.

$$\hat{f} = \arg \min_f (R(f)) \quad (4.2)$$

Of course, in practice and especially in the IRL setting, we are only given a finite amount of data, a training dataset $D_{train}(X_{train}, Y_{train}) = (x_1, y_1), (x_i, y_i), \dots, (x_n, y_n)$. We can only search f^* minimizing

an empirical risk $R_{emp}(f)$ while targeting $R(f)$ minimization.

$$R_{emp}(f) = \frac{1}{n} \sum_{i=1}^n (\mathcal{L}(f(x_i), y_i)) \quad (4.3)$$

$$f^* = \arg \min_f (\mathbb{R}_{emp}(f)) \quad (4.4)$$

When considering a parametric model such as in deep learning, we use specific family of function f_θ parameterized by weights θ . Minimizing the empirical risk consists then in finding θ^* such that f_θ minimize $R_{emp}(f_\theta)$.

$$\theta^* = \arg \min_{\theta} R_{emp}(f_\theta) \quad (4.5)$$

In general \hat{f} is different from f^* . Moreover, as f is trained on finite data, a valid solution for the empirical risk minimization can be to overfit the dataset by simply learning a one/one correspondence between X_{train} and Y_{train} . Such solutions do not generalize to the real risk minimization. Many techniques can be used to limit overfitting, and lack of data for real risk minimization improvements. We discuss in next section some common ML paradigms we used to develop our learning modules.

4.1.2 Some common techniques

There are several learning paradigms in machine learning depending on the problem considered (classification or regression) and how data is used to train the model (supervised learning, unsupervised learning and reinforcement learning). Various principles can be used to improve data efficiency and improve generalization of these learning paradigms, such as transfer learning and data augmentation.

Classification: Given data inputs x and a finite discrete number of classes y , the goal of classification is to separate those inputs and assign a class to each one of them. In the IRL setting, it is linked to categorize perceptual inputs x by assigning a meaningful, human understandable concept (y) to these data.

Regression: Regression allows to learn and predict continuous representations. In the IRL setting, it can be used to explicit non verbal concepts. For instance when grasping a part, the agent can learn to categorize the object (classification task) or directly output a grasping location in the continuous space (regression task).

Supervised: In a supervised learning context, data is collected as pairs of inputs x and output targets provided by a human. In an IRL setting, this is one of the most used paradigm. Target collection is done through Learning from Demonstration (LfD) [2], therefore, these type of learning is costly in terms of human resources. However, as stated in our specification, it is necessary to build a common ground between humans and the IRL agent. This requires the ability for knowledge sharing between humans and the IRL agent.

Unsupervised learning/Self-supervised learning: In an unsupervised learning context, data is collected *without* explicit targets. This type of learning is cheap as the agent do not need supervision. In that setting, data can be grouped by various similarity measures, depending on the nature of the data. Similarity measures can then be used with unsupervised clustering techniques to classify the data without supervision [3]. Learning can also be done through various specific techniques which apply known transformations to the dataset before training a model to reconstruct original data, based on the transformed ones. Ideally, the model learns relevant features leveraging structure in the data. As no labels are explicitly given by humans, the agent learns in a self-supervised manner. The simplest technique is to train the model to predict its input, without transformation. Such model is called an autoencoder [4]. Another common technique is to mask part of the data to predict the remaining data. For instance, in natural language processing, some models are pre-trained on text corpora [5] by predicting hidden words in the text, taking into account adjacent words. In our architecture, we do not exploit directly unsupervised learning but some modules exploit deep learning models which were pretrained in an unsupervised fashion (see chapter 6).

Reinforcement learning: The last major ML learning paradigm is Reinforcement Learning (RL) [6]. RL allows to deal with a sequential decision-making and control problem and is well adapted for robotics. Indeed, in the reinforcement framework, a robotic agent can be controlled in a partially unknown environment without necessarily needing to know its dynamic model (see Figure 4.1). To do so, an agent learns by trial and error after each action A_t and in interaction with its environment, the best way to reach its goal: maximizing the expected cumulative rewards, given reward feedback R_t from the environment when it arrives at state S_t .

Of course, at startup, the robotic agent does not know which state will give which reward. Therefore

the agent needs to *explore* the world first in order to discover what is good or bad for him. On the other hand, it also wants to maximize reward, and therefore have to *exploit* its knowledge of the world. This is referred as the *exploitation/exploration* dilemma. A common choice to deal with this trade-off is the epsilon-greedy method where the agent can choose with a certain time-decreasing probability, a random action instead of the best action according to the current policy.

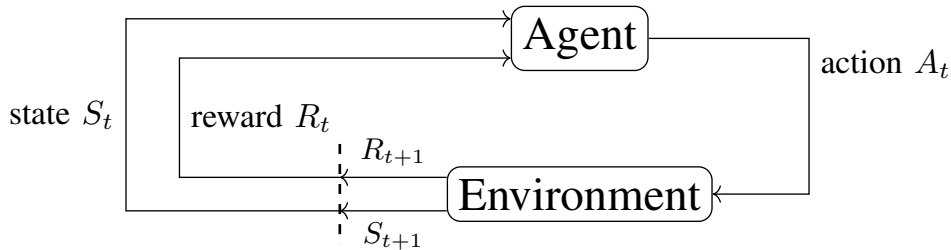


Figure 4.1: Reinforcement learning base description

Transfer learning: Transfer learning [7, 8] is related to any techniques that help transfer knowledge acquired in some domain to another domain. This is useful when a model has to be trained on a domain with few data or costly access to data. In deep learning models this can be done between closely related domains, by pretraining a model on a domain with rich available datasets and/or with unsupervised techniques. By leveraging these prior learned knowledge, another model can learn quickly on a new adjacent domain with much fewer data. For instance, by leveraging hierarchical nature of neural networks, one can extract discriminating features from hidden layers of a model and use them to train another model on a close domain.

Data augmentation: Data augmentation represents a set of techniques which consists in leveraging a dataset, in our context collected during the IRL interaction, and prior knowledge about a task to create artificial data. For instance, in visual classification tasks, classes are often invariant given orientation. In that case, datasets can be augmented with rotation and flipping to account for this invariance. We can also have some knowledge about sensors noise and augment data accordingly.

These different learning techniques are at the core of the connectionist components we can integrate in the architecture. We develop in the next section how they were leveraged to develop planar grasping modules.

4.1.3 Planar grasping formulation

The planar robotic grasping problem aims to find a good set of grasping parameters, given an image \mathcal{I} of a single object as input. For a vertical antipodal grasp, parameters (Figure 4.2) can be expressed as the Cartesian position (x, y) of the tool center point of the gripper in image coordinate system, the angle θ relative to the abscissa of the image, the width opening of the gripper w , the height h (representing the maximum gripper size) and finally the z coordinate of the gripper elevation, in the frame of the local plan on which lie the object: $g = (x, y, z, \theta, w, h)$. According to the task some of parameters of g can be fixed. For instance one can fix the opening of the gripper. An example of common metric to evaluate planar grasping quality is the Jacquard metric (see appendix A.1).

4.1.4 Base approaches in deep learning

Many approaches have been developed to address this vision challenge, based on various techniques such as geometric calculation [9], SVM [10]. Many of them use an underlying deep learning model for robotic grasping. These deep learning approaches can be split up into two categories: quality evaluation of grasp candidates [11] and direct regression of grasping parameters [12].

In the case evaluation of grasp candidates, one defines a quality metric S depending on the grasp parameters and the state of the object [11]. Such metric can be based, for instance, on the ability to lift the object or based on force closure. Then, given grasp candidates, the idea is to train a binary neural network classifier which predict, given the image, if the grasping candidates are good according to the chosen metrics. Through a direct regression approach, several methods have been developed [13–15]. They mostly rely on end-to-end deep learning architecture predicting g from an image \mathcal{I} . Learning

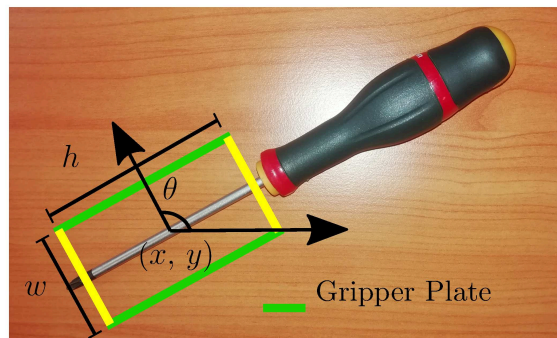


Figure 4.2: Grasping parameters

4.1. LEARNING PLANAR GRASPING

is usually done using online available datasets such as the Cornell dataset [10] (~ 885 images) or the Jacquard dataset [16] (~ 22.000 images) ones. They are composed of vertical RGB views $x \in \mathcal{I}$ of objects with several acceptable grasping parameters g_{true} . When it comes to learning with few data, such as in the online IRL setting, different methods can be used such as decreasing the input space size, using data augmentation or/and using transfer learning to initialise the network. In [17, 18], authors reduced their input space size using only the depth from the RGB-D camera. Data augmentation techniques increase the size and variance of the training set, while Convolutional Neural Networks (CNN) pretrained on image classification tasks are leveraged for transfer learning. The very rich ecosystem of image classification research provides access to a lot of high-performance architectures (VGG [19], Resnet [20], Densenet [21]), pre-trained on large image databases (ImageNet [22], Coco [23]). Despite being trained on RGB images, we can use those architectures for the creation of CNN processing depth images. In the grasping domain, many works have used this technique to create a grasping predictor.

For instance, some co-authors developed a variant of the uni-modal architecture proposed by [14]. This architecture is an end-to-end approach achieving an accuracy of 88.4% on image-wise split of Cornell Grasping dataset. The complete module, called GraspNet, uses VGG16 [24] extracted features (Figure 4.3). Since several grasping parameters g_{true} are available for the same image, training can be made with the minimization of the mean squared error loss function:

$$\mathcal{L} = \min_{g_i \in \mathcal{G}_{true}} (g_i - g_{pred})^2. \quad (4.6)$$

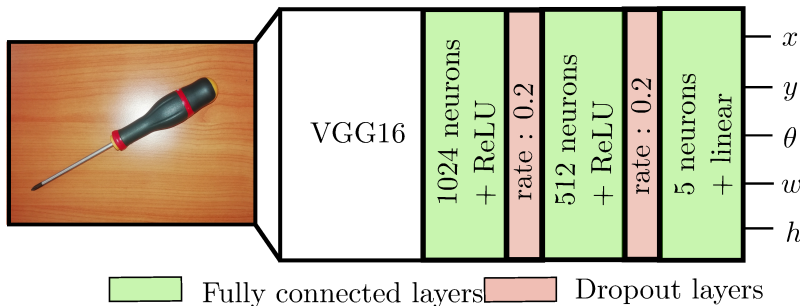


Figure 4.3: *GraspNet* architecture

In the following section, we now explain how we deal with planar grasping in bin picking and task

oriented grasping tasks by developing deep learning modules. For a comprehensive and more general overview on the use of deep learning for grasping, one can refer to [25].

4.2 Learning autonomously bin picking

All tasks in an IRL setting cannot be explained easily in a procedural manner. In that context, we would just like to fix the goal, constrained available actions and let the agent learn intuitively, how to carry out this specific task.

4.2.1 Bin picking module

Collaborative industrial robots often use parallel-plates gripper for manipulating objects. However, in industrial tasks, objects are often cluttered, in highly disorganized heaps such as in bin picking industrial applications. This is a very challenging task because of occlusion, unknown dynamics of objects and noise which limits the use of traditional grasping techniques. Traditionally, a CAD model of a part is used in problems of part gripping. However, it is not always possible to have a model of a part and it can be expensive to make one.

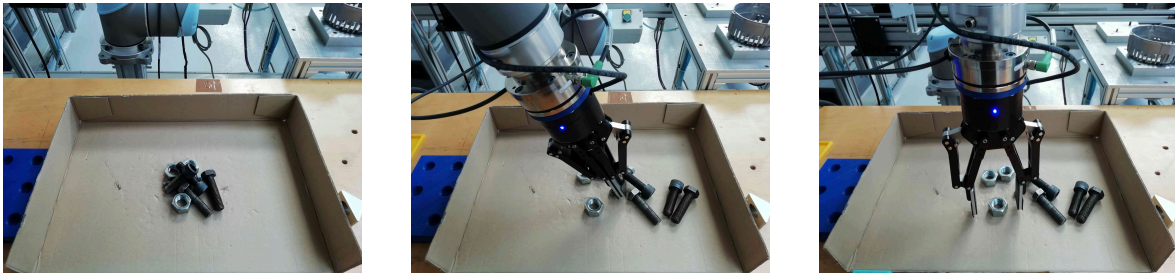
To overcome these challenges, bin picking techniques based on deep learning and reinforcement learning approaches have started to emerge [26], to predict the best grasp given an image of the heap. Yet, because parts are very close, there can still have grasp failures. A solution is to give more action capabilities to the robot. For instance, the IRL agent can be allowed not only to grasp but also to push objects. Pushing can help spread parts in the heap in order to ease future grasping. Authors in [1] proposed to adapt deep Q-Learning [27], a deep reinforcement learning algorithm, to learn grasping and pushing actions. They validated their approach on examples with toys. We developed a module based on an extension of their work, presented in section B.3.1 and validated with screw and bolts as can be found in an industrial setting.

The bin picking use case is a typical example of action where it can be hard to explain procedurally how to carry it out. Actually, explaining in a procedural manner how to carry out such a task is hard even for a human. We could hardly explain why we would spread the heap in one way rather than in another way. Still, we have some goal which is to pick all the parts.

Reinforcement learning fits well to learn such task in an autonomous way. Therefore, we reproduced

4.2. LEARNING AUTONOMOUSLY BIN PICKING

[1] and have extended it experimentally to our industrial context. We addressed the bin picking problems as an autonomous reinforcement learning strategy where the robot agent learns synergies between pushing and grasping as illustrated in Figure 4.4. Moreover, the use of reinforcement and deep learning allows the robot to learn to pick-up parts without the need of any CAD model. This is important as collaborative robots can be expected to work with parts that were not modeled by CAD specialists, especially in small scale industry.



(a) Heap of objects (b) Pushing action to spread parts (c) Grasping of an isolated part

Figure 4.4: Example of synergy between pushing and gripping. A pile of objects is presented none of which can be retrieved by direct grasping (a). The robot will first push the pile (b) and then separate the objects (b) and then grab an isolated object (c).

This work was valorized through a demonstration during the closing day of the European project ColRobot ¹ in the presence of members of the European Commission, various academic partners and industrial partners (Renault and Thalès). A video of this work can be found [here](#) ².

The experimental setup was the following (Figure 4.5): we installed an industrial grade, high definition depth sensor (a photoneo3D camera³) on top of a UR5 collaborative robot equipped with a Robotiq two-finger gripper. For industrial validation, we collected screws and bolts as objects of interest for the bin picking operation and carried out the experiment in a warehouse.

The task is to catch all objects from a cluttered heap of objects present in the workspace. So, the state is represented as an image of the global workspace using the affordance formalism.

¹<https://colrobot.eu/>

²<https://www.youtube.com/watch?v=T592ye7RPxQ>

³<https://www.photoneo.com/products/phoxi-scan-1/>

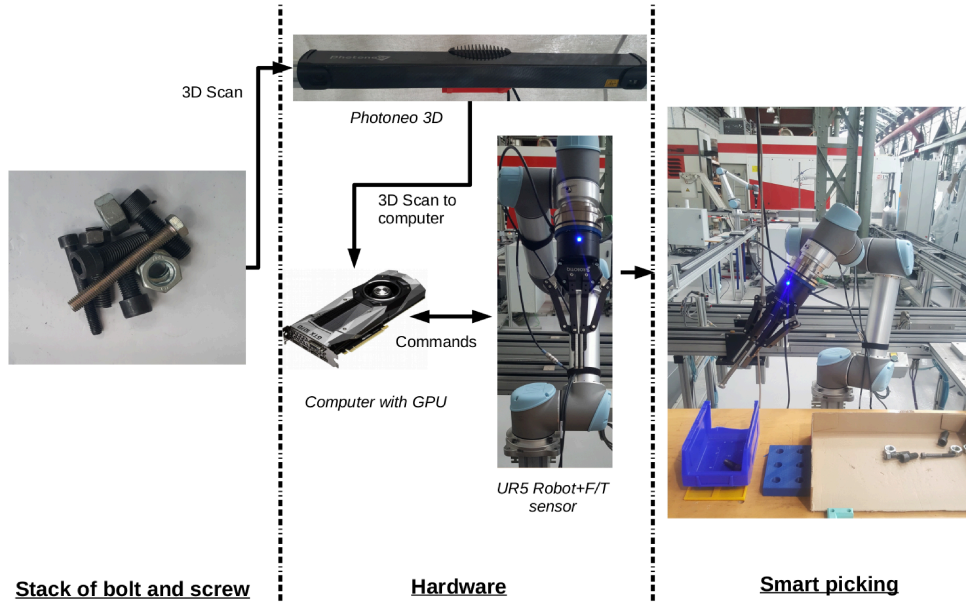


Figure 4.5: Hardware pipeline of the algorithm

4.2.2 Methodology

We reproduced the architecture of the network by implementing a DQN (Deep Q network) algorithm, illustrated in the learning pipeline (see Figure 4.6). The DQN algorithm allows to exploit neural networks to infer Q-maps (see outputs on Figure 4.6) from the state of the workspace. For a given action, a Q-map associates to each pixel of the input image a value determining the quality of the action at the considered location to maximize expected return.

As the state/action space can be large, following [1], we discretize the actions by considering 8 pushing acts, in the plane parallel to the working space, following 8 different directions. In the same way, for gripping, we consider 8 actions corresponding to 8 different angles of rotation of the gripper, always supposed to be perpendicular to the working space. The reward is sparse. Each action is associated to a Q-map. A pretrained neural network is used for transfer learning in order to reduce the amount of needed data.

In [1] for the gripping actions, if an object is picked up then the agent receives a reward of 1, otherwise he receives nothing. Pushing actions are rewarded if there is a change in the image by measuring the euclidean distance between the image before pushing and the image after pushing. As grasping is more important than pushing to solve the task, pushing actions get a 0.5 reward. In

4.2. LEARNING AUTONOMOUSLY BIN PICKING

practice, this has led in some training failures where the model optimized for the wrong objectives. In some training sequences, the robot was likely to push all parts outside of the working space rather than **spreading** parts for better grasping success. This is actually not surprising given the pushing reward function; pushing actions which spread parts outside of the working space lead to a high distance between successive images. In order to have better stability in the learning process, we investigated a simple change of the reward function. As we want to spread object to have room for grasping, a better reward function is to actually compute a metrics telling how well parts spread apart after pushing. For that we compute objects dispersion, between two images after pushing. If dispersion increases, there is a 0.5 reward, else there is no reward.

At each cycle, the action with the highest Q-value is selected among the Q-maps. In order to explore environment, during the *training* phase, an epsilon-greedy strategy is leveraged with a decreasing probability over training. After a few hours, the robot is able to pick-up and store a bunch of screws and bolts without any CAD model of the parts.

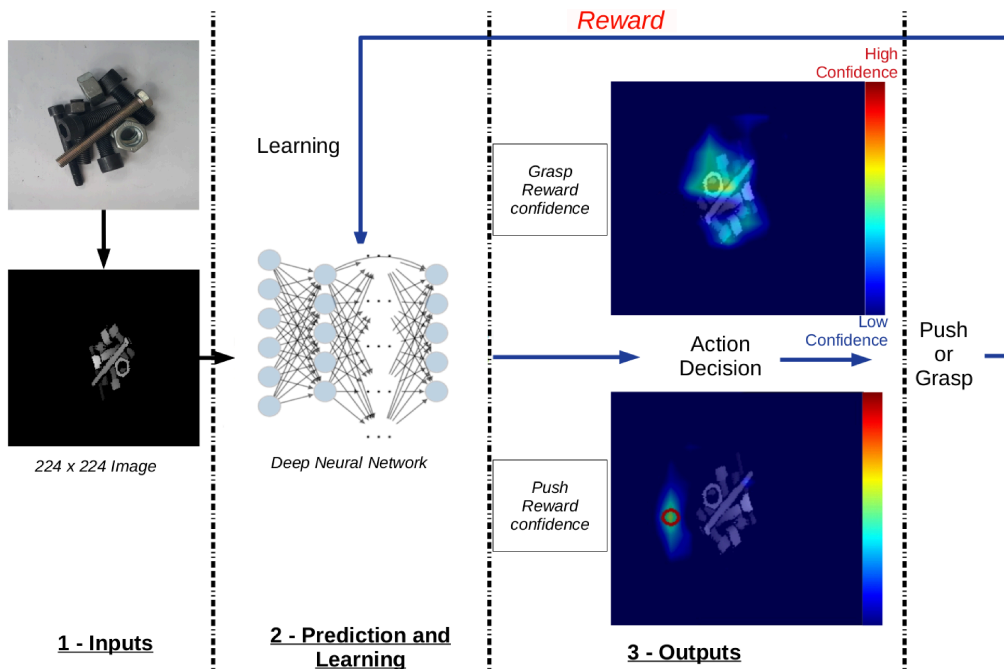


Figure 4.6: Pipeline of the algorithm

4.3. LEARNING GRASPING LOCATION AFFORDANCE FROM DEMONSTRATION

4.2.3 Experimental results

The results presented in [1] are as follows: the completion rate (i.e., all objects in the workspace have been removed) reaches 82.7% and the rate of pushing actions that were followed by successful grasping is 60.9%. Once trained, we obtained close results in terms of performance. On our system, the best performance is achieved after 2000 moves, with a success rate of 82%. On our implementation, this represents a time of about 11h (20 seconds per iteration). We made several training and we qualitatively observe less training failures with the updated reward function.

4.2.4 Module conclusion

In the end, the agent improves its performance over time and is able to adapt to objects it has never seen autonomously. However, there are some limitations. During our tests, we noticed that the robot can learn to catch coins in an unstable way depending on the first successes. For example, in some cases the robot learned to take the screws by the net, which shows the difficulty of developing a good reward function given a target goal. Indeed, taking the screws by the net was seen as a success since the robot was indeed taking "something". On the other hand, in an industrial context, objects may have specifications and should be captured in very specific way according to the object and the task. This motivates the fact that this type of learning is not enough.

There should be more interaction between the operator and the robot during the learning process, so that the robot can learn to adapt to the specific needs of the operators, always with our IRL specifications in mind.

4.3 Learning grasping location affordance from demonstration

With this module, we investigate the problem of an operator wanting to configure a robot to grasp an industrial object in a specific area. Our motivation is to create a fast learner grasping system which does not require any databases, CAD models or simulators, so that it can be easily reconfigured by the operator himself, which is a non programming expert. The transfer of knowledge from the operator to the robot is done through the most natural interaction: manual demonstrations of authorised and prohibited grasping locations (Figure 4.7).

This work was valorized in the following conference article [28]. A synthetic video of presentation

4.3. LEARNING GRASPING LOCATION AFFORDANCE FROM DEMONSTRATION

can be found on [youtube](#)⁴

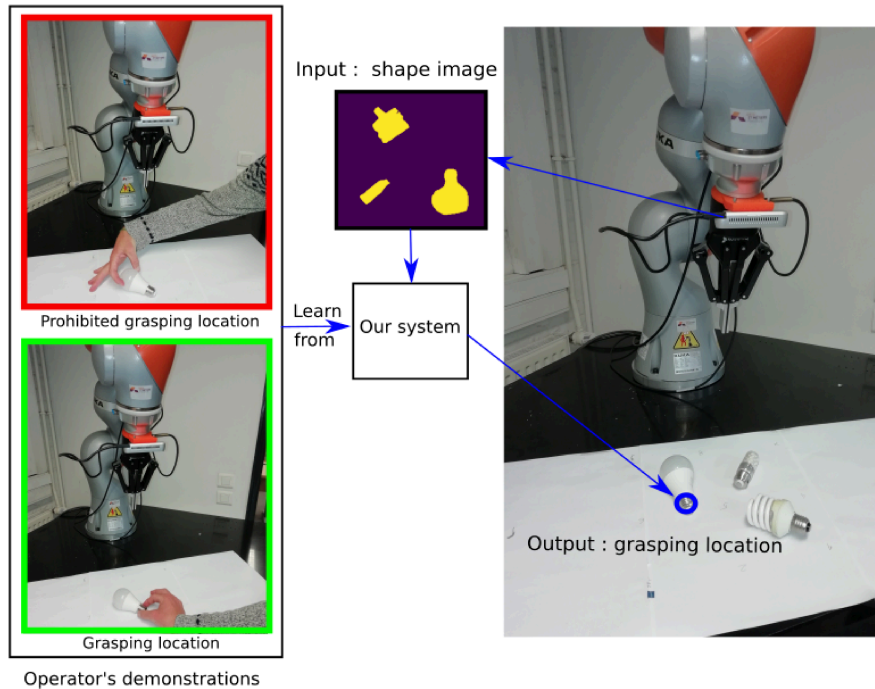


Figure 4.7: First, for a new object, the system learns from operator's demonstration. After few minutes of training, the system will be able to retrieve the demonstrated area on a depthmap.

4.3.1 Task oriented grasping

In task oriented grasping, the IRL agent faces different use case:

- grasp a tool by the handle or head depending on the task the robot has to perform.
- grasp an industrial object so that it can be placed in a chosen orientation as part of a pick and place operation
- grasp a fragile object in a safe area.

The use of Convolutional Neural Networks (CNN) partly solves this problem since they offer great results for object recognition and grasping [29–31]. However, they come with constraints such as the use of specific databases (Cornell database [29]) or huge training time [32].

⁴<https://www.youtube.com/watch?v=1UjehV1RCLc>

4.3. LEARNING GRASPING LOCATION AFFORDANCE FROM DEMONSTRATION

In Table 4.1 , we provide a summary of some task-oriented grasping works and compare them with our approach in section B.3.1-II.

Table 4.1: Summary of existing task oriented grasping works

Ref	Data generation	CAD model & simulator	Observation
[33]	From CAD model	✓	Training took 6 hours on Titan X GPU
[34]	From simulation	✓	1.5 M of data are generated for training
[35]	ShapeNet and ModelNet40	✓	Bayesian Optimization
[36]	RGB-D Part Affordance Dataset	×	Large dataset
[37]	From few views	×	20 minutes to reconstruct 3D mapping of object
our [28]	From few demonstration	×	< 5 minutes of training on RTX 2080 GPU

Task-oriented grasping uses the concept of affordance introduced by Gibson [38] which describes parts of objects according to their functional utility. In robotics, this concept is used for gripping and handling objects considering the work to perform afterwards [39, 40]. A task-oriented grasper can be created using behavior grounded affordance [34] or spatial maps [39] for instance. The semantic labels technique on images can also be applied [33, 41, 42] : using specific large datasets such as UMD [36] or shape database [35], each pixel is labelled independently according to the part of the object to which it belongs. Our work uses semantic labelization of images without databases, learning from a few examples demonstrations.

Demonstration learning can be used to transfer knowledge from an operator to a system, in our case to teach a robot a precise grasping location. Most previous works address this problem with a trial and error phase via a simulator or directly on the real system (Table 4.1). In [43], the authors propose a network architecture and data augmentation pipeline to design a controller able to grasp very simple objects (cube, cylinders. . .) from a single demonstration. However, the controller can not integrate important constraints into task-oriented grasping, like prohibited locations while defining such constraint is relevant to increase safety by putting emphasis on what can be done and what cannot be done.

In [37] authors learn a dense descriptors map for objects after building a 3D dense reconstruction

4.3. LEARNING GRASPING LOCATION AFFORDANCE FROM DEMONSTRATION

model of the object. As a result, they obtain a semantic representation of the object allowing them to grasp the desired location. In our work, we decided to work directly on images without any 3D reconstruction techniques which may take time.

4.3.2 Methodology

We study the problem of performing an antipodal grasp perpendicular to a planar surface, on a specific object for which an operator has taught authorised/prohibited grasping areas. A RGB-D camera is mounted on the robot’s wrist and capture a fixed height top view.

We define a pixel-wise semantic segmentation pipeline, based on grasping from a few demonstrations methods, where the input is a depth image of the scene and the output is the grasping parameters $\mathbf{g} = (x, y, z, \theta)$ for grasping the object on the demonstrated area. Coordinates (x, y, z) represent the tool-centre of the gripper, and θ is the angle of the gripper in the plane. Grasping parameters are directly derived from the image segmentation. We define a structure of our pipeline allowing fast training from a few demonstrations. This problem creates constraints that motivated the design of our pipeline:

- learn fast authorised/prohibited grasping areas
- generalize from a few demonstrations.

4.3.2-I Training dataset:

Data Capture: Training is done directly from an operator’s demonstration without using any external databases. The operator’s thumb and index fingers are covered with coloured pads so that they can be easily identified by the camera. Grasping gestures on authorised and/or prohibited areas are stored by recording the fingers coordinates (Figure 4.8 -a).

Then (Figure 4.8 -b), a 2D shape of the object is obtained from the binarization of the depth image: the table’s pixels are set to 0, and the objects’ pixels (above the table) are set to 1. We note it as $I \in \{0, 1\}^{n \times m}$ where (n, m) are the dimensions of the image. Labels are generated as images $L \in \{-1, 0, 1\}^{n \times m}$ where authorized pixels have value of 1, 0 for pixels without information and -1 for prohibited pixels. The use of 2D shape I reduces the size of the input space and contributes to our goal of generalization. If necessary, several demonstrations of the same object in different locations

4.3. LEARNING GRASPING LOCATION AFFORDANCE FROM DEMONSTRATION

are collected and stored as tuples (**I**, **L**). Note that demonstration can be done in several ways like with a computer interface. We chose to use the operator’s fingers to benefit directly from human-robot interaction without any hardware intermediate. In the remainder of this article, objects with/without demonstrations (i.e. for which we trained or not a specific network) will be referred to as *referenced/unreferenced* objects.

Data Augmentation: The number of tuples (**I**, **L**) is then increased by random translations and rotations (Figure 4.8 -c). We also randomly erase some areas of the input image to give variability in the training data, to reflect both the noise of the camera’s depth sensor and the objects’ shapes variations with its relative position according to the camera.

In an industrial context, the solution also needs to be robust in a dynamic setting such as lighting conditions and background variations. Since input 2D shapes **I** do not incorporate any brightness, color and texture information, our pipeline is robust provided that effects of such variations on the resulting 2D shape are reflected in the data augmentation.

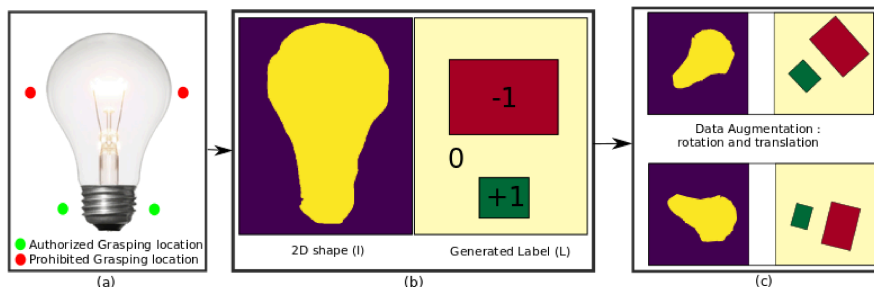


Figure 4.8: Data capture and data augmentation pipeline.

4.3.2-II Network pipeline:

Architecture: To map **I** to **L**, we address this problem as a regression one, and use a CNN composed of a partial pre-trained Densenet 121 [44] and a light CNN. The overall architecture is presented in Figure 4.9 -b, activation functions are "RELU" except at the output where we use the "tanh" function to distribute the values between -1 and 1 . Dropout rates are set to 40% to prevent the network from over-fitting and to generalize well to unseen data. This small convolutional network only has 6914 parameters and can be trained with our training dataset in a few batches using an appropriate loss function.

4.3. LEARNING GRASPING LOCATION AFFORDANCE FROM DEMONSTRATION

Output: Once trained, this network outputs a grasping affordance pixel-wise representation $\mathbf{G} \in [-1, 1]^{n \times m}$ (Figure 4.9 -c). To determine the grasping parameters g , we post-process the raw output \mathbf{G} . First, all pixels belonging to the table (0 in \mathbf{I}) are set to 0 in \mathbf{G} . Then, to determine grasping parameters, we select the highest grasping affordance pixel $(u_g, v_g) = \arg \max_{(u,v)} \mathbf{G}(u, v)$ as the target tool-centre point. The grasping angle α in the image’s frame is calculated by performing a PCA on a sub-region of the input centred around the grasping point (Figure 4.9 -d). Finally, geometrical transformations based on hand eye calibration are made to convert (u_g, v_g, α) to \mathbf{g} . Therefore, the robot can directly perform a grasping action in its workspace.

We also tried to address this problem as a classification one, where output G provides class probabilities (prohibited/neutral/authorised) for each pixel. Tests have shown lower performances than those of the regression approach, with convergence issues in some cases.

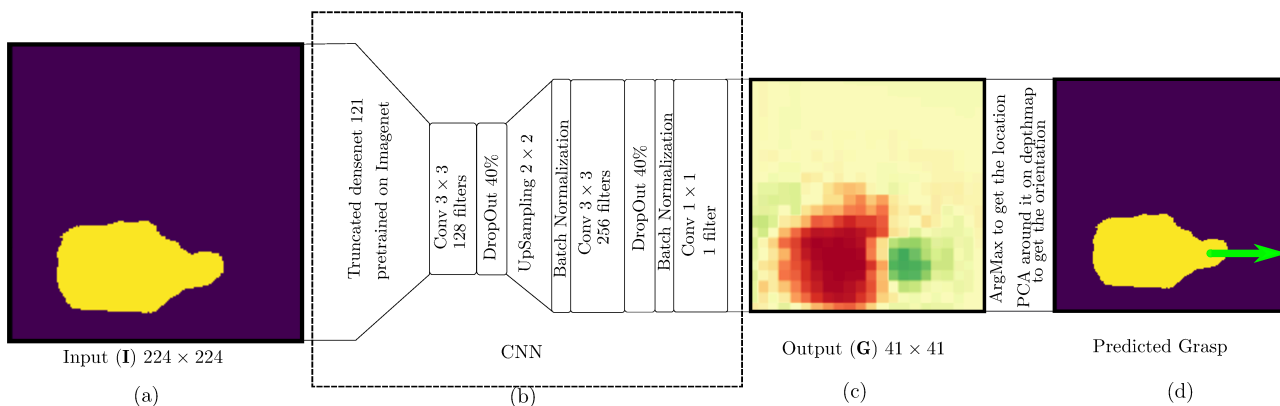


Figure 4.9: Overview of our CNN pipeline.

The **Loss function** (L) should allow our pipeline to generalize from a few demonstrations and to classify a pixel-wise representation with unbalanced area sizes (as shown in Figure 4.8 -b , prohibited, neutral, and authorised areas have different sizes). In order to accomplish this, we introduce a modified version of the pixel-wise L2-loss function [45] by multiplying each pixel error by a specific weight:

$$\mathcal{L}_{weighted-L2} = \frac{1}{n \times m} \sum_{i=1}^{n \times m} \omega_{i,label_i} (pred_i - label_i)^2 \quad (4.7)$$

The weighted factor $\omega_{i,label_i}$ is chosen as follows:

4.3. LEARNING GRASPING LOCATION AFFORDANCE FROM DEMONSTRATION

$$w_{i,label_i} = |pred_i| + \lambda_1 \times \frac{1}{N_{label_i}} \quad (4.8)$$

where N_{label_i} represents the number of pixels in \mathbf{L} containing label value $label_i$. The first component $|pred_i|$ in Equation 4.8 used to focus the network’s attention on interesting parts by focusing the gradient descent over areas of interest. The second component is used to accentuate learning over underrepresented areas of the label map by reducing the importance of large areas. The parameter λ_1 balances the two components. The benefit of this weighted loss function is studied in section 4.3.3-IV. To prevent overfitting, we use the L2-regularization loss $\mathcal{L}_{L2\ reg}$ applied on the weights and bias of the network. The finale composite loss-function is:

$$\mathcal{L} = \mathcal{L}_{weighted-L2} + \lambda_2 \mathcal{L}_{L2\ reg}$$

We trained the network using stochastic gradient descent, with a learning rate of 10^{-4} , a momentum of 0.9, $\lambda_1 = 20$ and $\lambda_2 = 5.10^{-5}$.

4.3.3 Experiments results

To evaluate the proposed algorithm, we plan a series of experiments. We studied 5 points:

1. grasping *referenced* objects at the right area in different positions
2. the benefits of our modified $L_2 - loss$ function
3. the benefits of using both authorised and prohibited demonstrations
4. the ability of the algorithm to generalize to *unreferenced* similar objects
5. performing grasp in an environment composed of several *unreferenced* similar objects

4.3.3-I real-world experiment:

We use a modified Python version of the Matlab Kuka sunrise toolbox [46] to control a Kuka iiwa LBR 7 DOF robot equipped with a Robotiq 2F-140 gripper. The robot’s workspace is a 30×30 cm flat square. An Intel[®]Realsense[™] Depth Camera D415 is mounted on the wrist of the robot. Computations were made on a PC with an Nvidia RTX 2080 8Gb graphic cards and Intel[®]8 Core[™]

4.3. LEARNING GRASPING LOCATION AFFORDANCE FROM DEMONSTRATION

i7 9700K 3.6 GHz CPU. The implementation was done in Python 3.6 using Tensorflow 1.13. We train a network using online data augmentation generating 1600 tuples randomly. On average the training lasts 250 seconds per object.

4.3.3-II Grasping in the right area:

We measure the ability of our network to find a specific grasping area learned during the demonstration phase. In Figure 4.10, we present our panel of *referenced* objects with the name of their grasping areas. We focus on simple industrial objects.

Protocol: For each object, we train a specific network from 1 to 3 demonstrations of the same authorised grasping area (with eventually a prohibited grasping area) under different object positions. Then we evaluate the network’s ability to find those areas on 36 unseen positions of the *referenced* object. The evaluation is done by placing the object at 9 points of the workspace and by rotating it in 4 orientations (0° , 90° , 180° and 270°). A grasp is considered valid each time the object is caught by the authorised area.

Results & Discussion: Our pipeline achieved good results (Table 4.2) with only one demonstration. For bulb, screw and pliers, the grasp success in the authorised area is over 90%. For socket wrench (81% and 86%) and cup (70%), the decrease in performance comes respectively from the geometric similarity of the authorised/prohibited grasping area and a more complex geometry. Adding 1 or 2 demonstrations from other positions seems to solve that issue. For socket wrench and cup grasping, results raised over 90% of success with 2 demonstrations. In these worst cases, we suppose that data augmentation does not reproduce efficiently the different possible views of the shape of an object. Prediction quality evolves depending on the input image. In Figure 4.11, we can see different cases where the system outputs a good, an average and a bad segmentation of the object. Bad segmentation occurs when the current shape is very different from the demonstrated one. It shows a limitation in the generalization abilities of our system.

4.3. LEARNING GRASPING LOCATION AFFORDANCE FROM DEMONSTRATION



Figure 4.10: Objects used for our experiment, with the name of grasping areas. Green colour (resp red colour) denotes authorised (resp prohibited) grasping area. The yellow colour is used to illustrate authorised/prohibited area and vice versa, depending on the task (for example the accessibility of the screwing operation). For the pliers, two different authorised areas are tested separately

Object	Area	Number of demonstration(s)		
		1	2	3
Socket wrench	handle	81% (29/36)	92% (33/36)	94% (34/36)
	head	86% (31/36)	97% (35/36)	100% (36/36)
Pliers	handle	97% (35/36)	100% (36/36)	100% (36/36)
	head	92% (33/36)	97% (35/36)	97% (35/36)
Bulb	foot	97% (35/36)	100% (36/36)	100% (36/36)
Cup	handle	70% (25/36)	92% (33/26)	97% (35/36)
Screw	head	97% (35/36)	100% (36/36)	100% (36/36)

Table 4.2: Results of our grasping test, percentage (number) of good grasps over the 36 unseen positions

4.3. LEARNING GRASPING LOCATION AFFORDANCE FROM DEMONSTRATION

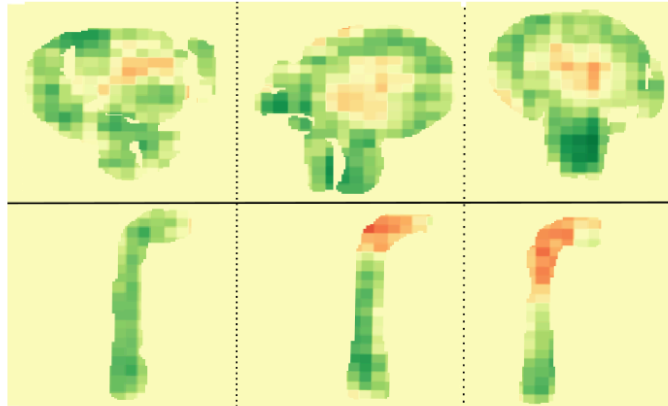


Figure 4.11: Different segmentation quality results. The top images (cups) were trained showing one authorised grasping area (handle). The bottom images (socket wrench) were trained showing one authorised (handle) and one prohibited (head) grasping area. The left column shows bad segmentation resulting in a bad grasping decision, the middle one shows average segmentation, and the right one shows good segmentation of the object. For the purpose of illustration, outputs were reoriented and resized. Those outputs were obtained using networks trained on 3 demonstrations.

4.3.3-III Benefits of learning both authorised and prohibited areas: We measure the benefits of indicating both authorised and prohibited areas on objects.

Protocol: We train our pipeline with exactly the same training data than in section 4.3.3-II, but without prohibited areas (by replacing -1 by 0 in the labels L), and we compare performances on the same inputs.

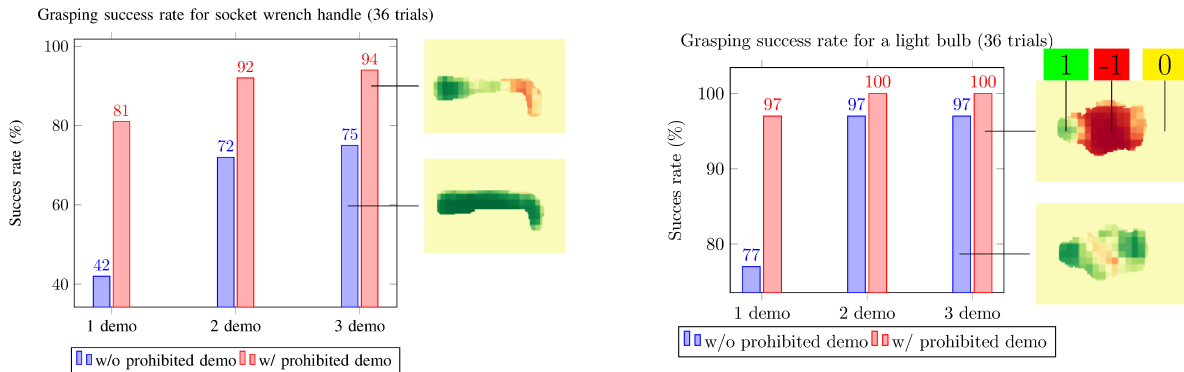
Results & Discussion: In Table 4.3 and Figure 4.12 we present the grasping success rate for this experiment. In parenthesis, we recall the grasping success obtained while training with both authorised and prohibited areas. Performance is less than or equal to that of the experience in section 4.3.3-II. The rate drop is particularly significant when only one demonstration is provided or for the socket wrench caught by the handle. Indeed, for the socket wrench, the similarity between the handle and the head makes a task-oriented grasping more difficult, by learning only from authorised area. This shows a first relevance of indicating both an authorised and prohibited areas. This relevance is even more important when comparing qualitatively networks' outputs trained with and without a prohibited area. In Figure 4.13, on the top row (without prohibited areas), the semantic segmentation is inaccurate resulting in values close to +1 for undesired areas (instead of value close to 0). Indicating prohibited areas is, therefore, a safety guarantee since their outputs tend to have negative values (bottom row). As a result, even if the network fails, the robot will tend to catch an object in a neutral area rather

4.3. LEARNING GRASPING LOCATION AFFORDANCE FROM DEMONSTRATION

than in a prohibited one, thus avoiding to damage fragile objects.

Object	Area	Number of demonstration(s)		
		1	2	3
Socket wrench (Figure 4.12a)	handle	42% (81%)	72% (92%)	75% (94%)
	head	77% (86%)	97% (97%)	100% (100%)
Bulb (Figure 4.12b)	foot	77% (97%)	97% (100%)	97% (100%)

Table 4.3: Comparing grasping results with only one authorised area and with both authorised/prohibited areas (in parenthesis)



(a) Results for the wrenches handle

(b) Results for the bulbs

Figure 4.12: Illustration of grasping results with respect to the number of demonstrations

4.3.3-IV Benefits of the weighted L2 loss function: We measure the impact of our weighted loss function by comparing networks trained with our proposed loss function and with its non weighted version.

Protocol: We followed the same protocol as presented in section 4.3.3-II and we only changed the training method by setting all weights $\omega_{i,label_i}$ to 1 (regularization parameter λ_2 remains equal to 5.10^{-5}). Training is performed using 3 demonstrations and results are compared with the corresponding ones of the previous experiment.

Results & Discussion: Table 4.4 highlights that adding our weights in the L_2 cost function has improved the system performances in every case. Moreover, without weights, we noticed that bad behavior could occur: the training fails to focus on the important area (+1 and -1) and does not converge to a solution. This experiment validates the relevance of our proposed loss function.

4.3. LEARNING GRASPING LOCATION AFFORDANCE FROM DEMONSTRATION

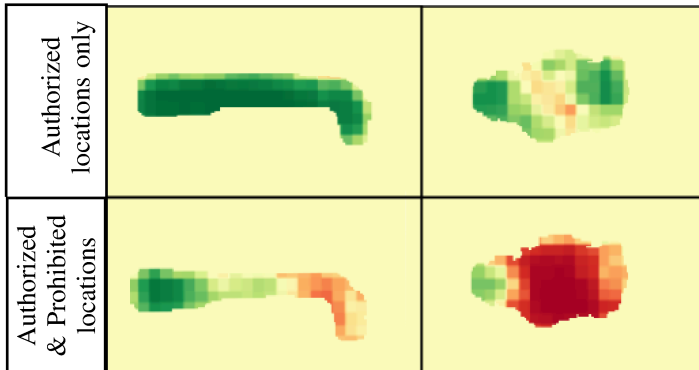


Figure 4.13: Qualitative comparison between pipeline output. Networks were trained using 3 demonstrations

Object	Area	L_2 loss	weighted L_2 loss
Socket wrench	handle	61% (22/26)	94% (34/36)
	head	86% (31/36)	100% (36/36)
Pliers	handle	92% (33/36)	100% (36/36)
	head	72% (26/36)	97% (35/36)
Bulb	foot	94% (34/36)	100% (36/36)
Cup	handle	75% (27/36)	97% (35/36)

Table 4.4: Influence of the weighted L_2 loss. Training was made with a set of 3 demonstrations.

4.3.3-V Grasping similar unreferenced objects: We measure the ability of our algorithm to generalize the grasping area learned for an object to another similar one.

Protocol: We used specific networks trained in section 4.3.3-II to perform the test with similar unreferenced objects (Figure 4.14). Similar objects were close from the referenced one varying in their size and geometry.

Results & Discussion: In Table 4.5 , except for bulb 2, we can observe a good ability of the network to generalize grasping area to other similar unreferenced objects with no degradation of the performances compared to those obtained on the referenced one. By taking a closer look at the different bulbs grasping affordance pixel-wise representations (Figure 4.15), the demonstrated knowledge is

4.3. LEARNING GRASPING LOCATION AFFORDANCE FROM DEMONSTRATION

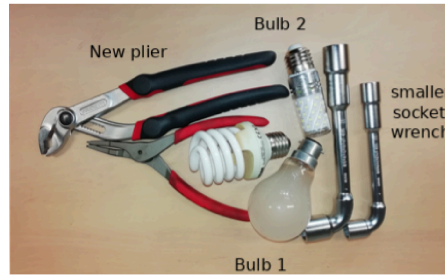


Figure 4.14: Similar objects used to test our algorithm generalization abilities

quite well transferred to bulb 1 with a precise segmentation of the unreferenced object. However, the shape of bulb 2 is quite different from the one of the referenced bulb, and the segmentation gives an inaccurate result, especially for the authorised area where grasping affordance values are low. In an industrial context, if a grasp fails for a similar object, then a (or a few) demonstration(s) can be done by the operator to train a specific network for this object (i.e.g bulb 2).

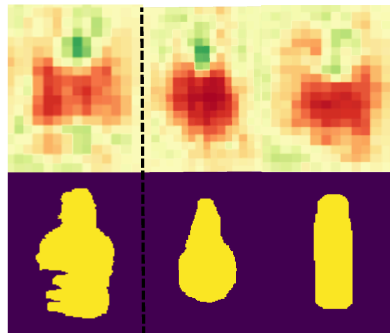


Figure 4.15: Bulbs grasping affordance pixel wise representations. From left to right : *referenced* bulb on which the training was done, bulb 1, and bulb 2.

Object	Loc.	Number of demonstration(s)		
		1	2	3
Smaller wrench	hand.	78% (81%)	92% (92%)	94% (94%)
New plier	hand.	97% (97%)	100% (100%)	100% (100%)
Bulb 1	foot	100% (100%)	92% (92%)	97% (97%)
Bulb 2	foot	89% (100%)	78% (92%)	81% (97%)

Table 4.5: Ability of the Network to generalize to an *unreferenced* similar object. The network was trained after demonstrations on a *referenced* object and the test was performed on *unreferenced* similar objects. In parenthesis, results obtained in section 4.3.3-II are recalled.

4.3. LEARNING GRASPING LOCATION AFFORDANCE FROM DEMONSTRATION

Comparison with other works presented in section 4.3.1: In some other works, the ability to generalize has also been quantified. In [33] authors achieved a grasping success rate between 69% and 82% while in [34], authors obtained results between 71.1% and 86.6%.

In an industrial situation, those methodologies are not applicable since it would require the operator to generate a 3D CAD model for each new *unreferenced* object. In [47], generalization ability was not quantified explicitly and requires more than 20 minutes to create a precise 3D representation of an object. However it requires more than 20 minutes to create a precise 3D representation of an object.

4.3.3-VI Grasping several similar objects: Grasping several identical or similar objects laying on a workspace surface is an important skill to the industry. It allows thereafter to place these objects in boxes for example. We measure the ability of our pipeline to grasp several similar objects laying on a surface while being trained with only one of them.

Protocol: Experiments are done on three different groups of similar objects (Figure 4.16) with their corresponding specific network trained from 3 demonstrations. For each group, several objects are placed in the robot’s workspace without touching each other. We let the robot grasp the objects one by one until the workspace is cleared. When a failure occurs, we remove the object manually. We repeat this process with other object’s configurations until we reach 20 attempts for each group.



Figure 4.16: The three groups of similar objects used in our test.

Results & Discussion: Table 4.6 shows performances slightly under those of individual objects. It is a promising result as our proposed algorithm, despite being trained with one object in its workspace, achieves relevant actions when several spaced similar objects are presented to it. We also evaluate the grasping capacity of our system when objects are in contact with each other. However it leads to failed grasping predictions because of very different 2D images. This highlights the limitation of our 2D representation in a cluttered environment, for example in the context of a bin picking task.

4.4. CONCLUSION

Bulbs	Socket wrench	Screws
19/20 (95%)	17/20 (85%)	19/20 (95%)

Table 4.6: Grasping accuracy results for groups of similar objects

4.3.4 Module conclusion

In this work, we have shown that a fast reconfiguration of a grasping robot is possible with one (or a few) demonstration. Furthermore, our proposed pipeline is able to generalize grasping strategies for several unreferenced similar objects. Our method combines a reduced state space, a light CNN and a weighted loss function. It is able to quickly learn from few data without requiring any datasets, CAD models or simulations. Our CNN network pipeline fulfills our initial motivation of creating a task-oriented grasping system that can be fastly and easily reconfigured by an operator. Moreover, the learning of prohibited areas, makes this process safer. Thus, it shows a good potential for integration in an industrial context.

This work is a powerful module that the IRL architecture can leverage for better teacher/learner interaction and for affordance location learning.

However, it presents limits that suggest further work. The selected input space limits our algorithm to simple 2D shapes. Working directly with the depthmap from the RGB-D camera will allow to consider more complex 3D objects and cluttered environments. In addition, the semantic segmentation of objects might be erroneous in some cases. Detecting these failures would allow to ask the operator for help when needed in a continuous learning scenario.

4.4 Conclusion

We have presented two modules in this chapter that leverage different learning paradigms according to the IRL situation. Given some reward function and constrained actions, we can teach an IRL agent how to carry out some tasks in an autonomous way. This is useful when teaching the procedure is hardly explainable. However, the behavior of the agent depends on how well the reward is defined, which is a hard task. Moreover, the learning duration can be quite long (several hours) for an online IRL setting. We have studied and developed a learning from demonstration module [28], which can learn fast and from a few natural demonstrations, object grasping affordance, with respect to our initial

4.4. CONCLUSION

specifications. A comprehensive experimental validation of this module has been made highlighting the interest of this approach. Finally, each module can be used for specific needs, and could be integrated and even coupled in the wider architecture in order to carry out more complex tasks. Integration and validation of the specific learning from demonstration module is discussed on chapter 6. But at first, we can observe that without a notion of known and unknown, these models will predict grasping location even on objects, or objects views far from the training ones. This could lead to potential wrong predictions and thus, to risky behaviors. In order to improve robustness, modules should also be able to account with the uncertainty of their predictions, which is investigated in next chapter (chapter 5).

Chapter 5

Learning under uncertainty

Contents

5.1	What does uncertainty means ?	73
5.1.1	Stochasticity and data shift	73
5.1.2	Aleatoric uncertainty τ_a	79
5.1.3	Epistemic uncertainty τ_e	80
5.1.4	Total uncertainty τ	80
5.2	Uncertainty estimation methods	81
5.2.1	Bayesian and variational inference methods	82
5.2.2	Deep ensemble methods	83
5.2.3	Distance aware uncertainty model	85
5.2.4	External measure	88
5.2.5	Conclusion on uncertainty review in deep learning	90
5.3	Uncertainty for decision-making	94
5.3.1	Calibration and sharpness	94
5.3.2	Performance metrics of a system	97
5.4	Active learning setting	100
5.4.1	Uncertainty for active learning setting	100
5.4.2	General active learning process for ITL	101
5.4.3	Uncertainty aware behavior model	103
5.5	Conclusion	106

Until now, we have described skills learning abilities in terms of symbolic and connectionist components. We showed that a lack of symbolic knowledge about a task, in terms of procedural or perceptual information, leads to a *failure* and triggers a mixed initiative, interactive learning event. However, this is not sufficient as perceptual modules, such as modules based on deep neural networks, can be brittle when facing new situations. As a consequences, failed predictions can lead to erroneous decisions. The IRL agent needs to know the level of certainty or uncertainty in its perceptual and reasoning

processes. This is a key indicator to endow the IRL agent with more insight about what it knows, what it does not know and what it is not certain about before taking a decision.

This chapter first develops in 5.1, through examples, the main principles of uncertainty that can be used for an IRL agent. We further illustrate in 5.2, several estimations techniques to derive uncertainty in learning modules. A focus has been made on the main underlying uncertainty principles and the state of the art for deep learning techniques. Then, we introduce how this uncertainty can be leveraged for decision-making (section 5.3) as a base for active learning (section 5.4). We specifically introduce in section 5.4.3, how uncertainty can be integrated in the IRL architecture by extending the behavior model. Uncertainty integration for active learning in our architecture is an on going work, which is further investigated in terms of short term perspectives in chapter 7.

5.1 What does uncertainty means ?

5.1.1 Stochasticity and data shift

Some machine learning models, especially in deep learning, give only prediction values without relevant tools to assess for the quality of these predictions. However, many physical phenomena are inherently stochastic and the lack of experiments and therefore of training data can lead to underconfident or overconfident predictions. As in most machine learning problems, we can distinguish categorical classification problems and regression ones.

For classification, a model should output a label with its confidence. For instance, asking a deep neural network to predict a result with 100% of confidence for a "head or tails" when flipping a coin is meaningless. We would like the network to be uncertain about its prediction and ideally, to output a distribution of possible outcomes ($1/2 \rightarrow \text{head}$, $1/2 \rightarrow \text{tails}$). In a general way, for a classification problem, uncertainty should output a prediction distribution over the classes. By this way, confidence in a predicted class, as well as particular confusions with other classes, can be highlighted. Even if softmax outputs of a classification network looks like such a distribution, they are known to be prone to miscalibration and overconfidence [1]. Consequently, they can not be trusted as a confidence and uncertainty measure.

For regression, a model should output a mean value with its variance which can be interpreted as a confidence interval around the predicted value. For instance, in our grasping location learning module, we mitigated risks of grasping in forbidden area, by learning a separating neutral area. Even in that case, the self-occluded cup example presented in previous chapter (Figure 4.11), shows that failures are still possible when the agent has not seen enough demonstrations, for some of the objects. The agent needs a way to estimate the relevance of the predicted authorized grasping location.

An IRL agent is most likely, especially in its infancy, to face things it does not know rather than things it does know. This is related to the fact that most of the world is unknown and that new perceived data can be far from the known world. In the machine learning literature, it is referred as *Out-of-Distribution (OOD)* robustness.

Currently, most deep learning systems assume an *Independent and Identically Distributed (IID)* data setting. Therefore, the network assumes test data domain D_{test} and train data domain D_{train}

5.1. WHAT DOES UNCERTAINTY MEANS ?

are taken from the same distribution: $p((x, y) \in D_{test}) = p((x, y) \in D_{train})$, with y the target value and x the input features. In classification problems, y is a class label commonly represented as the vector of the theoretical softmax output (1 for the class of the input, 0 for other classes). Concerning regression, y is the vector (scalar) of real output(s). Same distribution assumption, however, is regularly broken as in most real-world settings, D_{test} is a mixture of a train and OOD domain D_{OOD} leading to $p((x, y) \in D_{test}) = p((x, y) \in D_{train} \cup D_{OOD}) \neq p((x, y) \in D_{train})$.

$p((x, y) \in D_{test})$ can differ from $p((x, y) \in D_{train})$ in various biased ways that can appear simultaneously. This has resulted in an active fields of research in OOD robustness specification and mitigation. We list below some of the main types of dataset shifts found in the literature:

- *Covariate shift* occurs when distribution of features $p(x)$ changes but $p(y|x)$ is fixed. Often covariate shift occurs when x causes y . For instance, in classification based on vision, this can be related to a change of view, noisy data such as illumination of a previously learned object: the raw image or computed features vary while the object label remains the same. In the case of a regression problem, such as in our module for affordance location prediction, a change of light or view change features x and therefore might affect prediction. This kind of shift, if detected, can be mitigated by collecting relevant data such as more examples or data augmentation as proved our approach in [B.3.2](#).
- *Label shift or prior distribution shift* [2] occurs when distribution of label $p(y)$ changes and $p(x|y)$ is fixed. Often label shift occurs when y causes x , so when we try to learn an inverse model that is not stationary across time or space. Such setting has been studied a lot in medical setting for disease diagnostic modelling. For instance in [3], authors argue that one can train a binary classification model $p(x, y)$ to predict the diagnostic y to have flu based on symptoms x . The distribution $p(y)$ of flu prevalence (the number of case at a given time) varies throughout the year, but as symptoms are caused by the flu, $p(x|y)$ does not change. However, $p(y|x)$ does change, as given the same symptoms x , it is more likely to have flu y during an epidemic (i.e. $p(y|x)$ increases). Therefore, if the model has been trained on data outside of the epidemic, it might underestimate flu prevalence during the epidemic. In our case, let's imagine an ideal IRL agent observing the behavior of a human to induce, among several tasks, the current procedure he is working on. For instance, it could learn an intent classification model, predicting the task

5.1. WHAT DOES UNCERTAINTY MEANS ?

intent y based on some observed features x such as facial expressions and human movements. Because a task procedure is well-defined, one can assume that the task specification causes x and that $p(x|y)$ is mostly fixed. However, across time, tasks relative frequency, and thus $p(y)$, can change according to demand or supply chains.

- *Open-set-recognition issues* occurs when new classes appears at test time. An overview on recent advances in this field is presented in [4]. For an IRL agent, for instance, it occurs when new objects are learned.
- *Subpopulation shift* occurs when a model has to generalize at test time to new *sub*-classes that were not seen in training [5]. For instance, if a network has been trained to recognize different tools, but that the dataset contains some biases such as only *blue* screwdrivers, the model might underperform when trying to predict the class of a *red* screwdriver.

Standard deep learning networks are not efficient and usually overconfident given these shifts [1], especially for deep neural networks using ReLU activation function [6] (almost all modern architectures). Overall, a good uncertainty estimation and taxonomy could help in quantifying the confidence that one can have in the IRL architecture predictions. Moreover, it could serve as a *quantified* basis to help cope with those biases. From the human perspectives, it could improve safety guarantees and acceptability which are strong requirement for industry. From the IRL agent side, it is a way to question and reason about its own behavior in a flexible way. In the rest of the chapter, we will focus on dealing with covariate shift and open-set-recognition as they are the most studied shifts.

There is an extensive literature on uncertainty taxonomy (see [7] for a comprehensive survey). We briefly describe the main characteristics of an uncertainty metrics. First, in general, uncertainty τ of machine learning model can be decomposed in two types of uncertainty, *aleatoric* (τ_a) and *epistemic* (τ_e) uncertainty.

An intuitive way and quite general way to obtain this decomposition, in the supervised learning context, is through the traditional decomposition of mean square error into bias, variance and noise [8, 9] (equation 5.2). Given a finite dataset $D(x,y)$ with x the inputs and y labels, let's assume that there is some data generative process h and a zero-mean noise $N(x)$ such that $y = h(x) + N(x)$. Ideally, we would like to approximate h by learning a function f given a rich family of function from some hypothesis space. In case of deep learning it is a function f_θ parameterized by weights θ . The classical

5.1. WHAT DOES UNCERTAINTY MEANS ?

training method is to minimize the mean squared error $(y - f_\theta(x|D))^2$ for the training dataset and to expect generalization for test data. Because of the noise, of limitations of the chosen family function and of non infinite data, learning h is near impossible in real world use case. In general, the learning algorithm can produce many f_θ that can be compatible with the dataset D . We can then compute the mean predictor μ_f between different model solutions given the data and decompose the error in terms of uncertainty.

$$\mu_f(x) = \mathbb{E}_\theta[f_\theta(x)|D] \quad (5.1)$$

The total expected error of the model given the data is then computed as [8, 9]:

$$\begin{aligned} \mathbb{E}[(f_\theta(x) - y)^2|D] &= \mathbb{E} \left[\underbrace{f_\theta(x) - \mu_f(x)}_V + \underbrace{\mu_f(x) - h(x)}_B - \underbrace{N(x)}_N \right]^2 | D \\ &= \underbrace{\mathbb{E}[V^2|D]}_{\text{Model variance}} + \underbrace{\mathbb{E}[B^2|D]}_{\text{Biases}} + \underbrace{\mathbb{E}[(N - 0)^2|D]}_{\text{Variance of noise}} \end{aligned} \quad (5.2)$$

τ_e τ_a

Let's illustrate uncertainty estimation with simple examples. We first develop the case of classification which has been the most studied. The different method are extensible to regression task which is briefly illustrated in section 5.2.5.

For classification let's exploit the traditional two moons distribution datasets with an additional out-of-distribution data cluster. We based this work on the following libraries: `uncertainty_baselines`¹, Edward2 [10], Tensorflow Probability [11], which add high-level probabilistic layers to Keras and Tensorflow [12]. The dataset was produced with scikit-learn [13] utility tools (Figure 5.1a presents the dataset). There is an additional OOD cluster that the network cannot see during its training phase. We train a neural network based on ResNet to classify the dataset in two classes using a distance aware learning method to reproduce work presented in [14]. This method is explained in section 5.2.3 and is use here for illustration. The method is called Spectral-normalized Neural Gaussian Process (SNGP). This network is compared with a traditional ResNet deep network that is not trained with uncertainty handling in mind. Once training is done, we can plot for all points in the plan, the class probability

¹<https://github.com/google/uncertainty-baselines>

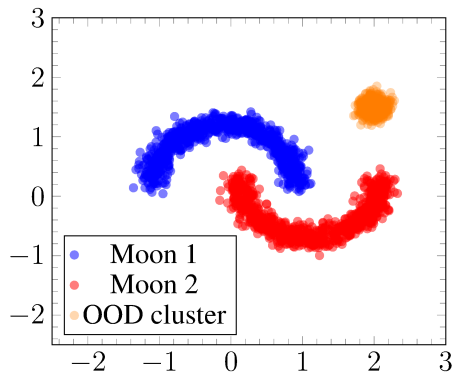
5.1. WHAT DOES UNCERTAINTY MEANS ?

p and the corresponding predictive uncertainty $p(1 - p)$ (variance for a Bernoulli distributed random variable). It is then possible to analyse how confident the network is about its prediction.

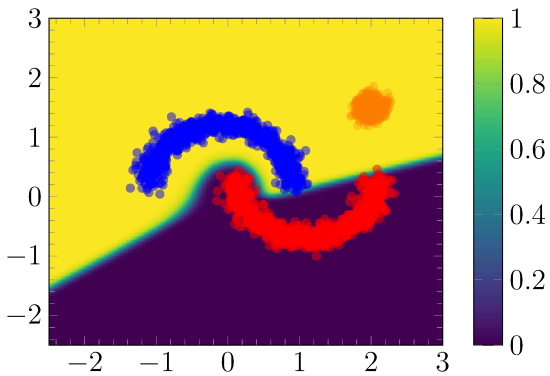
Figure 5.9a, and Figure 5.9b represents respectively the probability prediction and the predictive uncertainty for the ResNet network with a traditional training procedure. Figure 5.9a shows that as the network has not be trained with dataset shift awareness, it simply learns to separate the plan in two regions. Unfortunately, Figure 5.9b shows that the network, is also highly overconfident in its prediction, except at the separation boundary. It is no surprise that the third cluster is classified with high confidence as belonging to the blue moon (upper moon on the Figure 5.9a).

On the other hand, Figure 5.1d, and Figure 5.1e illustrate the probability prediction and the predictive uncertainty for an uncertainty aware based on a residual network ([14]). We can observe in this setting that now the model learns to classify data by proximity and not by merely cutting the plan in half. Therefore, even if the third cluster is still classified as belonging to the blue moon, it is now associated with an uncertainty which increases as the cluster get farther from the moons.

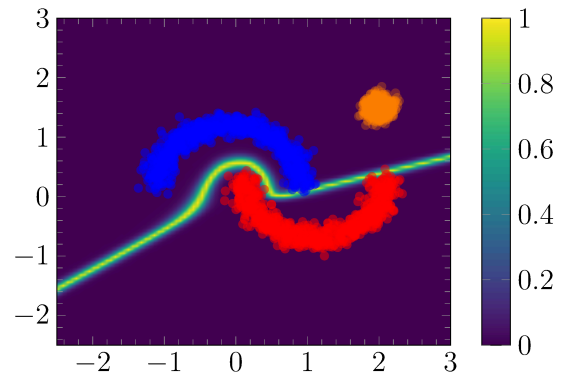
5.1. WHAT DOES UNCERTAINTY MEANS ?



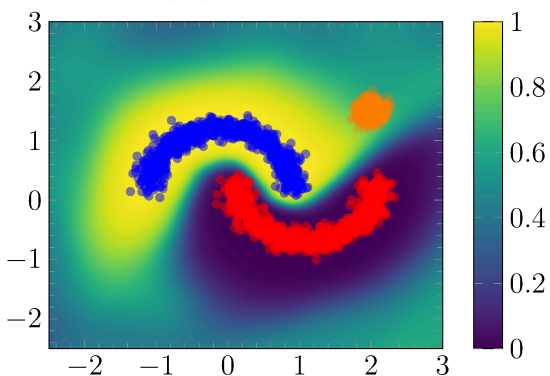
(a) Two moons distribution dataset with an OOD cluster



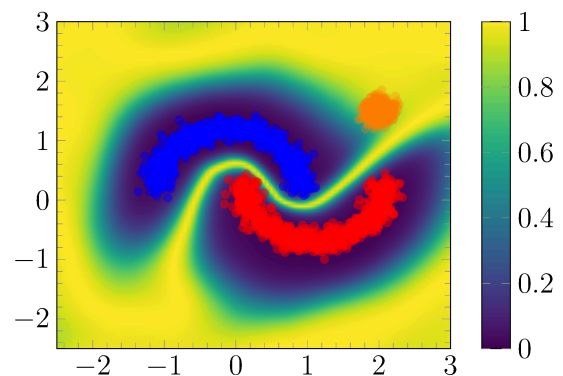
(b) Two moons classification: standard predictions for a trained ResNet [15]



(c) Two moons classification: standard uncertainties for trained ResNet



(d) Two moons classification predictions with the SNGP (Spectral-normalized Neural Gaussian Process) [14] uncertainty method



(e) Two moons classification uncertainties with SNGP

Figure 5.1: Behaviors of deep learning classification models given the two moons distribution dataset with an additional OOD cluster, with and without uncertainty aware leveraged techniques.

5.1.2 Aleatoric uncertainty τ_a

Aleatoric uncertainty refers to the inherent stochasticity of physical phenomena. This is the *irreducible* part of uncertainty as it is not linked to IRL agent representations but it is purely related to nature. In other words, even with an unlimited amount of data, it is not possible to reduce this uncertainty. For instance, aleatoric uncertainty about the coin flipping problem could no go below $1/2$. In the case of the two moons distribution dataset, the aleatoric uncertainty, denoted τ_a is associated with the variance of data around the *true* moon represented in dark blue in Figure 5.2. The network, whatever the amount of train samples, will never be able to predict a sharper representation of the moon without better sensing or without additional assumptions such as the physical nature of the noise.

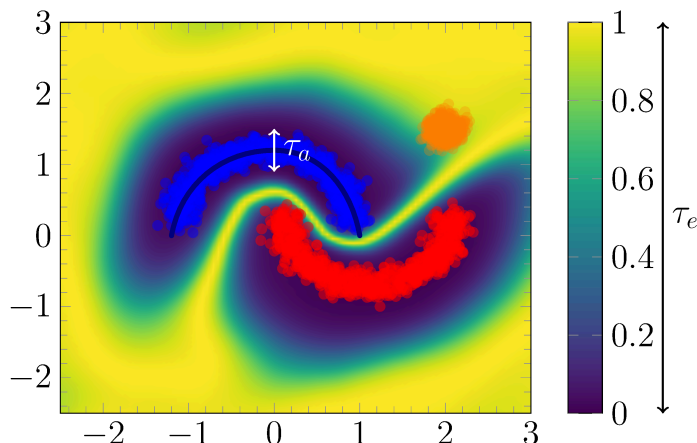


Figure 5.2: Two moons classification with an uncertainty aware model (here SNGP). We illustrate the aleatoric uncertainty τ_a and the learned epistemic uncertainty τ_e given the two moons dataset.

The notion of aleatoric uncertainty can even be further decomposed in heteroscedastic and homoscedastic uncertainty given a model.

Homoscedastic uncertainty: it is the part of uncertainty that stays constant for different inputs and through time. For instance aleatoric uncertainty in head and tails does not depend on the coin and date. It will be the same for all instance of coins on earth and at every time.

Heteroscedastic uncertainty: it is the part of uncertainty that depends on model inputs and can vary with time.

5.1. WHAT DOES UNCERTAINTY MEANS ?

Aleatoric uncertainty can be learned from data. For instance, in [16], authors learn heteroscedastic aleatoric uncertainty in visual segmentation tasks as a loss attenuation by computing the following loss for the neural network f_θ :

$$LNN(x, y, \theta) = \frac{1}{N} \sum_{i=1}^N \frac{1}{2\sigma(x_i, \theta)^2} \|y_i - f_\theta(x_i)\|^2 + \frac{1}{2} \log \sigma(x_i, \theta)^2, \text{ where } \sigma^2 = \tau_a \quad (5.3)$$

By minimizing this loss over weights θ and σ (which is an output of the network), they are able to learn implicitly heteroscedastic uncertainty in various visual segmentation tasks.

5.1.3 Epistemic uncertainty τ_e

Epistemic uncertainty on the other hand is the part of uncertainty that is linked to the inner model ignorance, not to the physical underlying process. Therefore, this uncertainty can be *reduced* by collecting more data and by updating the model. In Figure 5.2, we represent the epistemic uncertainty, τ_e which is associated with the predictive uncertainty of the model. Far from the moons, aleatoric uncertainty is low, therefore epistemic uncertainty can be supposed equal to the predictive uncertainty.

5.1.4 Total uncertainty τ

The total uncertainty for an input x is then $\tau(x) = \tau_a(x) + \tau_e(x)$. Ideally, as the IRL agent collect more data, it should be able to reduce its epistemic uncertainty $\tau_e(x)$ to zero. The only remaining uncertainty is then the irreducible aleatoric uncertainty. If the predicted physical phenomenon is deterministic, then $\tau_a(x) \approx 0$. If it is stochastic, then $\tau_a(x) \gg 0$.

In Figure 5.3, we can observe as expected that epistemic uncertainty is reducing, during training over epochs, around data close to training domain while increasing on unseen points. As expected, at the end of training the epistemic uncertainty is almost zero close to data, only aleatoric uncertainty remains.

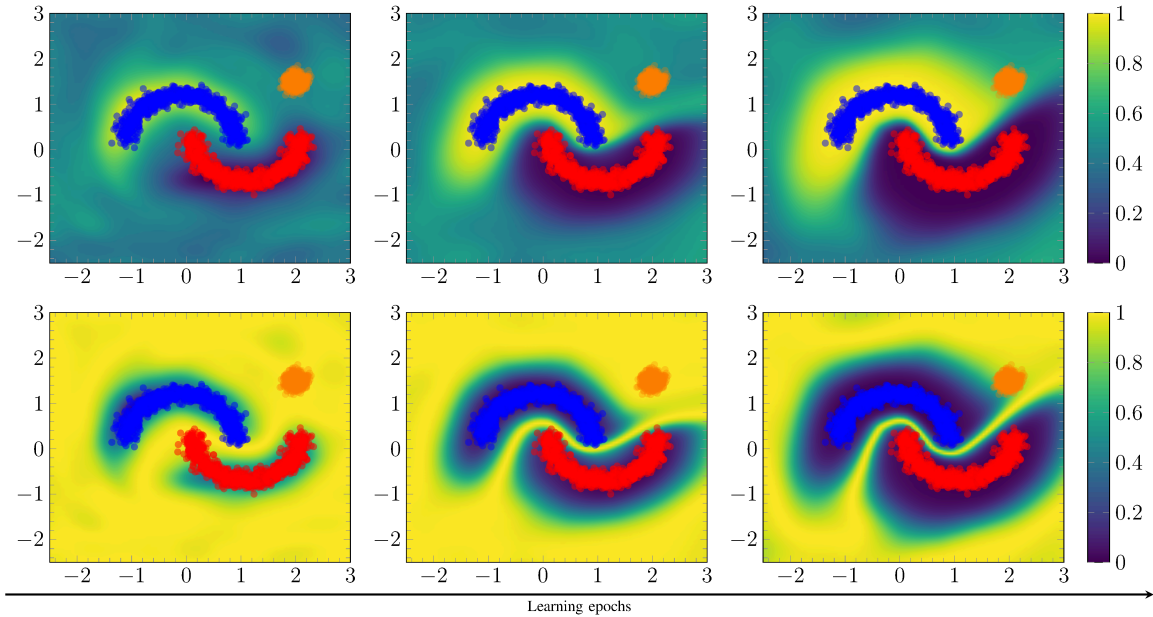


Figure 5.3: Two moons classification: evolution of prediction (top) and uncertainty (bottom) with time (learning epochs)

5.2 Uncertainty estimation methods

Representing a neural network uncertainty is an open topic in deep learning. A good uncertainty metric τ should be least intrusive by limiting the amount of architectural modification. Ideally, it should not decrease the performance and additional computation requirements should be minimized. In the literature, several approaches are being investigated to learn such models. The main techniques are based on models which rely on external measures and the ones that learn their own uncertainty. In the latter, we can further distinguish :

- bayesian based learning techniques
- ensemble methods
- distance aware model
- external measure

5.2.1 Bayesian and variational inference methods

In deep bayesian learning methods (illustrated in Figure 5.4), one model $f_\theta(x)$ of parameters θ is explicitly built to learn an output distribution $p(y|x, \theta)$ of mean $\mu(x)$ and variance $\sigma(x)^2$ based on a given input $x \in \mathbb{R}^m$ (see equation 5.4). This is done through different variational inference techniques:

$$\mu(x) = \mathbb{E}_{p(\theta|x,y)}(p(y|x, \theta)) \quad \text{and} \quad \tau(x) = \sigma(x)^2 = \mathbb{E}_{p(\theta|X,Y)}(p(y|x, \theta) - \mu(x))^2. \quad (5.4)$$

Standard variational inference techniques for deep learning [17], such as stochastic variational inference or sampling based on variational inference, learn a Gaussian distribution at each weights of the neural network, see Figure 5.4a. If the target value y is a vector composed of N components, the uncertainty metric is given as the mean value of the variances over each components σ_i : $\tau = \frac{1}{N} \sum_{i=1}^N \sigma_i$. With a mean μ and a standard deviation σ , the amount of required parameters is doubled compared to standard networks. Therefore training such networks, can be computationally difficult as it requires much more data and memory. For this reason, some works learn a distribution only at last layer, see Figure 5.4b. This kind of network is then referred as proper scoring networks [18]. In this approach, each neuron on the output layer learns a Gaussian distribution $\mathcal{N}(\mu_i(x), \sigma_i^2(x))$, with $\tau_i(x) = \sigma_i^2(x)$ output both, for each component i of target vector y , a prediction $\mu_i(x)$ with its uncertainty σ_i^2 . As each neuron learns a mean and a variance, it doubles the number of outputs parameters. Training of these Gaussian distributions is done using the negative likelihood loss function (equation 5.5). Learning $\log(\sigma^2)$ is sometimes prefer to σ^2 for numerical stabilities.

$$\mathcal{L}(y, (\mu(x), \sigma(x))) = \frac{1}{N} \sum_{n=1}^N \frac{\log \sigma_n^2}{2} + \frac{(y - \mu_n)^2}{2\sigma_n^2} \quad (5.5)$$

These methods require a change of architecture, but many other approximation methods, like Monte Carlo dropout for instance, have been developed allowing to minimize architecture changes.

Monte Carlo Dropout method, proposed by [19] makes this approximation by using Dropout layers [20]. Dropout layers are already widely used in standard neural architectures to prevent overfitting by deactivating a neuron with a probability p . Performing several inferences with active dropout layers on the same input x^* is equivalent having an ensemble of networks sharing some weights in contrast to ensemble presented in 5.2.2 and is illustrated in Figure 5.5a. This technique is easy to implement since it requires minor changes in the original architecture or no changes if dropout is already used.

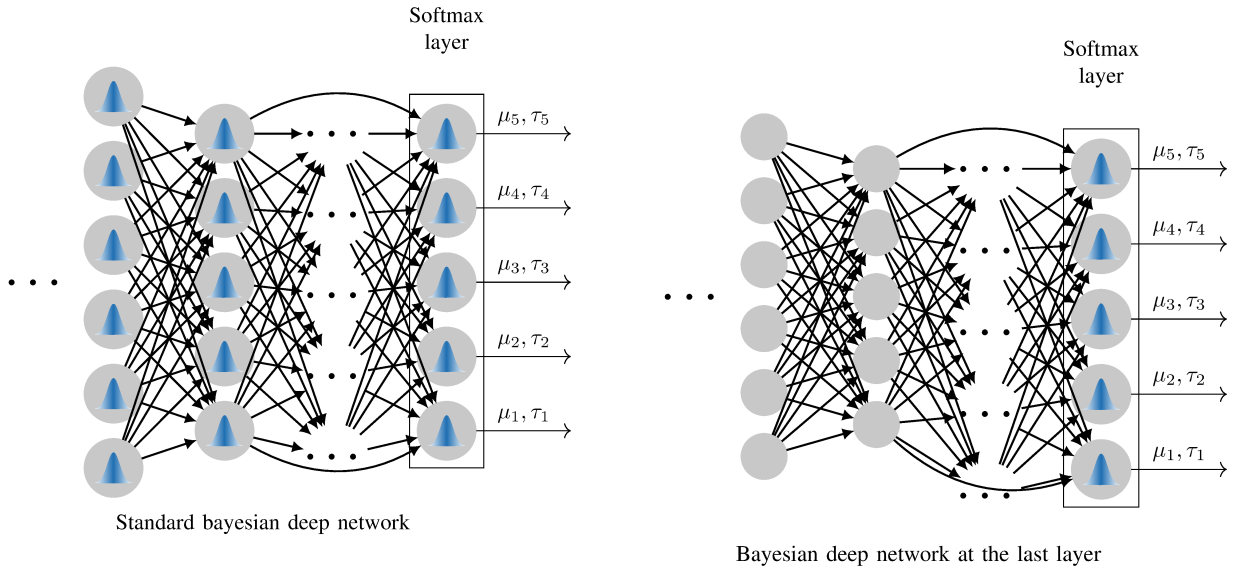
Reported performances of this approach in the literature are not even. While working well in some cases as regression task [19] or image segmentation [21], this approach does not perform well in others like in some active learning scenario [22] or classification’s failure prediction [23]. An updated version called Monte Carlo Concrete Dropout [24] consists in learning the dropout parameter p during the training process. Each dropout layer is replaced by a continuous concrete distribution relaxation allowing to compute gradient and to tune parameter p . This technique requires to replace common Dropout layers by custom ones, however the learning scheme remains unchanged. It has been shown to perform slightly better than standard dropout in uncertainty estimation [24].

In complex tasks, dropout approach might be too simple and underperforms. More recent works, such as SWAG (Stochastic Weight Averaging-Gaussian) [25], take a different approach by exploiting the space of solution during the gradient descent optimization process. The key idea of SWAG is to leverage iterations of Stochastic Gradient Descent (SGD) via a specific learning rate schedule [26] based on Stochastic Weight Averaging (SWA). Learned weights at the end of each iteration are seen as samples from a Gaussian distribution. We illustrate this idea in Figure 5.4c. Authors store the network weight parameters θ and the **average** weights parameters θ_{SWA} and weight covariance $\bar{\theta}$ over different epochs. The weights covariance matrix is then exploited to provide a measure of uncertainty. In that setting only θ is learned during training, and $\bar{\theta}$ can easily be computed as a moving weights average. In practice, that means that computation overhead is low. Memory requirements is tripled but additional variables, θ and $\bar{\theta}$ can be stored outside of the GPU memory according to authors [25].

Dropout methods are less memory and computationally intensive than standard variational inference methods. However, as several forward passes are required to estimate uncertainty, the inference cost can still be prohibiting in high rate demanding tasks such as in vision. Most recent approaches, like SWAG, reduce the amount of computational power while still being competitive but it requires learning an architecture from scratch, preventing the used of most of pretrained networks.

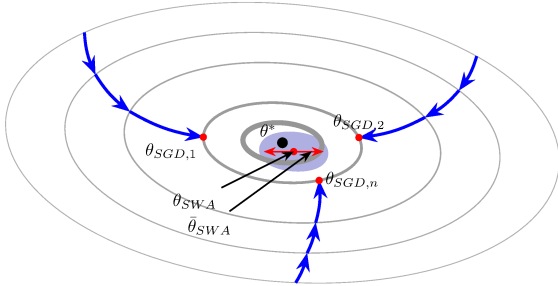
5.2.2 Deep ensemble methods

In [27], authors proposed a method based on a pure ensemble approach (see Figure 5.5b). From different weight’s initialization, they trained an ensemble of networks using the same architecture on the same data. Trained networks have different weights values θ_i . The idea is that in the prediction features space, regions with fewer training points should have greater uncertainty, which is reflected in



(a) Illustration of a bayesian deep network. In this setting gaussian parameters are learned for each individual neurons.

(b) Illustration of a bayesian deep network where only the last layer learn an output distribution.



(c) Illustration of the SWAG principle. After each epoch, SGD can get stuck at some boundary in the optimal parameter plan. By computing means and covariance of networks weights, one can get closer to the optimal parameter while having a notion of weights uncertainty.

Figure 5.4: Bayesian methods for uncertainty estimation of deep neural networks

greater variance in the predictions. Prediction and uncertainty estimation are then straightforward to compute using equation 5.4. Additionally, they also proposed to use proper scoring networks by mixing ensemble with the bayesian techniques seen in previous section 5.2.1, where the network learns its own uncertainty during training. Finally, by gathering T proper scoring networks within an ensemble, prediction and uncertainty can be computed as the mean and the variance of a mixture of Gaussian distributions [27]:

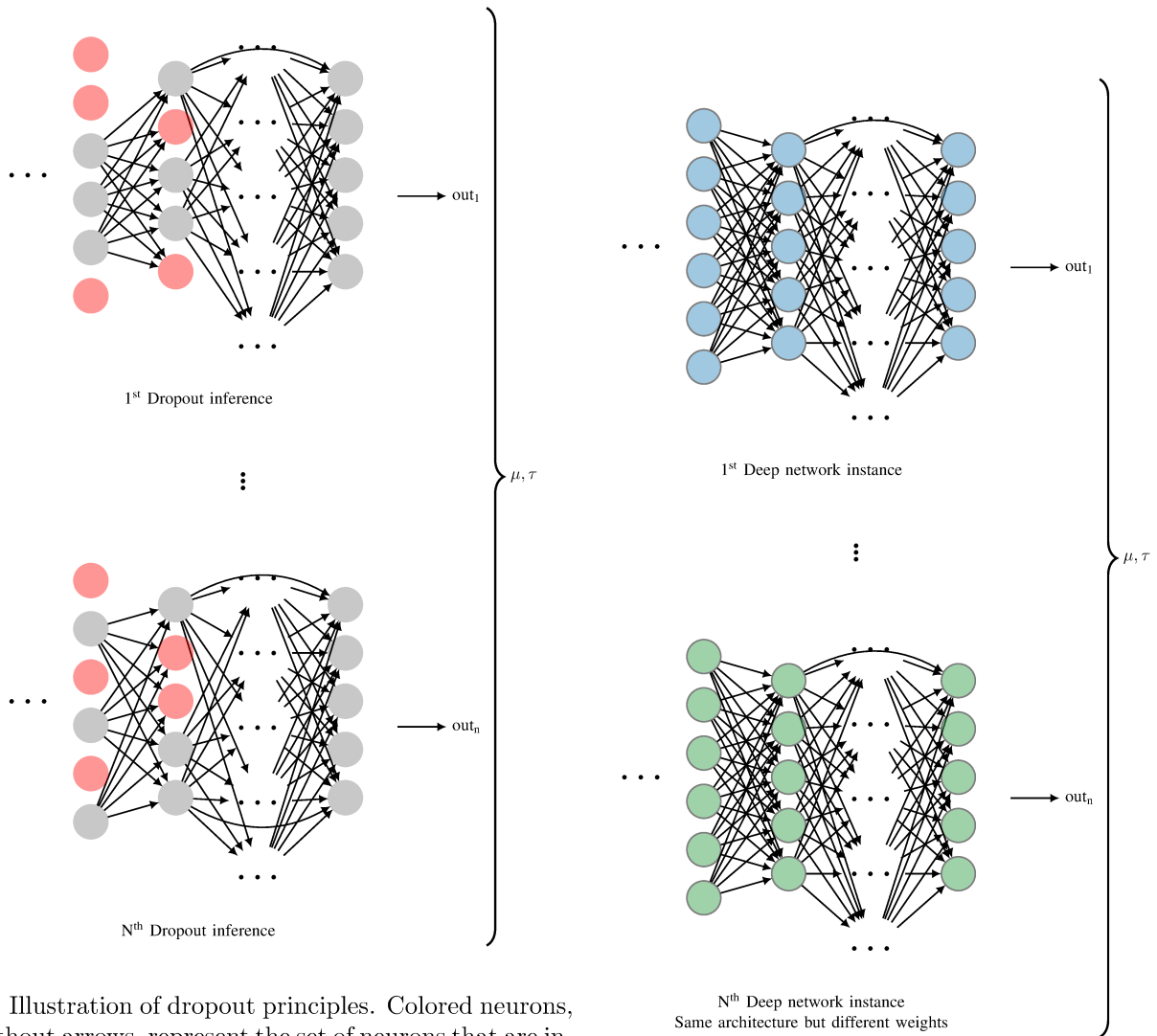
$$\tau = \sigma^2 = \frac{1}{N} \sum_{n=1}^N \left(\frac{1}{T} \sum_{t=1}^T (\sigma_{t,n}^2 + \mu_{t,n}^2) - \bar{\mu}_n \right) \quad \text{with} \quad \bar{\mu}_n = \frac{1}{T} \sum_{t=1}^T \mu_{t,n} . \quad (5.6)$$

5.2.3 Distance aware uncertainty model

The last type of model estimates uncertainty by computing a relevant distance between the features space and the input space or between samples in the feature space.

Learning a distance between the feature space and the input space: In this first approach, we want the measure to be low if the test data is close to training and high in the contrary. In [28], authors train a deep auto-encoder in visual navigation tasks, to reconstruct data seen during training. Denoting $x \rightarrow f(x)$ the input reconstruct from latent learned features, we want the lowest discrepancy between x and $f(x)$ on the training data i.e. ($f(x_{train}) \approx x_{train}$). The chosen discrepancy function $\tau(f(x) - x)$ can be simply the euclidean distance over images as features are the outputs. Test inputs close to training are likely to have a low reconstruction error while OOD data are badly reconstructed and have a high reconstruction error. Therefore, authors exploit this error as a proxy of the network uncertainty.

Learning a distance in the feature space: Another approach is to learn directly a latent features space where, if predicted features are close, then input are also likely to be close. Conversely, if predicted features are distant, then input features are distant. This motivates training distance aware uncertainty models, which can quantify the uncertainty of new data, by preserving a notion of distance between points in the input space and points in a latent features space. Figure 5.6 illustrates the general principle:



(a) Illustration of dropout principles. Colored neurons, without arrows, represent the set of neurons that are inactivated during inference with probability p_i for each layer i .

(b) Illustration of deep ensembles

Figure 5.5: Ensemble methods for uncertainty estimation of deep neural networks

Formally and ideally, for a predictive model f , such as a neural network, for every input x_1, x_2 , we would like to have:

$$k_1 \|x_1 - x_2\|_{features} \leq \|f(x_1) - f(x_2)\|_{input} \leq k_2 \|x_1 - x_2\|_{features} \text{ with } k_1, k_2 \in \mathbb{R} \text{ and} \quad (5.7)$$
$$\|\cdot\|_{input} \text{ and } \|\cdot\|_{features}, \text{ respectively a distance in input and features space}$$

In that case, features that are close in the input space are more guaranteed to be close in the output space. In other words, we would like to reduce the class of learnable function to bi-Lipschitz functions. While this setting is interesting in terms of uncertainty estimation and for safer learning, it reduces the expressive power and consequently might affect the accuracy of the network if the problem to solve is more complex than what the learned space can represent. Authors in [29] reviews some related techniques in the area of safe robotics and control. Authors notably show that integrating Lipschitz deep network with reinforcement learning and more traditional techniques on control theory help in building safe and more general controllers. One can indeed build a controller with a trade-off between safety guaranty, provided by stability analysis, and prediction accuracy provided by Lipschitz-constrained deep networks. Moreover these methods are especially suitable for out-of-distribution detection as shown in the simple two moons distribution dataset presented in 5.1. One of the of the current advanced method on distance awareness at current writing is SNGP (Spectral-normalized Neural Gaussian Process) in [30]. SNGP combines standard deep neural networks, which do not handle uncertainty by default, with traditional gaussian processes which are the standard when it comes to uncertainty estimation in classical machine learning. This last approach can adapt to several existing *residual* architecture by applying *spectral normalization* (SN) to the hidden residual layers and by replacing the dense output layer with a *Gaussian process* (GP) based layer. Spectral normalization detailed in the paper, allows a residual network to be distance preserving and therefore bound $\|f(x_1) - f(x_2)\|_{input}$ relatively to $\|x_1 - x_2\|$, as required. Features output can then be fed to a distance aware GP that model uncertainty.

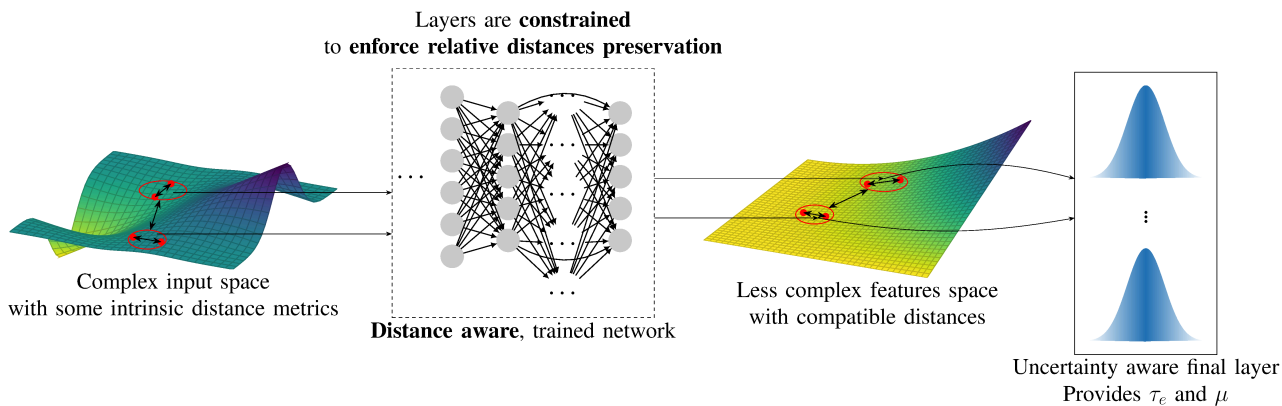
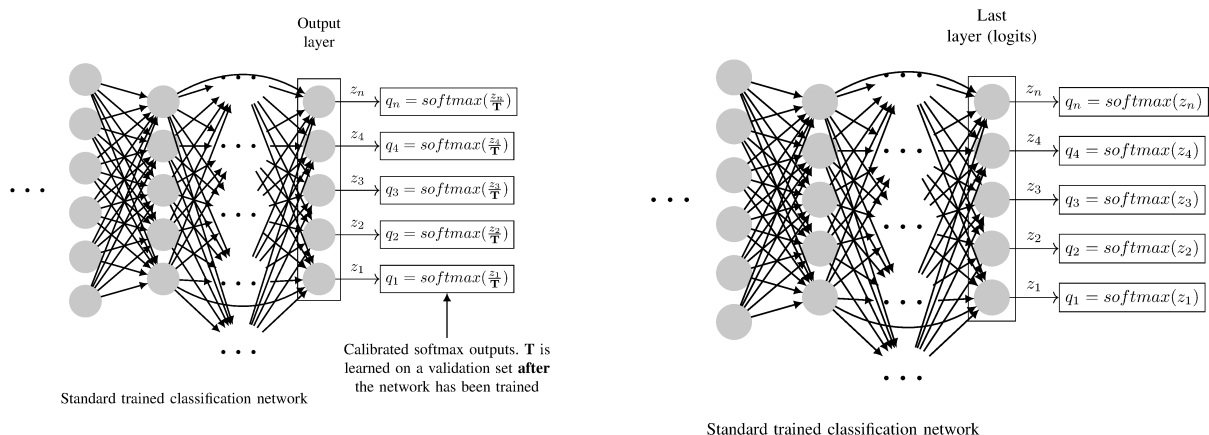


Figure 5.6: Illustration of the general principle of distance aware based model. SNGP is specific case. Plotting does not come from real data and serve just as an illustration.

5.2.4 External measure

External metrics can refer to methods that compute uncertainty based on *a posteriori* exploitation of the trained neural network architecture and without modifying the network structure. Various approaches were developed. Figure 5.7 shows the most simple method, *temperature scaling*, illustrated in Figure 5.7a in comparison to standard classification networks (Figure 5.7b). In [1], authors have shown that, while adding more layers to a deep networks improve accuracy, it also affects the network uncertainty quality. Typically in classification tasks, the softmax outputs will result in either overconfident or underconfident prediction for certain classes. A not intrusive method, that do not assume to change the neural network architecture, is to add a temperature scaling parameter in the softmax output. Once the model has been trained, a parameter T is trained on a validation set such as the confidence $q_i = \text{softmax}(z_i)$ become $q_i = \text{softmax}(z_i/T)$. Parameter T helps smooth output confidences leading an overconfident network to be less confident and reciprocally. Such external methods also are likely to better estimate epistemic uncertainty than other methods as they are less sensible to misspecification and bias of the model [31]. A disadvantage is the need to train two networks with different dataset which can be difficult in low data setting. The second network also add computing and memory overhead which can be limiting in complex robotics systems which leverage several deep learning modules.



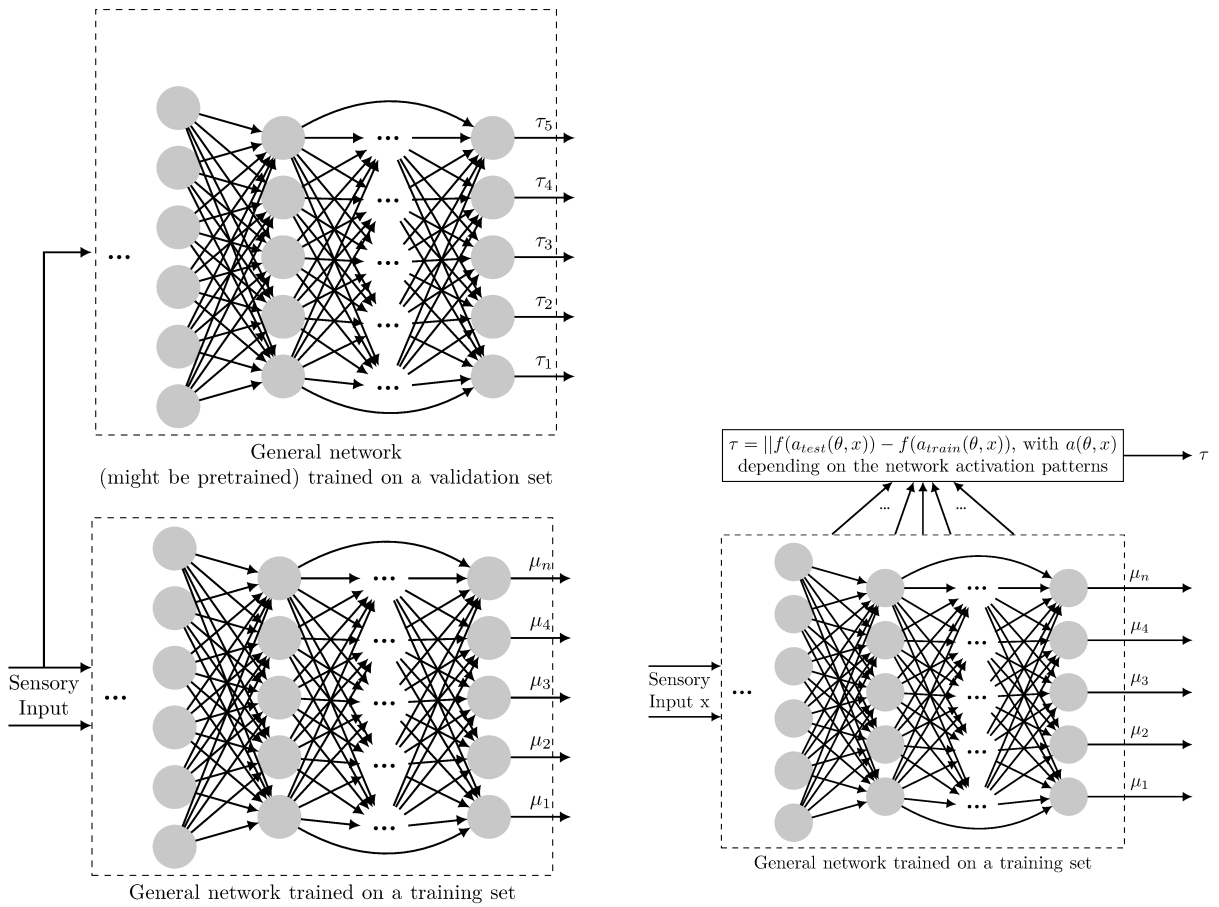
(a) Illustration of temperature scaling to correct confidence

(b) Illustration of standard classification outputs

Figure 5.7: Illustration of temperature scaling against standard classification

Competitive techniques approaches learn uncertainty based on an external model (Figure 5.8). In [32], author learn a trust score based on nearest-neighbours methods. Model can also be a deep neural network (Figure 5.8a). For instance, in [23], authors proposed to add an external deep neural network called *ConfidNet* trained to predict the True Class Probability of a prediction. This methodology seems promising since it is simple to implement and to use. Moreover, it has seen competitive results against other existing methods in classification problems. In aim of integrating uncertainty in our architecture, with a grasping scenario for validation, we have tested this approach and explored integration in the architecture with grasping modules. This study is presented as perspective in section 7.2.

Finally, a rather different approach rely on computing a measure quantifying how much vary the pattern of neurons activations between training and test data (Figure 5.8b). In [33] for instance, authors use topological data analysis, a data analysis field, that study the shape of data in high dimension in order to build topological descriptors characterizing the networks. They validated their method, on simple datasets, which is able to detect OOD data by analyzing changes in activation patterns. Overall, the interest of these methods is that they can be used on any existing networks without architectural change, or retraining of weight parameters. It can be especially interesting when using pretrained neural networks which has not been trained to predict their uncertainty with the methods presented the next sections.



(a) Illustration of an external measure approach based on an external learned model. (b) Illustration of an external measure approach where an external measure predicts uncertainty based on neurons activations.

Figure 5.8: External measures of uncertainty estimation of deep neural networks

5.2.5 Conclusion on uncertainty review in deep learning

We compare here some of the aforementioned metrics in the classification case, still with the two moon distribution dataset. Then, we show some examples on a regression task.

Case of classification: We compare different methods on the simple moon distribution dataset in Figure 5.9:

Standard ResNet is overall overconfident except on a thin layer around the separation curve. In this example, they Monte Carlo Dropout and Ensemble are output very close results but that is not true in general. They improve uncertainty on the separation line and are likely to provide more

robustness for OOD samples and slight covariate shift close to this separation. For dataset shift and OOD data that are farther to the separation line, they fail as the standard ResNet. Only SNGP based ResNet provide a good OOD uncertainty performance. This impressive results comes from the use of Gaussian Processes that are learned on top of the regularized (distance aware) neural network. The dataset is very simple and generalizing this analysis to higher dimension seems risky. However, it is worth noticed that author from [14] still outperforms other methods on OOD data, in traditional images classification benchmark such as CIFAR-100. Interestingly authors in [34] compared several approach on image classification and natural language processing tasks. Despite the simplicity of the approach, they suggested that ensemble models can be robust and outperform other methods when it comes to dataset shift such as dropout and variational methods. If temperature scaling calibration methods can be fine to quantify uncertainty for IID data, it significantly underperforms when it comes to dataset shift and OOD data.

An upside of classic dropout, ensemble methods is that we don't need to change the architecture of the network. Moreover one can use pretrained networks weights found in the literature for transfer learning. One downside of dropout and ensemble, however, is that they need several forward pass which can dramatically increase the inference time and memory and computing requirement. Thus these methods can't be easily used in setting where inference speed is a high requirement. Distance aware based methods and external metrics in this regards are interesting as they can use a single model in a single pass. They seems to be the best suited tool for OOD data detection in the toy benchmarks we carried out as stated in [14]. The method has yet to be validated on harder, real-world problem. As a downside, their specific regularization techniques imposes to learn new weights even for traditional architecture, therefore one could not benefit from pretrained available weights.

5.2. UNCERTAINTY ESTIMATION METHODS

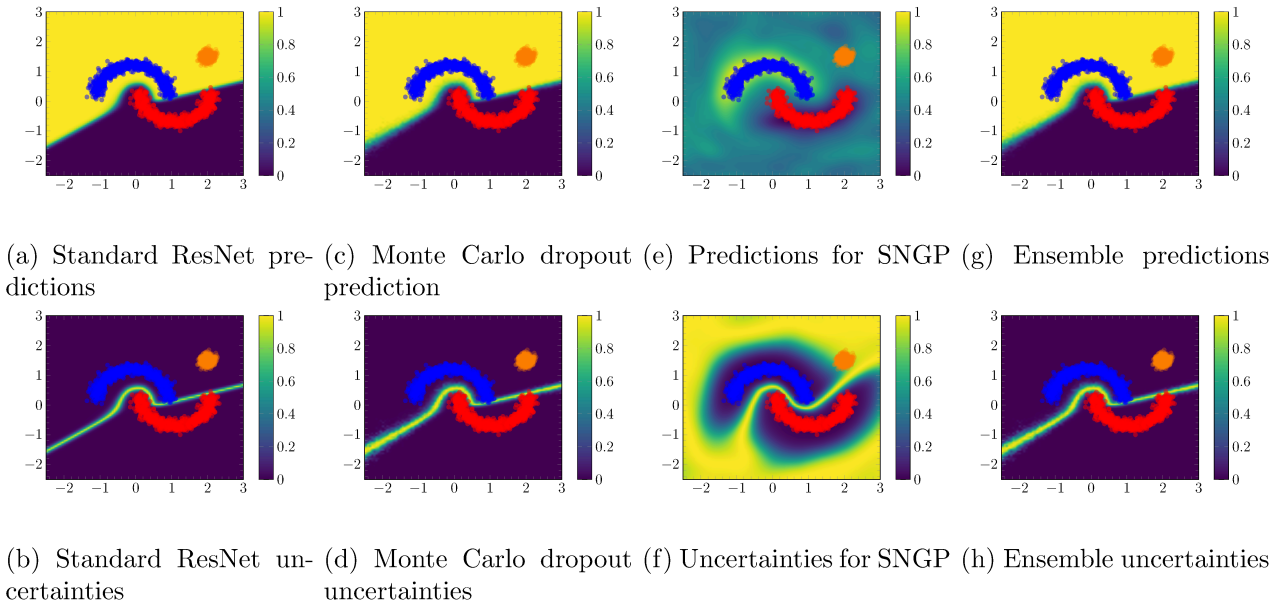


Figure 5.9: Classification probability with uncertainty for different uncertainty metrics τ .

Case of regression: Aforementioned methods can be extended to regression tasks. For instance, in a simple curve fitting example, we can obtain results where uncertainty represent the confidence interval around some mean value with respect to input features x . We compare here bayesian Monte Carlo dropout (Figure 5.10), learning of uncertainty (Figure 5.11) at last layer and a 5 network ensemble methods (Figure 5.12). For Figure 5.12, top graph shows individual predictions while bottom one presents total uncertainty related to the ensemble. Pros and cons of each methods are the same than in the classification case. We can see that close to data, total uncertainty is low and actually close to aleatoric uncertainty. In OOD region, uncertainty rises, but the ability to detect OOD samples is limited and not consistent accross regions. Inference at the last layer method have limited abilities cause uncertainty remains low in regions with lack of data (regions $[5,8]$, $[1.5,3]$) where predictions are not relevant. The same behavior is observed for ensemble method in region $[1.5,3]$. In this simple example, only Monte Carlo dropout has a relevant predictive uncertainty through all the domain for OOD data.

5.2. UNCERTAINTY ESTIMATION METHODS

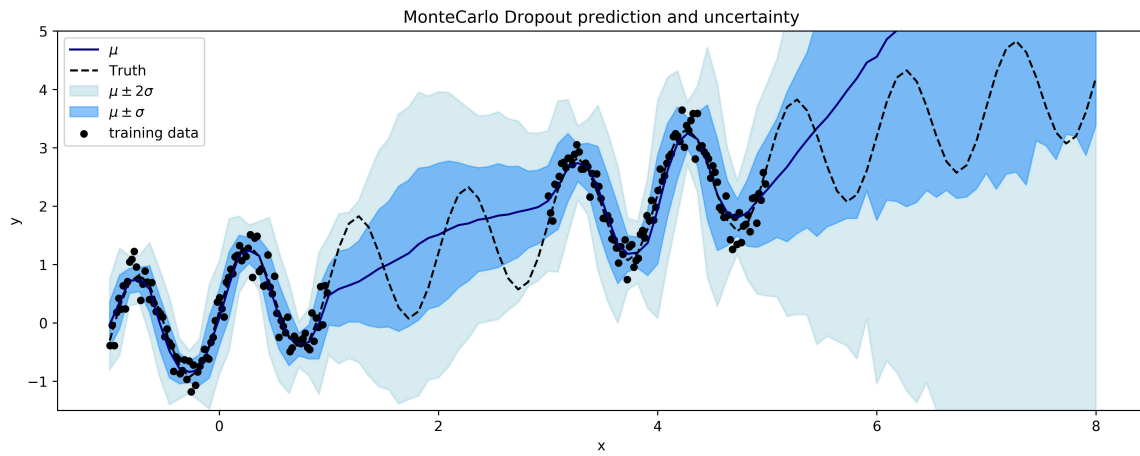


Figure 5.10: Regression task with Monte Carlo Dropout

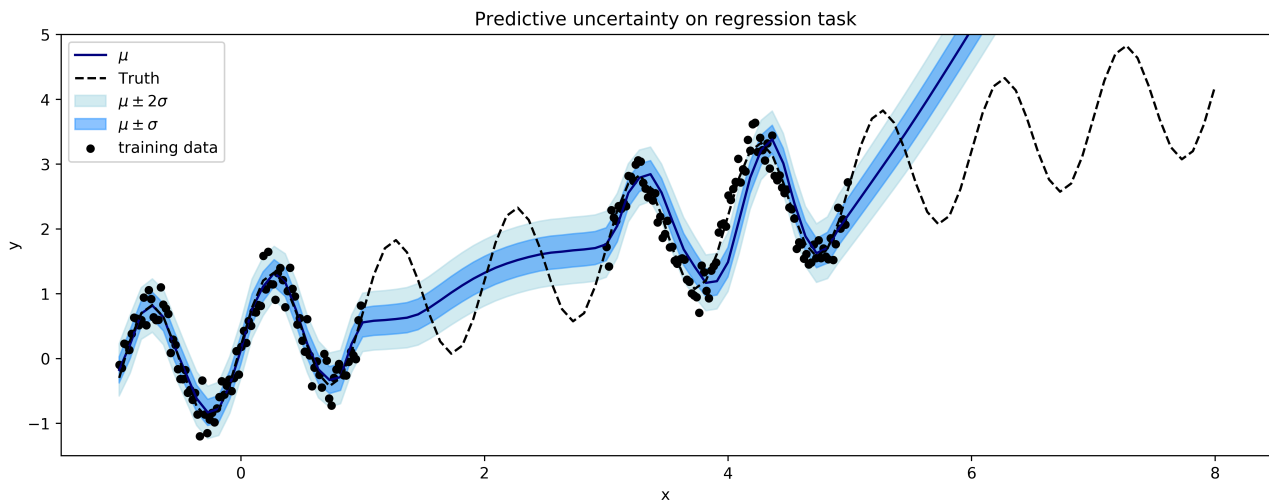


Figure 5.11: Predicted uncertainty inference at the last layer

5.3. UNCERTAINTY FOR DECISION-MAKING

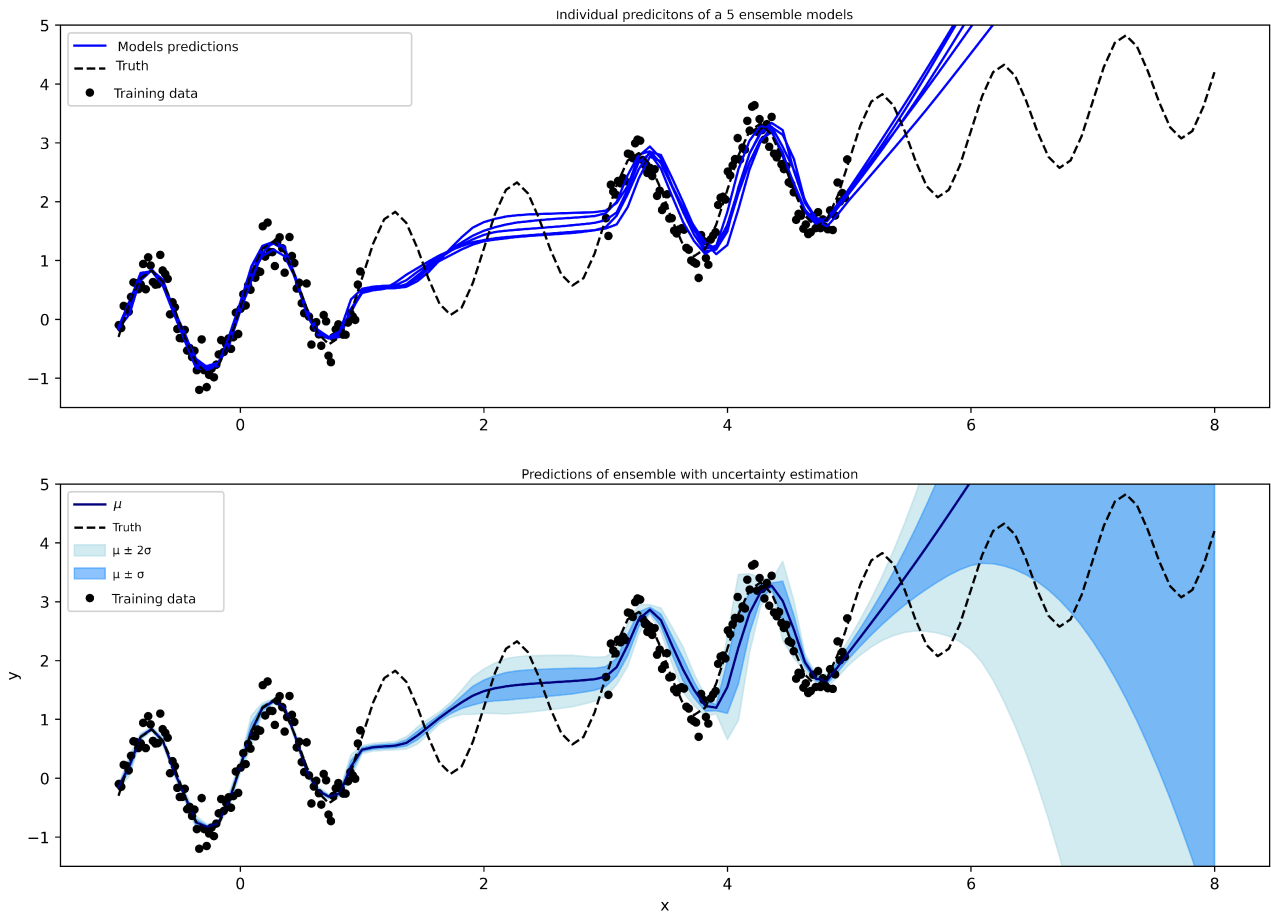


Figure 5.12: Estimating uncertainty using predictive uncertainty of the ensemble

5.3 Uncertainty for decision-making

Once the model has learned an uncertainty, the quality of this uncertainty estimation can be qualified by specific metrics on validation set and on tested data once in production. Determining the best metrics according to the model or the task is still an open research.

5.3.1 Calibration and sharpness

Once we have an uncertainty metrics, we have to characterize the quality of the uncertainty for two reasons. First, it allows to compare uncertainty estimation methods for a given task and second it provides a way to interpret and reason about the uncertainty level of the agent. Ideally, in order to create a safe decision system, we want some properties in terms of prediction accuracy on test

data and confidence. In general and intuitively, it is considered that a model has a good predictive uncertainty estimation when its accuracy is close to its confidence. In other words, calibration means that the true and predicted frequencies of an event should match. In that case the model is said to be *calibrated* [1]. Calibration Δ_{cal} can be simply expressed in both cases as:

$$\Delta_{cal} = |accuracy - confidence| \quad (5.8)$$

In classification tasks, to be calibrated means that predictive uncertainty should be close to 0 (respectively close to 1) when class prediction is good (respectively wrong). For regression, that means that confidence interval should be larger with the prediction error value.

It is important to notice that a good calibration, do not imply a good accuracy. For instance, if a model has to classify two balanced classes, a random classifier with 50% of confidence imply a perfect calibration. That's why notion of sharpness is also introduce in calibration literature. A model is said *sharp* if its accuracy is high.

We can find several metrics deriving from the base definition (equation 5.8). Several measures and calibration errors have been used in the literature to quantify *a posteriori* the uncertainty calibration. Integrating such information in the IRL decision process would allow to reason at higher level about learning modules.

Assessing the calibration uncertainty can be done by building a reliability histograms [1]. The idea is to partition predictions into M bins according to their output confidence. From this partitioning, the following metric can be computed:

- **ECE** (Expected Calibration Error) [35]: the accuracy is computed for each of the M bins. Then ECE is obtained by averaging the error across bins, weighted by the number of points n_b in each bin. (Figure 5.13 illustrates a reliability diagram).

$$ECE = \sum_{i=1}^M \frac{n_b}{N} \cdot \Delta_{cal}(b), \text{ with } \Delta_{cal}(b) = |accuracy(b) - confidence(b)| \quad (5.9)$$

- **MCE** [35]: While averaging uncertainty in ECE can be relevant to reduce noise, in some cases we might want to be much more cautious. In MCE instead of averaging, we consider the maximum bin uncertainty. This could be required in tasks with a risky consequences in case of failure.

$$MCE = \max_{b \in bins} \Delta_{cal}(b) \quad (5.10)$$

Overall, by binning predictive uncertainty, we can have more insight on the model behavior according to datasets, both in terms of accuracy and in terms of calibrated uncertainty which can then serve as input for higher level process. One downside, however, is the lack of relevance of bins with a low number of samples.

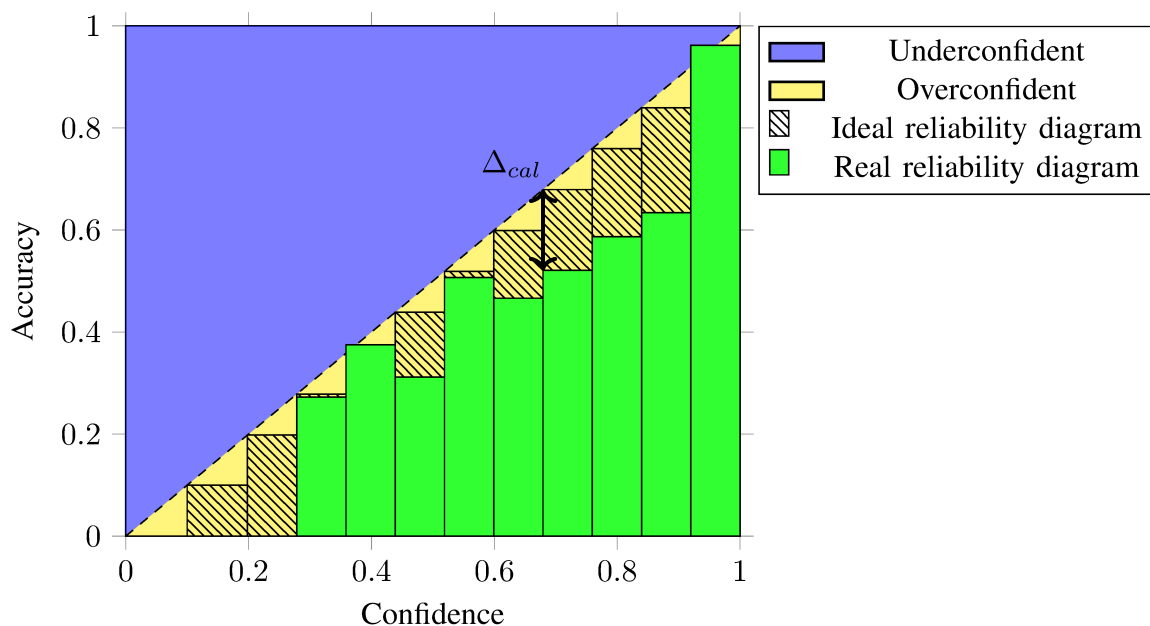


Figure 5.13: Confidence histogram illustrating ECE metrics applied to the predictive uncertainty of an uncalibrated classification network based on EfficientNetb0 [36], pretrained on ImageNet and fine-tuned on cifar100 with coarse label (20 different classes).

- **Proper scoring rules** : several metrics called proper scoring rules derive for instance from convex functions, information and entropy measure. Each metric has its upsides and downsides according to specific problems but they have not been further investigated during the thesis. For further detail, a comprehensive list is given in [37].

The aforementioned metrics help in measuring uncertainty calibration in a global way after training uncertainty aware module. In our ITL setting, as learning from new data changes network parameters, these metrics can be used as a monitoring tool, to check if uncertainty calibration remains stable across learning. This is important for performance and safety. Network predictive uncertainty can be

checked with a calibration dataset that is maintained and enriched across time and interaction. Given calibration metrics results, a re-calibration of the network might be necessary.

5.3.2 Performance metrics of a system

Once uncertainty is well calibrated, it is possible to use performance metric for selective predictions or reject options [38–40].

Following [41], we can adapt the classical contingency matrix [42] used in failure predictions methods, to exploit an uncertainty measure, as shown in Table 5.1. From this matrix, we can derive performance metrics, as well as an active learning process for our IRL which is discussed in next section .

		Prediction result		
		Wrong	Good	
Uncertainty	High (Ask)	True Positive (TP) (correct alarm)	False Positive (FP) (false alarm)	τ_{thresh}
	Low (Act)	False Negative (FN) (missing alarm)	True Negative (TN) (correct non-alarm)	

Table 5.1: Contingency table with uncertainty measure τ . Upper a threshold uncertainty, an alarm is triggered, for instance in an active learning framework.

Given a threshold uncertainty τ_{thresh} , predictions can be classified in two categories: confident (“Low” uncertainty) and uncertain (“High” uncertainty), for which the respective decision is to trigger (“Positive”) and not (“Negative”) an alarm. Relevance in triggering or not an alarm is denoted by “True”. Given a prediction, “True” corresponds to the main diagonal of the contingent matrix that is to say to have triggered (resp not triggered) an alarm if the prediction was indeed wrong (resp good).

This contingent matrix can be seen as a binary classification to evaluate failed predictive selections, for instance concerning predictions of a neural network. A good uncertainty metric should maximize the proportions in the main diagonal (TP and TN), and eradicate missing alarms (FN) corresponding to failed predictions which are viewed as certain, while at the same time limiting the number of false alarms (FP). Indeed missing alarms (FN) could lead to safety issues in industrial collaborative robotics

applications, and false alarms (FP) represent an additional cost in an industrial context. An alarm will trigger a failure in the system which has to be solved by human interactions (demonstrations and utterances) in the IRL process. Therefore, we want to limit the number of unnecessary interactions (FP).

Various statistical tools and Key Point Indicators (KPI) relying on computing several ratios can be leveraged to assess the performances. One of the main classical tool for predictive selection given this binary classification problem is **The Receiver Operating Characteristic (ROC)** illustrated on Figure 5.14. This curve plots, by varying the threshold, the *False Positive Rate or specificity* ($FPR = \frac{FP}{FP+TN}$) representing proportion of false alarms among good predictions, against the *True Positive Rate or sensitivity* $TPR = \frac{TP}{TP+FN}$ representing the proportion of relevant alarms among failed predictions. Alternative curves exists such as the **Risk-Coverage** curve. This curve plots, by varying the threshold, *the risk* which is usually the accuracy of the model against *the coverage*, which is the ratio of the number of confident predictions over total number of predictions $\frac{FN+TN}{TP+FN+TN+FP}$.

From the ROC curves we can extract three different scores also represented in Figure 5.14:

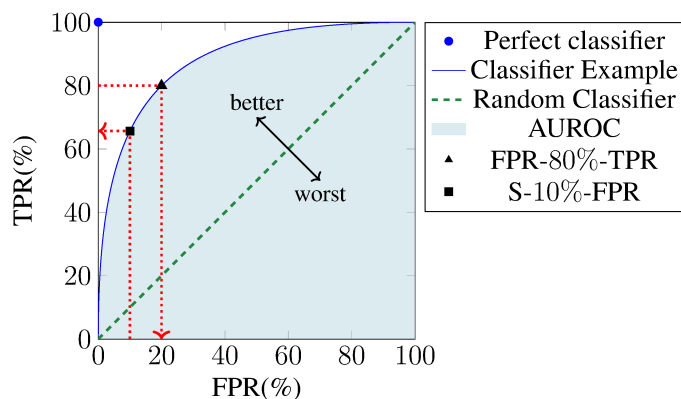


Figure 5.14: ROC curve.

- The **AUROC** (Area under the ROC Curve) score [42] which represents the ratio Area Under the ROC curve. A random detector has a score of 50% while a perfect detector has a score of 100%. Use of AUROC, however, can sometimes be misleading when comparing different models. In [41] authors showed that between two slightly different trained DenseNet models, in certain cases, one with a greater AUROC metrics can have less total accuracy. They proposed the **AURC** metric (Area under the RC curve) to solve this limitation.

5.3. UNCERTAINTY FOR DECISION-MAKING

- **FPR- α %-TPR** : False Positive Rate for a given percentage α of TPR. It represents the rate of false alarms for a given security level.
- **S- β %-FPR** : Compute the success rate (TPR) at β % of FPR. This score is representative of an economic cost due to the number of false alarms it can yield.

The contingency table can be leveraged to set the threshold τ_{thresh} on the amount of acceptable true positives with respect to false positives, represented in Table 5.1. Above this threshold, the agent is confident enough and decide to Act. Below this threshold, it must decide to Ask. This is illustrated in Figure 5.15. Given an histogram of predictions, we represent in green the histogram of good predictions and in red histogram of wrong predictions for a given uncertainty τ . A good predictive model associated to a well-calibrated uncertainty model should be able to separate clearly the two distributions. By setting τ_{thresh} , we can put more or less emphasis on how safely the IRL must act.

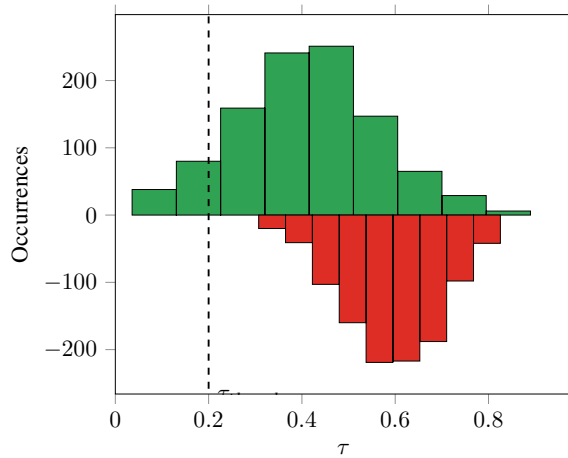


Figure 5.15: Illustration of τ_{thresh} usage to limit wrong actions

The same idea is represented in Figure 5.16 directly on the ROC curve. It shows that the IRL agent can choose a compromise between safety level (TPR) and false-alarm cost (FPR), by setting this τ_{thresh} .

Overall, in function of τ_{thresh} , we can also derive and exploit the two following KPI to assess the performance of the ITL decision process:

- The accuracy $ACC = \frac{TN+TP}{FN+FP+TN+TP}$ to evaluate the relevance of decisions between acting

and asking (main diagonal density). When this ratio tends to 1, this means a good expertise of the IRL. In that case, distributions of good and wrong predictions are well separated among uncertainty τ .

- The Negative Predictive Value $NPV = \frac{TN}{TN+FN}$ which represents the number of TN (correct non-alarm) against the total number of negative alarms. This score is a key indicator of the safety level in acting decisions, and we want this ratio to be close as 1 as possible for our ITL.

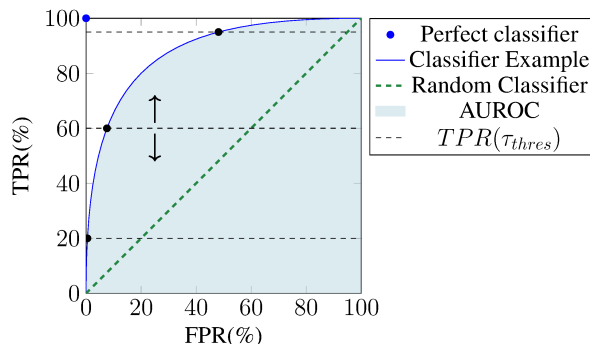


Figure 5.16: Based on the ROC curve, one can choose a threshold to balance the trade-off between the amount of true and false positive.

5.4 Active learning setting

5.4.1 Uncertainty for active learning setting

Now that we got a better idea of what is uncertainty and how it can be estimate, we can focus on how integrating it in real-world robotics application. Notion of uncertainty have been used in several works in robotics for self active learning purposes. There is no clear definition, but it is usually related to novelty detection [28], curiosity [43, 44] or motivational intrinsic reward [45], play [46, 47]. Overall, these methods exploit unknown to drive specific behaviors. Active learning field focuses on strategies which can reduce the unknown and improve the underlying *learned* model. In this active learning review, [48], authors distinguish a query system and an oracle as the two main building components of an active learning systems. Then they discuss two methods to retrieve more information: human feedback and semi-supervised/transfer learning. As suggested in [49] they are strong connections between active learning and robotic interactive learning.

- The query system is the component that is responsible to ask an oracle for more information when needed. In an IRL system, this component is build around communication modules to inform an oracle about the lack of knowledge when a failure occurs.
- The oracle is the component that answers to the query system. It can be a fully supervised human feedback where a human give all label each time the system is not certain. A more flexible approach is the semi-supervised setting, where the agent uses a proximity metrics to label unknown data based on already labeled ones.

As real-world data are scarce and domain specific, there is a growing interest in integrating active learning with deep learning in robotics. It has been experimentally proven, indeed, that modeling uncertainty can be crucial for better exploration vs exploitation tradeoff. For instance in [50], authors took inspiration from an early active learning querying techniques called query-by-committee [51]: they exploit an ensemble of deep neural network to learn an intrinsic reward function, that is then leveraged in an end to end deep RL application. They validated this framework in navigation simulation and on learning pushing and grasping objects on a real manipulator. However, they do not try to represent and explicit inner learned models, making it a black box. In [52], authors proposed an uncertainty aware deep RL framework that aims at disentangling both aleatoric and epistemic uncertainty. They exploit auxiliary networks that learn both the epistemic and the aleatoric uncertainty. They validated their framework by adapting a distributional RL variant of the DQN algorithm on simulation and Atari games tasks. Though interesting to learn specific skills in predefined tasks, these methods still lack, however, general high-level representations that are needed for better interpretability and better cross domain generalization in interaction. Adapting such techniques to the robotics context in an IRL architecture is a promising area of research for interactive task learning.

5.4.2 General active learning process for ITL

More precisely we can exploit uncertainty notions presented in section 5.3 in order to reason about uncertainty in the active learning context and for integration within the IRL agent architecture. A general methodology is illustrated in figure 5.17 and can be summarized with the following procedure:

1. A neural network is trained on a dataset with a loss L . According to estimation techniques, such as the one presented in 5.2, uncertainty can be learned during this training. For external

measure, uncertainty is learned on a validation dataset.

2. Calibration can be checked with metrics mentioned in 5.3.1 and assess the quality of uncertainty estimation at time being. If calibration is not good, that means that the uncertainty estimation techniques have not learned a representative uncertainty suited for the task, given the provided datasets. That means that we should train with more data or change inner predictive and uncertainty models if more data is not sufficient.
3. Then, the model prediction and uncertainty are exploited within the IRL agent process loop.
4. After a prediction is done, uncertainty is propagated in upper level and predictive selection is carried out following 5.3.2 to decide between act or ask.
 - (a) If it is uncertain ($f(\tau) > 0$ given τ_{thresh}), then a request for label is asked to the human according to the current task knowledge. Labeled data are collected and can be augmented to update learning and calibration datasets. Then back to 1. Ideally, the TPR, ACC and NPV should get closer to 1 after this active learning step.
 - (b) If it is certain ($f(\tau) < 0$ given τ_{thresh}), then the IRL agents acts. If uncertainty estimation is representative and threshold τ_{thresh} is adapted, then missing alarms should be rare but can still occur. In case of a missing alarm, humans should be able to stop the action. As a last resort, unmet post-conditions leading to a failure in the architecture could be leveraged by the IRL agent. Then, the IRL agent should also ask help to correct its behavior. Then back to 1. FN is likely to decrease, reducing the risk of unexpected actions, while TPR and ACC should increase.

This is a first approach to active learning that is quite general and might be used for different learning (such as learning perception) modules in the architecture. To be fully adaptable, we need more research on how to manage efficiently datasets and τ_{thresh} updates as the agent learns new data. Moreover, tests should be done to validate practical use and evolution of the different rates. This is discussed as perspectives in chapter 7

5.4. ACTIVE LEARNING SETTING

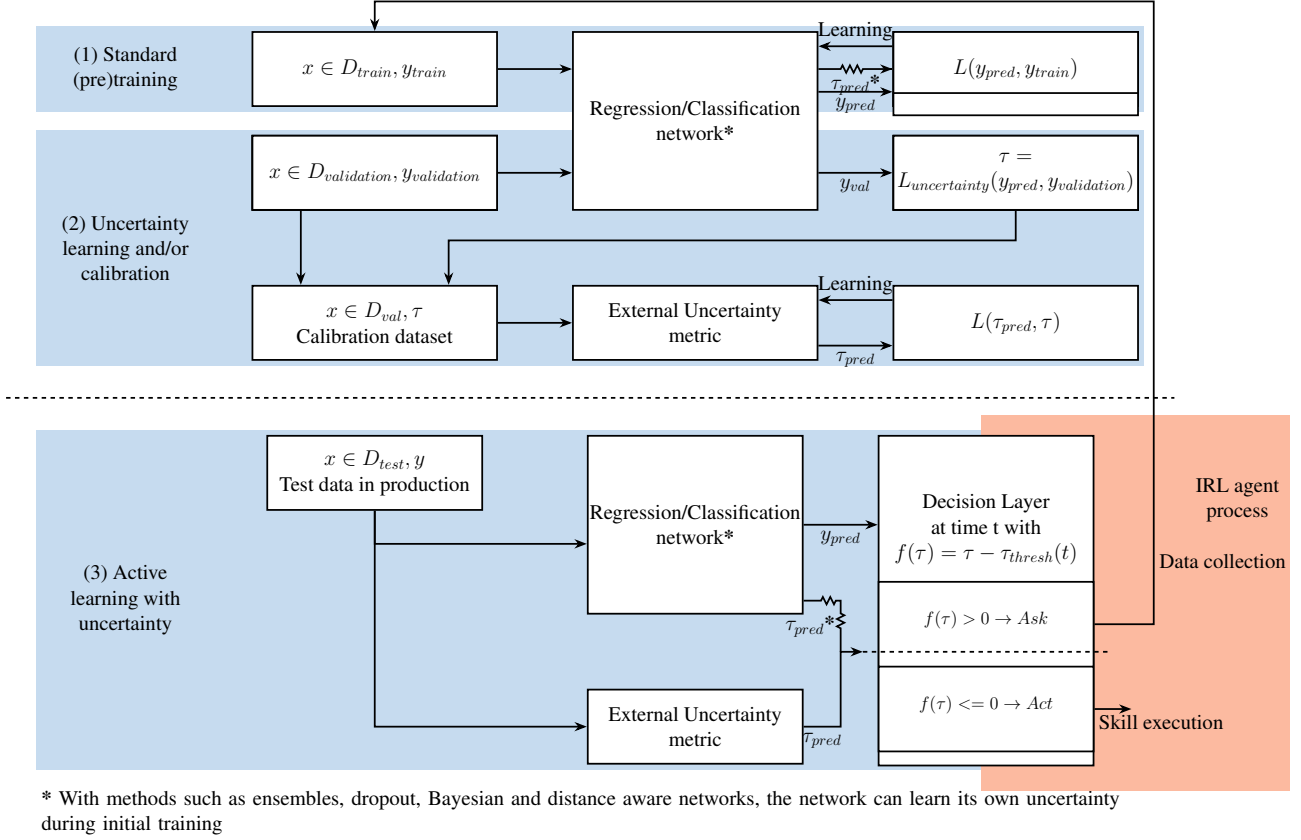


Figure 5.17: General pipeline for active learning

5.4.3 Uncertainty aware behavior model

In order to integrate aforementioned works on the architecture, at the action/decision-making level, we must extend the behavior model with uncertainty handling and propagation capabilities. A natural extension of traditional BTs with uncertainty is to consider that tests do not work only in a fully deterministic way, that is to say are not reduced to binary outputs true or false. Uncertainty propagation can be carried out through Belief Behavior Trees (BBTs) [53]. In that setting, the agent can reason over a belief state within its working memory b_{wm} and propagate uncertainty through nodes of the tree. This belief b_{wm} is a distribution of states s_i over a set S , associated to a predictive uncertainty τ_i : $b_{wm} = \cup_{s_i \in S} (\tau_i, s_i)$. The actions of the IRL agent are also allowed to have non deterministic outcomes.

For illustration in the most general case, we can consider a simple planar grasping use case. An IRL agent has a hypothetical uncertainty aware model, able to predict 3 grasping rectangles candidates

along with their predictive uncertainty. The IRL agent is asked to pick-up an object on the working space. Figure 5.18 represents the kind of decision process that could occur in such a scenario with BBTs for a specific behavior. In practice, following [53], BBTs extend nodes representations for conditions and actions.

Conditions node: Conditions node $b_{wm} \rightarrow c(b_{wm})$ are functions of the belief states (the three predicted rectangles). According to some tests $s_i, \tau_i^c \rightarrow f(s_i, \tau_i^c)$, condition node also return a set of belief states along with their termination status r_i (*success* or *failure*) and the level of associated uncertainty τ_i^c . More specifically, we can express condition nodes as :

$$c(b_{wm}) = \cup_{s_i} c(s_i) = \cup_{s_i} (\tau_i^c, s_i, r_i | f(s_i, \tau_i^c))$$

According to the return status (two grasping rectangle are certain, one is not certain), the condition node can decide to propagate the belief state and uncertainty in parents and following nodes, such as action nodes.

Actions nodes: Action nodes $a(b_{wm})$ in the general case can have probabilistic outcomes. For instance, even for a good grasp prediction, the object might slip after a grasping attempt. In general, an action a is executed if preconditions are validated and certain for some believed states s_i . Action a is then applied to those valid state s_i and can be associated to non deterministic postconditions. Indeed, for each specific starting state s_i , an action can lead to different outcomes $a(s_i) = s'_{i,j}$ with an uncertainty τ_j^a . The uncertainty τ_j^a can be a probability over possible states such as $\sum_j \tau_j^a = 1$. Assuming individual postconditions are independant events, a postcondition can then be expressed as:

$$c_a^{post}(s_i) = a(s_i) = \cup_{s'_j} (\tau_j^a, s'_j)$$

By considering an independence between action and the perceptual state, we can assume the overall uncertainty of being in state s'_j is $\tau_{i,j}^a = \tau_i^c \tau_j^a$. Therefore, the set of possible outcomes when applying the action to the belief state is:

$$a(b_s) = \cup_{s_i, s'_j} (\tau_i^c \tau_j^a, s'_j), k \in \mathbb{N}$$

5.4. ACTIVE LEARNING SETTING

Such a view can then be leveraged for acting and local planning by propagating uncertainty in the tree.

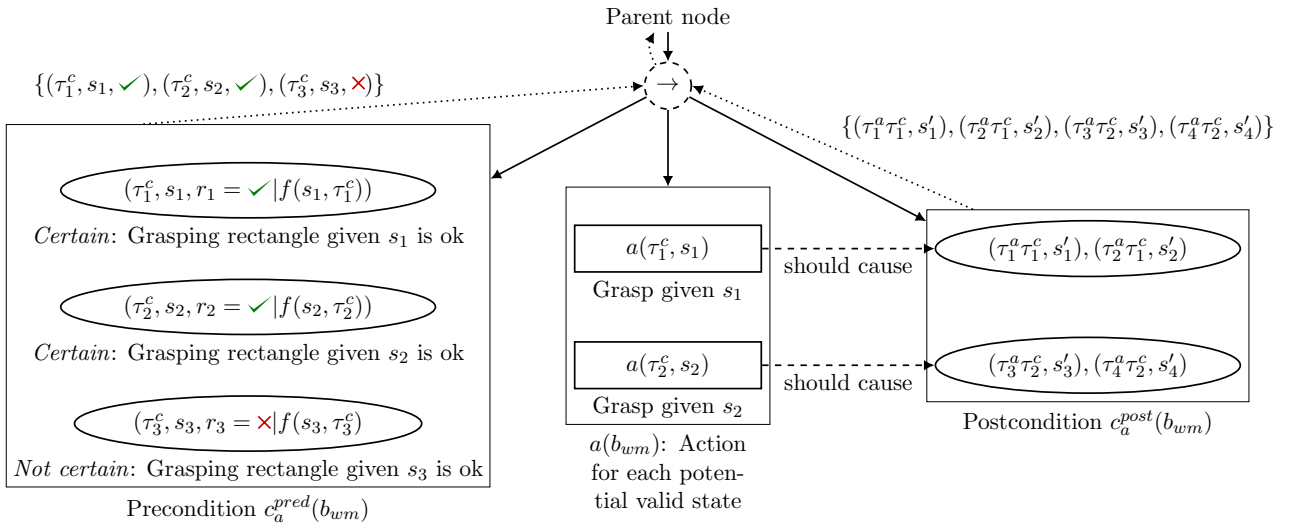


Figure 5.18: General extension of belief behavior trees for condition and action nodes for uncertainty propagation. Dotted arrow represent what is returned by the node after its termination.

In that setting, we can use uncertainty estimation and predictive selection function $f_{\tau_{thresh}}^{ROC}$ (section 5.3.2) as the test function for the condition nodes.

We can then integrate the active learning process within this framework and the interactive pipeline (see Figure 5.19). Given several conditions and uncertainty aware perceptual modules, the IRL agent selects predictions based on predefined threshold τ_{thresh} . Predictions and uncertainties are propagated back to upper nodes, a failure occurs and can be solved by asking demonstrations in an active learning setting, if no grasping candidate is valid.

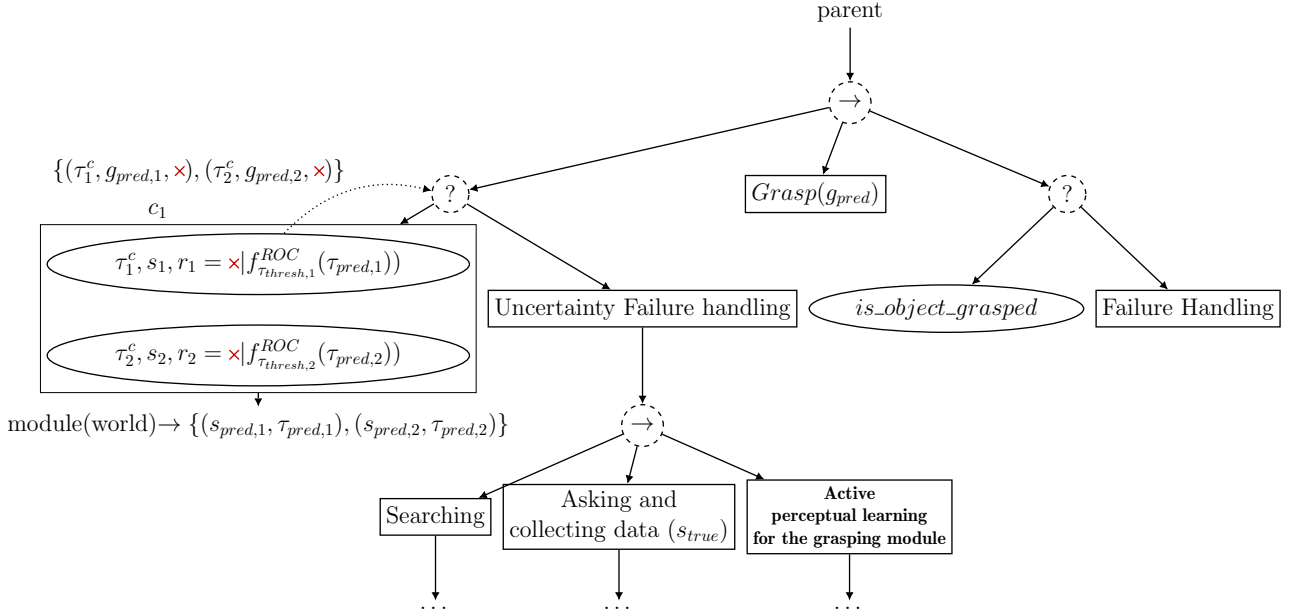


Figure 5.19: The IRL agent can exploit specific BBTs structure to handle uncertainty provided by lower level modules and the belief state in working memory.

5.5 Conclusion

In this chapter, we defined how uncertainty can be viewed as a composition of different types of uncertainties, aleatoric and epistemic uncertainty which are related respectively to stochastic nature of data and to the nature of the model. Learning such an uncertainty is determinant, if we want an IRL agent being able to reason and to cope with several types of biases related to dataset shift. The most studied ones, being out-of-distribution samples and covariate shifts. We explained the main techniques used to adapt deep learning modules to learn uncertainty and we present a few metrics to qualify the calibration of model uncertainty: how well predicted uncertainty matches with the observed accuracy on test data. Furthermore, we showed that a tradeoff uncertainty can be at the core of an active learning process. Indeed, by applying a threshold to the amount of acceptable uncertainty, we can derive a notion of curiosity or motivation that can be leveraged by the agent during interaction for active learning by deciding to ask rather than act. Thus, an IRL agent can have much more insight on its predictions abilities, leading to safer behaviors and the ability to ask for help in richer scenarios.

Chapter 6

Implementation and validation on a planar pick and place learning task

Contents

6.1	Choices of sensors for IRL perception and interaction	108
6.1.1	Non-verbal communication	109
6.1.2	Verbal communication	113
6.2	Perceptual and acting modules integration	117
6.2.1	Speech recognition and understanding	117
6.2.2	Prior object segmentation and tracking	118
6.2.3	Joints pose estimation for non verbal interaction	118
6.2.4	Location affordance learning	118
6.2.5	Acting	120
6.3	Validation scenario	120
6.4	Conclusion	124

In this chapter, given design choices made in chapter 3, we implemented an architecture prototype for interactive task learning (ITL) with preferences, in a human/robot collaborative industrial context. We first explain our sensors choice in terms of human/robot interaction modalities in section 6.1. Then, we explain which modules were integrated in the architecture to validate our main specifications 6.2. Finally section 6.3, validate the integration experimentally on a real UR10e industrial collaborative robot. The cobot is taught, online and incrementally, a simple task with variations based on human preferences. During the interaction, it leverages some prior knowledge to learn, online and incrementally, both low level and high-level task information in the most natural way as possible for humans.

We recall in Figure 6.1 the organization of the architecture and we now provide a justification for the choices of sensors and modules we integrated, in terms of interaction modalities, for the implementation and validation of the architecture.

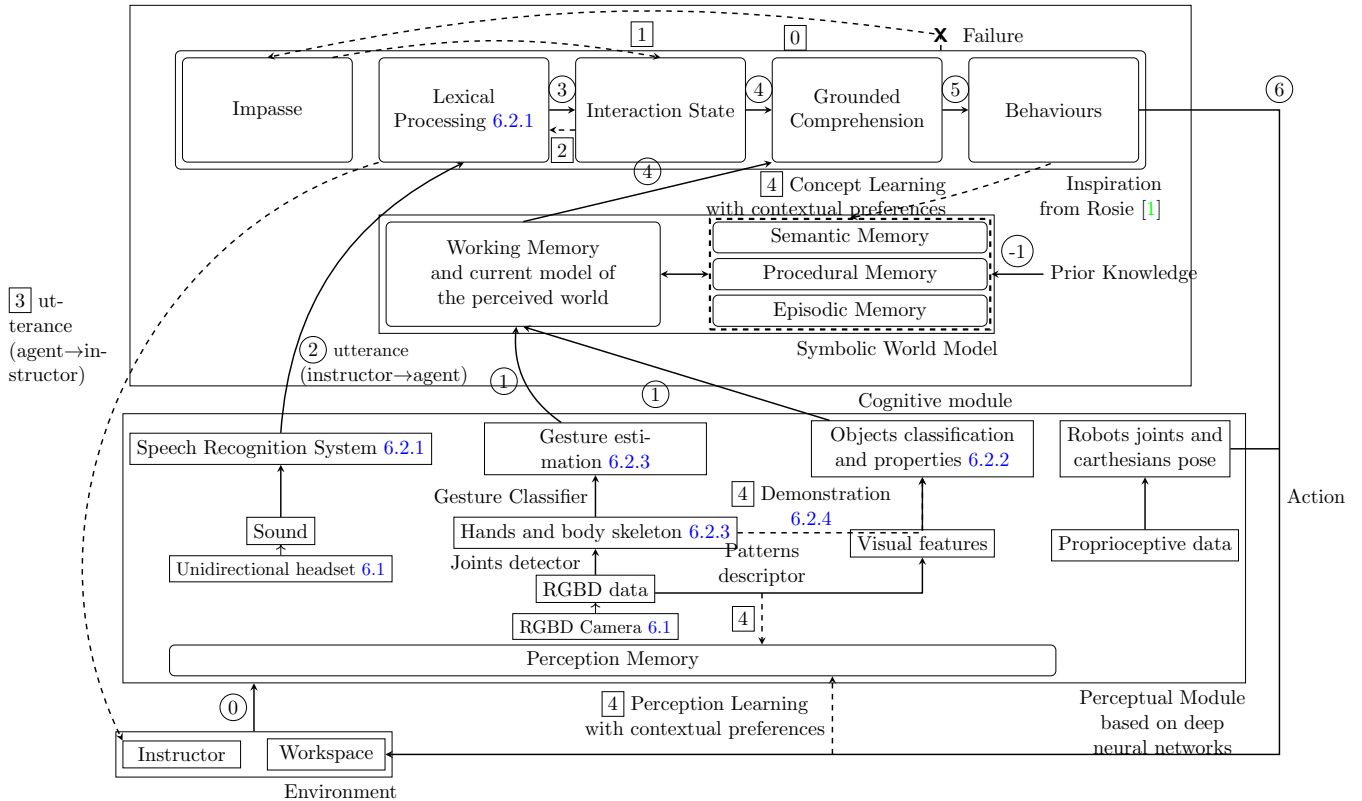


Figure 6.1: Details view of the architecture in terms of integrated modules.

6.1 Choices of sensors for IRL perception and interaction

A robot can sense the environment using exteroceptive sensors. Specifically, we want the IRL agent to adapt to the human and therefore, sensors must be the least cumbersome for workers and allow to sense and perceive semantically meaningful communications means. This led to a non exhaustive listing of sensors types, that could be used for different interaction modalities to validate the architecture. We can distinguish non-verbal and verbal communications means. Both are important when defining a task, as some information can be more easily shared by words or by non verbal interaction.

6.1.1 Non-verbal communication

Vision in general: Vision sensors are interesting because they allow to capture rich information on the environment (images and stream of images) and with a wide spatio-temporal perceptive field. Technological solutions are very diversified. They are non-contact devices, non-intrusive and very flexible sensors.

Thanks to their non-invasive upsides and the important quantity of information that can be extracted from an image or video, vision is one of the key modalities in most of human/robot interactions. For example, in the same video, we can both detect the pose of the operator and more generally do scene analysis. Classical cameras are passive sensors that convert the light emitted and reflected by the environment into a grid of color pixels. There are different spaces of representation of the colors. The RGB space (red, green, blue) is the most traditional but to limit the effects of change in brightness, we can prefer other color spaces such as the HSV space (hue, saturation, value).

We can distinguish these sensors according to their sensitivity to the wavelength of light. Thus, most cameras allow to have information in the visible light range. The use of sensors using other areas of the spectrum of light may be relevant to improve the knowledge of the environment, but in that case the IRL agent could perceive more than the human, and therefore should provide insights on its perception. For instance in the case of thermal cameras, for example, the body temperature of a human can be used as a means of presence detection while avoiding the problems of reverberation and illumination of conventional cameras [2].

Compared to 2D sensors, 3D sensors naturally provide more information about the scene. For example, it is easier to segment a scene with an area of interest by simply filtering on the depth. We can distinguish between passive 3D sensors such as stereoscopic cameras that rely on depth inference via parallax, and active 3D sensors that rely on the emission of light rays and the processing of reflected rays, such as structured light cameras. Cameras that can output both RGB data and depth are referred to as RGBD cameras.

Finally, we can mention the TOF (time of flight) cameras and the LIDAR. These sensors send a light beam, generally in the infrared range, and measure the return time of flight of the beam. By knowing the speed of light one can compute the distance between the laser emitter and surrounding objects. They allow in general a scan of 360° at the opposite of classical cameras and are thus very

used in the field of mobile robotics.

Communication requires a common representation of the environment. Therefore, the IRL agent must be able to create a semantic representation of its environment that can be understood by an operator. Scene detection and segmentation techniques based on vision are then relevant for a high-level communication between humans and robot.

Biological signals: Biological signals are also a way for human-robot collaboration. They usually involve the use of sensors to be worn by the user. Having sensors directly connected to the operator's body has advantages. They can give information on elements that are difficult to access with external sensors, either because they are internal signals of the human body, or because of occlusions in the case of vision sensors. On the other hand, wearing a sensor can be annoying for some people. Many sensors exploit various biological signals:

- Myoelectric sensors (EMG sensors), such as the Myo sensors measure the electrical activity of muscles. This type of sensor can be used for gesture recognition [3].
- We are beginning to see the development of electrical impedance tomography (EIT) based sensors for body imaging. In [4], authors showed that it is possible to use these impedance images for gesture recognition.
- EEG (electroencephalogram) headsets measure the electrical activity of the brain with contact electrodes. A distinction is made between wet and dry electrode systems. Dry electrode systems are less accurate than wet electrode systems because of the less controlled skin/electrode contact. However, the installation of dry electrodes is easier and less restrictive. When an EEG headset is used in a system interacting with a computer or robot, it is called a BCI (brain computer interface). The location and number of electrodes influence the quality and type of brain signals that can be measured by the EEG headset, such as some emotions or ideas. In [5] authors implemented the possibility of controlling a robotic arm using brain signals measured with an EEG headset. However, this requires significant learning on the part of the human to perform this task.
- Some types of brain signals can limit the need for people to learn. For example, the use of SSVEP (Steady state evoked potential) is quite common in BCI applications. These are specific

signals that appear in the brain at the same frequency as periodic visual stimuli (flashing leds). In the context of human-machine interfaces in robotics [6], authors use flashing icons on a screen to generate SSVEP signals in a BCI helmet wearer. The latter can then mentally select the icons on the screen to send commands to a robot. To gain mobility, some authors evoke the use of augmented reality glasses as a display medium instead of a screen. In [7] authors show that it is possible to direct a mobile robot using a BCI interface and augmented reality glasses (hololens glasses). The SSVEP has the advantage of requiring little or no learning but requires a focus of the human to limit noise. Overall, BCI interfaces return noisy signals which still makes it difficult to leverage for robust real-world applications.

Pose estimation: Pose detection is of great interest in human-robot interaction because it is a representation that can then be used in scene analysis and gesture detection. Also locating the pose of an operator, is often used for human demonstrations to robots or for emotions recognition by tracking facial key points [8].

In the field of co-manipulation, pose detection is also used, so that the robot helps the operator to reach more ergonomic positions [9]. From the sensor point of view, there are wearable sensors like the Xsens suit [10], and more generally devices leveraging Inertial Measurement Units (IMU) and visions. For instance, authors in [11] leverage IMU, magnetic sensors and lasers to track the pose of humans and adapt off-line generated motion paths.

Actually from the processing of the video stream of a simple RGB camera, it is possible to estimate the pose in a less invasive way. On the other hand, the sensitivity to occlusions is higher. Among pose estimation methods, we can distinguish between methods based on joint detection [12] and dense methods. In the first case a neural network produces a probability map of presence for each joint of the body, and we obtain the skeleton of the individual. In the case of dense detection, the network assigns to each pixel the limb to which it belongs. As instance of joint detection method, in [13], authors estimate a 3D body pose from the 2D pose provided by Openpose [12]. They then exploit this pose to make a robot learns trajectories such as opening a small chest with a handle.

Movement: Some signals can require the IRL agent to have especially good temporal resolution representations to fully understand its surrounding and human interactions. Indeed, movement can

be key in several cases such as high speed gestures interpretation. Depending on the dynamic of the observed phenomenon and cost, several sensors can be considered for the accurate detection of environmental and human movements.

Wearable sensors such as Xsens [10] and IMU based devices, can have a good temporal resolution for human gesture recognition, but vision sensors are less cumbersome. For low dynamic movements, such as low movement human gestures, standard cheap RGB-D camera can be leveraged by computing optical flow between successive frames. For higher dynamic movements, however, motion blur can become too important and make optical flow not exploitable. In that setting, much costlier or bulky RGB-D sensors, such as high-speed cameras can be used.

Recent specialized vision sensors, neuromorphic cameras, also called event cameras, have specialized in high dynamic and high temporal resolution vision with more affordable costs. They are composed of cells that are not able to sense colors, but respond asynchronously to a change in light intensity. Therefore, at each time step, in contrast to standard RGB-D cameras which outputs a synchronous image of pixels, event cameras output, asynchronously, a sparse point cloud, as only pixels that undergo a change of intensity are triggered. This allows these cameras to be very efficient for high dynamic range and high temporal resolution tasks, such as hand gestures recognition [14], tracking, simultaneously localization and mapping or structure from motion [15].

Eye gaze analysis: Gaze detection can also be used for nonverbal communication. In [16], authors use an eye-tracking system, combined with an object detection system to infer which part of the object to pick-up based on where the operator is looking. Thus, the operator can indicate to the robot an intention to pick-up an object simply by looking at it. It can also select various actions such as pouring the contents of a cup into another container. The oculometric sensors (e.g. Tobii Pro Glasses 2) generally consist in using glasses equipped with a camera which ensures a precise tracking of eye gaze.

Tactile and haptic: The use of tactile sensors, such as force sensors, provides information about the robot's contacts with the environment. This allows better management of collisions and facilitates the control of robots under stress. Artificial skin systems capable of detecting pressure variations are beginning to appear but are still usually at the state of research prototype. These sensors can be

used alone or with other perception modalities. Thus, in [17] authors combine a tactile sensor uSkin, a vision system and a neural network model to qualify the texture of some materials with a simple image.

In [18] authors developed an artificial skin by exploiting the variation of impedance of a conductive fabric submitted to a certain pressure. This impedance variation is measured by an imaging method: electrical impedance tomography (EIT). Using a neural map it was possible to infer the pressure location of a robotic arm and control it by touch.

Giving tactile capabilities to a robot can also help improving its grasping abilities. Thus in [19] authors use the Biotac sensor on a robotic gripper to determine the normal and tangential force during the gripping of different objects. From the tangential force information, they were able to create a controller that adapts the gripping force to prevent the object from slipping when it is carried, while normal force information prevents the object from being crushed.

Force control: Using force and torque sensors, it is possible to teach trajectories, with kinesthetic learning, to a robot in a fairly natural way. By adapting its compliance, a robot can be controlled in effort by an operator. The operator can show the robot a sequence of points. The robot can then interpolate a trajectory. Combined with reinforcement learning, this type of control can provide demonstrations that can accelerate learning for a given task. Authors in [20] leverage a force/torque sensors mounted on a collaborative robot for a precise and smooth hand guiding at the end effector level. Such methods can be leverage in kinesthetic learning. In [21] for instance, authors teach a robot how to insert a part by demonstration and deep reinforcement learning. They first record human movements sequences via effort control. The robot then learns by reinforcement the insertion task by leveraging demonstration data, which accelerates the convergence of the network.

6.1.2 Verbal communication

As with images, deep learning techniques have made great progress in speech processing, both in speech recognition and speech synthesis. The interest of speech recognition and synthesis in human-robot communication is the ability to program a robot in a natural way and share information at a high-level. That's why most of the IRL decided to exploit speech as their main modality.

Microphones are generally used for speech recognition. One or more microphones may be used de-

pending on the application. For example, source location requires several microphones. In the context of human/robot interaction, in addition to the ability to capture the frequencies of the human voice, the microphone must also be robust to surrounding noise. Among the most noise-robust technologies, we can cite

1. unidirectional microphones, which focus the recording only in the direction of the speaker, thus limiting the phenomena of reverberation and external noise.
2. Throat microphones or laryngophones which convert vibrations in the throat into speech sounds. Some studies show that voice recognition capabilities in noisy environments, such as industrial one, are far superior to traditional microphones [22]

Voice classification can be relevant in the context of human-robot collaboration because it can allow a robot to detect who is speaking based on its voice. For instance, in [23] authors propose a neural network architecture based on LSTM to obtain an encoded representation of the voice, in order to classify different speakers.

Speech synthesis capabilities are also interesting. Its use in social robots is common and could be interesting in an industrial context. By integrating a vocal synthesis, the robot could then share its decisions not only visually through a screen but also naturally and orally.

Choice of sensors for validation

After listing some of the main sensors and associated interaction modalities, we have selected the ones that seem the most useful for a first integration and validation in the architecture. We want to ensure the most possible natural communications between man and the IRL agent. This requires comparing sensors and signal types to determine which ones are the most relevant and how they can be combined for more robustness and complementarity. The main considered criteria to fulfill or interaction specifications are the following:

1. Potential perceptions after processing signals must be semantically meaningful and natural for a human
2. Sensors must be as non-invasive as possible to be accepted by the people interacting with the robot.

3. Programming and processing must be simple and fast to allow near real-time communication.
4. Sensors must be robust enough with respect to environmental disturbances.

The choice of signals was centered on humans, therefore we fostered natural and non-invasive communication means criteria. Of course, the choice of sensors is likely to depend on each individual specificities. For instance for a disabled person, such as someone that cannot speak or cannot move, even if a BCI can be seen as invasive, it could be the best sensor for human/robot interaction. For our validation, we decided to consider a valid worker for our sensors choices. Table 6.1 shows a qualitative comparison of the proposed signals and modalities according to these criteria. The sign "+" (respectively "-") means that the signal is evaluated positively (respectively negatively) for the criteria. Some commercial sensors at the time of writing are also provided as illustration. Our sensors choices are represented in green.

One of the most natural ways of non-verbal communication is through gestures which can be easily detected by vision sensors, provided there are no occlusions. Vision sensors are among the least invasive and most easily acceptable sensors, because unlike wearable sensors (BCI, EMG, EIT or motion capture sensors) it is not likely to interfere with the operator's movements, or to lead to cumbersome contacts. For their versatility and ease of use for environment understanding, we have considered using RGBD camera.

Speech remains our main way of communication. Therefore it is essential to equip robots with spoken language processing capabilities in order to ensure a natural communication for the high-level components of the architecture. As for vision, microphones are relatively non-invasive sensors. To limit the phenomena of ambient noise, we have chosen a unidirectional headset.

Table 6.1: Summary table of sensors and perception modalities with qualitative comparison. Green sensors were chosen for the validation and the implementantion of the architecture

Modality	Sensors	Sensory input	Perception and semantics	Robustness	Acceptability (non cumbersome or invasive)	Ease of integration	Commercial sensors example
Vision	RGB camera	2D Image	Pose, Movement, Gestures, Facial detection, Emotions, Scene analysis,	++ (sensible to occlusion/more or less to light changes)	+++ (natural modality and without cumbersome wearable sensors)	++ (important pre/post-processing to extract useful semantics, but we can extract a lot of information and there is a rich ecosystem of devices and libraries)	Intel Realsense D435, Photoneo
	RGBD camera	2.5D Image					DAVIS346, DVS-Gen4
	Event camera	Light intensity variation (asynchronous output)					RPLIDAR, Velodyne,
	LIDAR	3D Point cloud	Dictionary of pattern, meaning		+ (use of tags is simple but does not scale well when a lot of tags are necessary)		Tags, QR code, ArUCo code
	Camera + scannable figures	Predefined patterns			Classical microphone		
Sound	Omnidirectional microphone	Speech (air propagation)	Speech, emotion from speech	+ (sensitive to noise in the environment, voice reverberation)	++ (natural modality, wearing a headset can be slightly cumbersome)		Jabra headset
	Unidirectional microphone			++ (less sensitive to noise)			
	Throat microphone (throat vibration)	Speech		+++ (far less sensitivity to noise, ideal for noisy industrial environment)	-- (wearing a throat microphone can be very cumbersome)		
Biological	EMG	Muscular electrical activity	Gesture	+ (sensitive to contact between skin/electrodes)	+ (forearm wearable device)	++ (data is usually accessible and at a high rate)	MyoWare sensors
	EEG	Neurons activity (skin based BCI)	Emotions, mental images, SSVEP detection	-- (sensitive to individual change, skin/electrodes contact, involuntary muscle activity)	- (head wearable device, several electrodes, operator often needs to learn)	- (data seems not always easily accessible in current commercial headsets which focus on specific applications)	Muse/Emotiv/Open-Bci BCI
	EIT	Impedance variation	Gesture, images	- (mostly sensitive to skin/electrodes contact)	+ (forearm wearable device)	--- (not easily accessible technology)	Not to our knowledge
	Motion capture	Tracking of wearable frames attached to the human body/face	Gesture, pose, face and emotion detection	+++ (no risk of self occlusion)	--- (wearing a complete motion capture device is cumbersome)	++ (good proprietary solution, seems easy to setup)	Xsens
	Oculometric glasses	Ocular tracking	Visual attention	++	- (wearing glasses is slighty cumbersome, but could be integrated to existing protections)	++ (good proprietary solution, seems easy to setup)	Tobii Pro Glasses 2, HoloLens 2
Tactile	Tactile skin	Variation of diverse physical quantities	Texture, touch	++	++ (touch is a natural interaction means)	--- (quite new technology, not very accessible)	Biotac
	Force/torque sensors	Force, torque				+++ (accessible as already integrated in most collaborative robots)	IIWA/UR proprioceptive sensors

6.2 Perceptual and acting modules integration

The chosen sensors are then leveraged by the IRL agent thanks to several perceptual modules we integrated in the architecture for workspace understanding and human/robot interaction. Specifically we integrated modules for speech recognition, semantic processing, pose and gesture estimation, and teaching from demonstrations.

6.2.1 Speech recognition and understanding

Speech recognition: For our validation, we adapted an online speech recognizer, Google Speech platform combined with a voice activity detector (python interface to WebRTCvad)¹, in order to carry out speech to text STT (Speech To Text) part. Spoken words are processed one by one and are ordered by confidence level according to speech recognizer. Speech recognition is often faulty and there is no way to update the speech recognizer. Therefore, we exploit a predefined base of written words to check if the recognized words are compatible with the current working domain. If it is not the case, the human agent has to repeat.

Semantic analysis: Once the utterance has been converted in textual entry, it is possible to infer human intents by exploiting a simple natural language understanding module based on FLAIR [24]. FLAIR implements an architecture based on bidirectional LSTM networks for text analysis [25]. The architecture is based on:

1. A first bidirectional LSTM network (BiLSTM) that has been pretrained on a very large and unlabeled text corpus. The training consists in predicting the next character from a sequence of characters. At the end of the training, the network is then able to provide a representation of each word incorporating the context related to the surrounding words.
2. The representation from this first network can then be used in a second network for semantic processing tasks

Pretrained neural networks are used to parse semantically the utterances, with Part Of Speech (POS) tags. For now, we also assume that the parser is perfect and that sentences are simple as

¹<https://github.com/wiseman/py-webrtcvad>

we only exploit verbs, nouns and prepositions. However, this is sufficient to validate the main principles of our architecture, perceptual grounding and the interaction between modules.

6.2.2 Prior object segmentation and tracking

We want the IRL agent to be able to learn online new unknown objects and to reason about them. However, this is possible only if the IRL agent can first detect object locations and create a proto-object in the working memory. In order to detect the object, we rely on a hand crafted module that exploit a pretrained agnostic object detector neural network (tensorflow hub)³, and computer vision algorithm based on depth and color of a RGBD vision system. Moreover a very simple tracking algorithm is implemented based on centroid distance of detected proto-objects.

6.2.3 Joints pose estimation for non verbal interaction

To build a human aware IRL agent, one of the most important feature is to be able to detect the human pose and location. To build real-time, interpretable representations of humans in our interactive setting we developed a first module to detect human pose by integrating Openpose [26, 27], pretrained deep neural network predicting joint pose estimation. It takes as input an RGB image and output joints estimation. We also exploit the depth channel of the RGB-D camera in order to be able to locate the joints in the robot frame. Joints estimations are then at the base of the non verbal communication. Vision often conveys a lot of information that we do not naturally explain by speech but rather with body cues such as pointing gesture. Pointing is indeed a very important non verbal cue that is commonly used to focus on the same spacial location. By exploiting the joint poses output of the Openpose based module, we track the current pointing location of the dominant hand. When the human points to a salient object during a learning interaction, it is assumed that the IRL agent should *focus its attention* to the pointed object or location. By this way there is a strong but natural signal that helps the teacher describe the non verbal cues of the tasks for the learning by demonstration module.

6.2.4 Location affordance learning

One of the upmost task for an industrial robot manipulator is grasping and manipulation. This motivates a module to teach the IRL agent how to grasp unknown object in a reconfigurable way. We

6.2. PERCEPTUAL AND ACTING MODULES INTEGRATION

integrated to that purpose the learning by demonstration module detailed in section B.3.2. We recall that this module aims at giving the IRL agent the ability to learn visually specific affordance locations on an unknown object. It exploits object shape information within a natural interaction. By showing the agent authorized and prohibited locations on an object, a human can teach how to perform a task oriented grasping according to its preferences or the task specificities. For instance one could require the robot to avoid the fragile parts of an object.

Figure 6.2 and 6.3 recall the main stages of the deep network learning pipeline algorithm for a wrench.

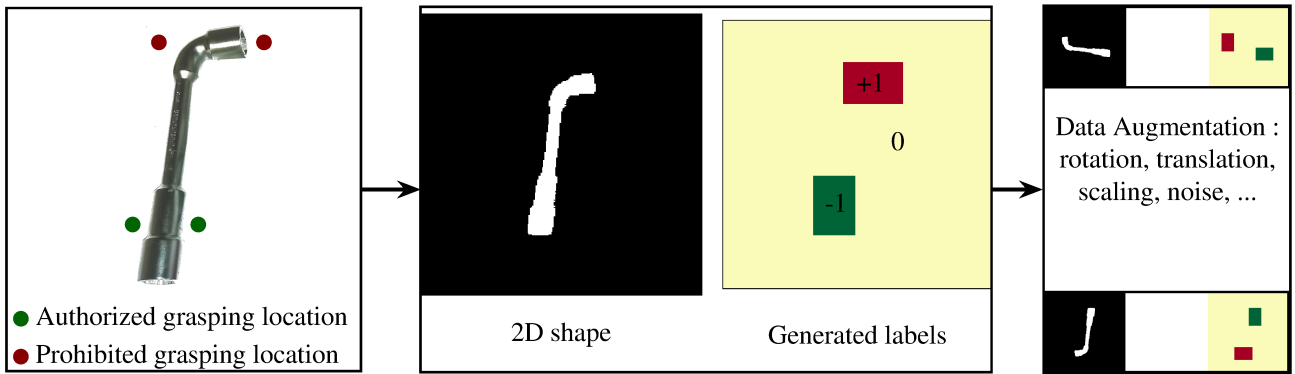


Figure 6.2: Learning authorised and prohibited grasping location : one demonstration (couple input I, label L) and data augmentation pipeline

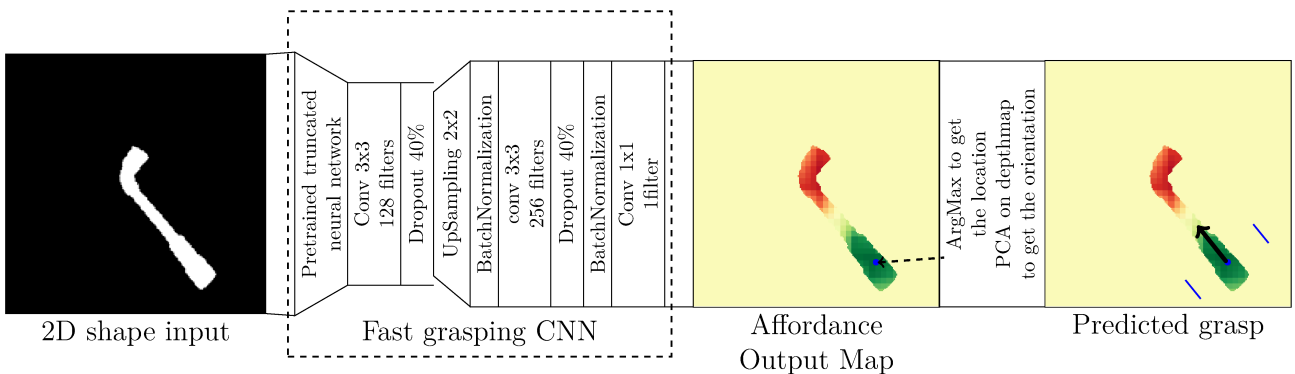


Figure 6.3: Convolutional Neural Network (CNN) pipeline and prediction illustration. The approach used is a regression one, where CNN outputs a pixel-wise affordance map (pixels values are between -1 and 1).

6.2.5 Acting

In practice, modularity in robotic architecture software and IRL agent is usually promoted through the exploitation of message passing and data marshalling based open source middlewares. Such middlewares are for instance ROS [28], YARP [29] used mainly in iCub robots, LCM [30] or proprietary ones such as NAOqi from SoftBank Robotics. They are used for data exchange between modules and actuation of robots. The choice of the middleware can depend on several technical requirements such as accessibility of source, programming language, available packages for a specific robotic platforms, integration with simulation tools, requirements in terms of true real time ability, resilience to networks disturbance for distributed systems. Authors in [31] review such requirements. In our architecture, we mainly focused in simple integration of reasoning, learning and acting without true real-time requirements and in a single computer system. Those parts we developed in python3. Acting on the real robotic platform, however, is provided by bridging on a specific robot middleware, ROS in our experiment. It is indeed one of the most used open source middleware with a strong community support and with up to date package for our robotic platform (UR robots). Moreover ROS comes with several tools that are helpful in terms of interface and debugging tools to visualize the agent inner processes and perceptions.

6.3 Validation scenario

The current architecture has been validated on a UR10e 6 DOF collaborative robot. The robot leverages prior knowledge to learn variations of the task to give, for different objects and according to human grasping preferences. This prior knowledge is integrated at different level and is presented in Table 6.2. As stated in our specifications, learning is done through an online, mixed-initiative, incremental process taking into account human preferences. Table 6.3 presents the main unknown and the knowledge that is acquired at the end of the interactive teaching scenario. Table 6.4 details the scenario and shows the incremental and interactive learning process which leverages both prior knowledge, learned information and humans demonstrations or instructions.

The interactive teaching scenario used for validation can be decomposed in two main phases :

- An interaction with a human H_1 to validate the IRL's ability to leverage knowledge, to ask only for the missing knowledge and to learn incrementally variations of the task "to give" for different

6.3. VALIDATION SCENARIO

objects.

- An interaction with a human H_2 who is unknown to the IRL and have different characteristics and preferences than H_1 . The task "to give", which is now a known task, is asked by H_2 for an object learned with H_1 . This phase, during which the IRL naturally asks and learns H_2 's preferences, validates adaptation to individual.

The teaching scenario is initialised by H_1 . This is done by presenting the robot an ArUco tag [32, 33] linked to a unique identifier. The IRL agent has only the prior knowledge given in Table 6.2. During the interaction H_1 asks the robot to give him a screwdriver but "give" is an unknown task and "screwdriver" is an unknown object for the IRL agent. By natural interaction with H_1 , the IRL agent incrementally learns what is the task "to give" and a skill to solve it. During this process, the IRL agent learns visual features of the screwdriver and a grasping location affordance by asking H_1 demonstrations and leveraging our developed learning from demonstrations module [34]. Finally, the IRL agent gives the screwdriver by locating and moving to the right hand of H_1 in order to adapt to its characteristics. Then, H_1 asks the agent to give him the wrench. As now the IRL agent know what "to give" is, it only learns the specific missing information: what a wrench is and a grasping location affordance, specific to H_1 (the head of the wrench). This second part highlights the modularity and reuse abilities of the learned representations.

In the second phase, H_2 is also identified by presenting an ArUco tag. This human is unknown to the agent and is asked for information about its characteristics. H_2 informs the agent about his name and that he is left handed, leading to an update in the database. H_2 then asks to give the wrench. The IRL agent can leverage all its knowledge about the "to give" task by just learning H_2 's preferences. It learns that H_2 prefers the wrench to be taken by the tail by requesting H_2 for a grasping demonstration. Finally the IRL agent adapts to H_2 by taking the wrench by the tail and giving it to the left hand of H_2 . Figure 6.4, illustrates preference adaptation according to the grasping location affordance.

Thanks to the provided human demonstrations, neural network is trained to efficiently predict the grasping affordances (prohibited, neutral and authorised locations) for the require object according to grasping preferences. Concretely the respective branch of BT's for H_1 and H_2 points towards a specific set of weights stored in a relational manner. This is done through the incrementally updated database,

6.3. VALIDATION SCENARIO

with respect to identifiers, gathering known characteristics and preferences for each individual.

A video of demonstration can be found [here](#)²

Table 6.2: Overview of prior knowledge

Representa- tion	Prior knowledge built in the architecture (Step -1)
Perception	Features extraction from pretrained neural networks Sensory segmentation abilities : background removal, proto-object segmentation Human pose recognition ArUco tags detection Word recognition abilities from Speech to text (STT) Semantic analysis with a base communication protocol
Actions/- tasks	to pickup(obj) as a behavior tree (BT) to putin(location) as a BT
Preferences	Humans have preferences and characteristics, H_1 is known and is right-handed

Table 6.3: Synthesis of what will be learned during the incremental interactive learning process of the unknown task to give

Representa- tion	Unknowns	Learned knowledge
Perception	wrench and screwdriver	perceptual features and grasping location affordance of wrench and screwdriver
Actions/- tasks	to give(obj)	to give(obj) as a BT
Preferences	Affordances and acting preferences	Preferred grasping affordance (wrench caught by the head or the handle) Adaptation to give the object in the dominant hand

²<https://www.youtube.com/watch?v=EAuLMnQULBO>

Table 6.4: Detail of the learning process during the incremental interactive learning process of the unknown task to give

Interaction steps	Unknowns \rightarrow Learned Knowledge	Leveraged knowledge	Human interactive intervention
Step 0: <i>H₁ initiates interaction</i>		<ul style="list-style-type: none"> Built-in prior knowledge with vision and speech (BPK) see table 6.2 Database containing <i>H₁</i>'s characteristics and known preferences. 	ArUco identifier
Step 1: <i>H₁ asks: "give screwdriver"</i>	<ul style="list-style-type: none"> to give(obj) \rightarrow new goal G1: give = In(screwdriver, hand) 	<ul style="list-style-type: none"> Built-in prior knowledge with vision and speech (BPK) 	Speech recognition and semantic analysis
Step 2: <i>H₁ asks: "The goal is screwdriver in hand"</i>	<ul style="list-style-type: none"> screwdriver \rightarrow perceptual features 	<ul style="list-style-type: none"> Built-in prior knowledge with vision and speech (BPK), G1 	Pointing demonstration with speech validation
Step 3: <i>H₁ explains: "pickup screwdriver"</i>	<ul style="list-style-type: none"> to give(obj) \rightarrow pickup(obj) + ... 	<ul style="list-style-type: none"> BPK G1 	Speech recognition and semantic analysis
Step 4: <i>H₁ shows its preference</i>	<ul style="list-style-type: none"> to give(obj) \rightarrow pickup(obj) + ... affordances preferences of H₁ for screwdriver \rightarrow screwdriver affordance as <i>H₁</i>'s preference (the agent stores specific neural networks weights) 	<ul style="list-style-type: none"> BPK G1 pickup(obj) needs grasping affordance 	Authorized and prohibited locations demonstration with speech validation
Step 5: <i>H₁ explains: "put in hand (screwdriver)"</i>	<ul style="list-style-type: none"> to give(obj) \rightarrow to give(obj) = pickup(obj) + putin(hand) 	<ul style="list-style-type: none"> BPK G1 putin(loc) dominant hand of <i>H₁</i> 	Speech recognition and semantic analysis
Step 6: <i>H₁ asks "give wrench"</i>	<ul style="list-style-type: none"> wrench \rightarrow wrench perceptual features 	<ul style="list-style-type: none"> BPK new goal G2 = give(wrench) 	Pointing demonstration with speech validation
Step 7: <i>H₁ shows its preference</i>	<ul style="list-style-type: none"> affordance preference of H₁ for wrench \rightarrow grasping location affordance as <i>H₁</i>'s preference 	<ul style="list-style-type: none"> BPK goal to give(obj) whose sub-action pickup needs a grasping affordance dominant hand of <i>H₁</i> 	Authorized and prohibited locations demonstration
Step 8: <i>H₂ initiates interaction</i>	<ul style="list-style-type: none"> name of H₂ dominant hand of H₂ \rightarrow <i>H₂</i>'s characteristics in database 	<ul style="list-style-type: none"> BPK 	Keyboard entries
Step 9: <i>H₂ shows its preference</i>	<ul style="list-style-type: none"> affordance preference of H₂ for wrench \rightarrow wrench affordance as <i>H₂</i>'s preference 	<ul style="list-style-type: none"> BPK new goal G3 = give(wrench) to give(obj) + need of grasping affordance for pickup dominant hand of <i>H₂</i> 	Authorized and prohibited locations demonstration with speech validation

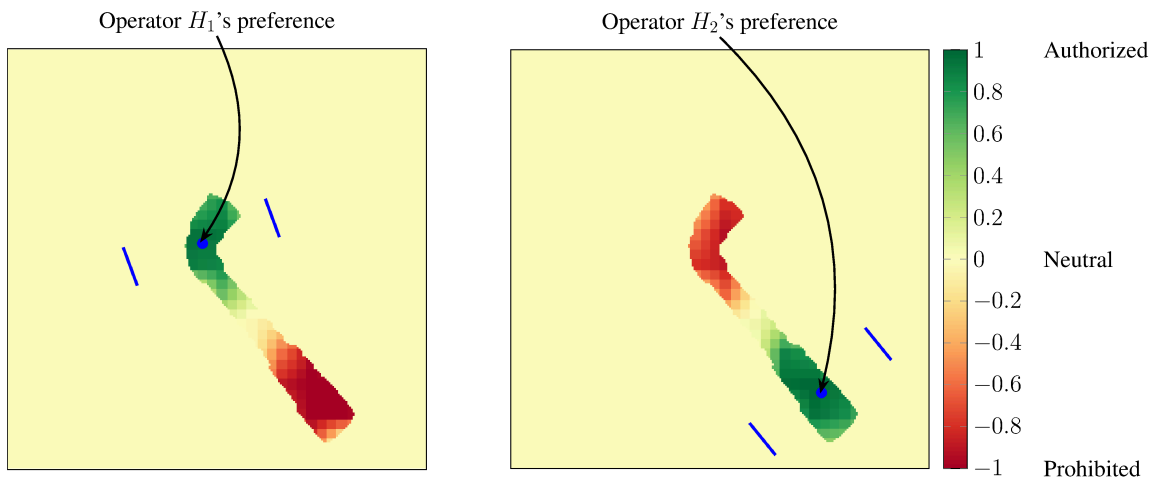


Figure 6.4: The IRL agent can dynamically adapt its affordance prediction according to the interacting human as shown here on a wrench example. On the left, H_1 has taught the IRL agent to grasp the wrench by the head. On the right, H_2 has taught the IRL agent to grasp the wrench by the tail. The agent can then dynamically choose the preference according to the interacting human.

6.4 Conclusion

In this chapter, we provided an implementation and a validation of several of the main architecture specifications. The core architecture was developed in python3 to be easily linked with different modules (perception, learning, ...). We made a real robot learn an unknown task, by building procedural and perceptual knowledge taking human preferences and characteristics into account. The use of behavior trees provides an efficient way to handle knowledge representations, task sequences learning and execution. Indeed, by this way, when the agent faces failures while executing behaviors, it only has to request for the missing features. Information is provided interactively by human in a mixed initiative teacher/learner setting. Using speech and gestures as means of communications and by using integrated modules such as our learning by demonstration module, non-programmers experts can naturally reconfigure the robots. Thus, we have shown that the choices we made have brought our architecture closer to our smart robotic assistant specifications. However, there is still a long way to go before reaching the ideal SRA, such as integration and validation of multimodal uncertainty

6.4. CONCLUSION

in the overall architecture. The next chapter (chapter [7](#)) concludes our work and discusses several perspectives.

6.4. CONCLUSION

Chapter 7

Conclusion and perspectives

Contents

7.1	Main points	128
7.2	Integration of an uncertainty aware grasping module	130
7.2.1	Grasping under uncertainty	130
7.2.2	Trustnet	131
7.2.3	Extension for active learning by demonstration	133
7.3	Long terms perspectives	136
7.3.1	Continuous active learning setting	136
7.3.2	Reasoning about specific uncertainties	136
7.3.3	Multimodal fusion	137
7.3.4	Improve preferences generalization	139
7.3.5	Improve behavior models in terms of learning and representations	140
7.4	General conclusion	144

The work carried out throughout this thesis has led to the design and development of a first core architecture for skills learning in a collaborative industrial context. As a conclusion, we review in section 7.1 our main specifications and the developed methodology to fulfill them. We balance our results by underlying the current limitations of our system and how it opens many research paths. Section 7.2 presents how a first uncertainty aware, deep learning module developed by Laurent Bimont, a former PhD student, can be integrated with our learning from demonstration module. We present finally in section 7.3, thoughts on longer terms perspectives and cross domain research that could benefit the architecture.

7.1 Main points

The main objectives of the thesis were to build a Smart Robotic Assistant for industrial collaborative robotics. Inspired by works in cognitive architecture and interactive robot learning, we defined several specifications for our cognitive architecture to contribute to Interactive Robot Learning and Human Robot Interaction (HRI). Throughout this thesis, we discussed the main building blocks of the core cognitive architecture and the integrated learning modules to fulfill them. The IRL agent must:

- reason and have at least partial explanations abilities
 - The IRL agent has been endowed with several symbolic memories for an interpretable representation of high-level knowledge and reasoning over those representations.
 - An ontology is used to share coherently representations across the different memories
 - The choice of Behavior Trees as a behavior model eases the representation of a goal directed, procedural knowledge and allow a certain level of reactivity.
- learn quickly and incrementally a new task from low level to high-level abstractions
- leverage a prior knowledge base for tasks execution and learning online
 - Use of symbolic memories allows to leverage symbolic prior knowledge.
 - Behavior Trees help to build a modular system and therefore promote quick behavior reuse within the decision process architecture
 - Transfer learning with deep neural networks is leveraged for learning and interpreting the world and human interactions.
 - Transfer learning also allows to learn fast, to reconfigure behaviors as we showed with our learning from demonstration module
- interpret and react to human interactions in real-time
- interact intuitively with non-programmer experts
 - Several perceptual learning modules, developed and adapted from the literature were integrated into the architecture.

7.1. MAIN POINTS

- Failure handling through the behavior model allows to teach the IRL agent in a mixed initiative human/robot interaction loop, an unknown task at both high and low levels.
- The choice of sensors and perceptual modules for validation, based on speech and vision of gestures, were chosen to exploit natural communication means for information sharing
- The developed learning from demonstration module eases non verbal interactions.

- adapt to preferences and specificities such as disabilities
 - The architecture is human centered and therefore, the IRL agent specifically represents human preferences and learns them during the interaction.
 - Validation has been made on a real-world system to assess the relevance of the approach.

- handle uncertainty in moto-perception and its knowledge
 - A state of the art was made on uncertainty definition and its estimation with deep learning module
 - Uncertainty handling and reasoning is a work in progress and will leverage extension of the behavior model and aforementioned uncertainty estimation techniques.

This work is a first step in the development of a powerful IRL cognitive system for a non-programmer expert human to teach, more naturally, flexible behaviors to an industrial collaborative robot. Overall, we have built the architecture with modularity in mind, as using modular components is key for better understanding, confidence assessment and the potential evolution of the architecture. Teaching behaviors is a complex task, so we took inspiration from the iterative learning process in humans. Robots are taught incrementally, how to represent the world with and for human agent. This type of learning is promising as it gives much more control and likely confidence to the end users, by teaching the robots behaviors according to their needs. In the long run, it could help build more interpretable representations of the real-world environment. Of course, the ideal IRL is still far from being achieved. We develop in the next section some limitations and relevant paths of research, with a focus on ongoing work in uncertainty integration.

7.2 Integration of an uncertainty aware grasping module

In section 6.2.4, we developed a module that could learn grasping location affordance with a good accuracy level. However, the agent can still predict wrong grasping location (for example, if the training dataset is not diverse enough). In an industrial scenario, for safety concerns, it can be much more acceptable not to act rather than doing wrong. In that context, we want the IRL agent to be able to measure its confidence in grasping prediction. If it is not sure, it does not act and this specific failure can interactively be handled by asking for new demonstrations. Once the IRL agent is confident enough, it can act. Therefore, to exploit uncertainty in the architecture and later in a multimodal way, we have started to develop a specific module to evaluate uncertainty at the same time as learning grasping parameters. It consists in two neural networks: *Graspnet* presented in section 4.1.4 which is responsible for learning the rectangle grasping parameters and *Trustnet* which is responsible for predicting the uncertainty of Graspnet. It is a reproduction of the work done in our laboratory by Laurent Bimont, a former PhD student. Constraints due to the Covid situation, limited the maturity of this work. Therefore it has not yet been valorized in a publication and integrated into the overall architecture. We first introduce in section 7.2.1 grasping under uncertainty before presenting a global overview of Trustnet module and its use for grasping under uncertainty in section 7.2.2. Main experimental results are provided in section 7.2.2. Finally, in section 7.2.3, we present how it can extend the IRL learning from demonstration capabilities. Annex A.2, provides the experimental details obtained by Laurent Bimont.

7.2.1 Grasping under uncertainty

In the deep learning community, estimating network uncertainty has been addressed in various works (section 5.2). As to the best of our knowledge, few papers have investigated uncertainty measurement of vision regression problem in grasping, we decided to focus on the direct regression of the grasping parameters along with uncertainty. Learning to grasp and prevent failed attempts has already raised interest in the robotic community. Several approaches rely on leveraging geometric characteristics of the grasping contact area [1–3] to compute a grasp quality metric. To obtain good performances, those approaches require knowledge about mechanical properties and a 3D model of the objects and the gripper. Such information are hardly available in an online interactive setting

such as the one we face with an IRL agent. Predicting a *confident* grasp directly from a real-world image can therefore be a very flexible approach. In that context, we have seen in section 4.1.4 that an important part of current research explores deep learning. However, compared to geometric and analytical methods, traditional deep learning approaches are much more opaque. This has led to bring uncertainty estimation techniques with deep learning to the visual grasping context. For instance, in [4], authors train a network to predict, given CNN features computed on an RGB image, several potential grasps as belief heatmaps, which are then fitted to 2D Gaussian Mixture Models (GMM). This allows the model to predict several possible grasping poses which are then ranked according to the fitted GMMs likelihood. The mean of each predicted belief maps represents the center of a gripper plate, while the variance has been related to a measure of uncertainty. The methods gains in accuracy before staggering, as the number of predicted heatmap increases.

7.2.2 Trustnet

With Trustnet, we chose another uncertainty estimation approach as we only want to predict one grasping pose. We have extended a methodology [5] where authors proposed to leverage an external neural network called *ConfidNet*, trained to predict the True Class Probability of a prediction. This technique is related to the external measure approach (section 5.2.4) in uncertainty estimation techniques. Here, TrustNet has been applied to predict an uncertainty associated with GraspNet predictions. The overall module architecture is presented in Figure 7.1. We note $g = \mathcal{G}(X) = (x, y, \theta, \omega, h)$ a prediction of GraspNet and $\tau = \mathcal{T}(X)$ a prediction of TrustNet for an image input X . TrustNet outputs an uncertainty metric representing the probability of failure given the trained *GraspNet*, $\tau = P(Y = failure | \mathcal{G}, X)$ for any input X .

The steps of the main method is the following:

1. We train *GraspNet* on a dataset $D_{train} = (X_{train}, Y_{train})$.
2. Based on a validation set $D_{val} = (X_{val}, Y_{val})$, we build an uncertainty calibration dataset $\mathcal{D}_{cal} = (X_{cal}, \tau)$ where the target value τ is related to a certain grasp (0) or an uncertain grasp (1) determined thanks to the Jacquard metrics (equation A.1 in annexe A.1).
3. Then, this dataset \mathcal{D}_{cal} is used to train *TrustNet* as a classification problem.

4. Finally, for any unseen image X , we can compute the tuple result (g, τ) for an image X .

Since *Graspnet* and *TrustNet* work in parallel (Figure 7.1), it makes this technique non-intrusive for *GraspNet* and with little inference time overhead.

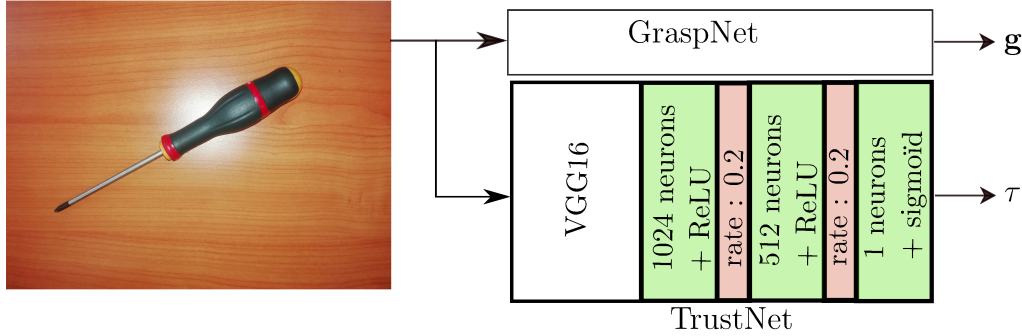


Figure 7.1: *TrustNet* architecture

The Neural Network architecture used for *TrustNet* is similar to the *GraspNet*'s one except for the last layer which outputs a single value with a sigmoid activation function to provide an uncertainty measure. Since we trained *TrustNet* as a classification Network, we used the classical binary cross-entropy loss between the predicted τ and the true one $\hat{\tau}$ as loss function:

$$\mathcal{L}(\tau, \hat{\tau}) = \tau \log \hat{\tau} + (1 - \tau) \log \hat{\tau} . \quad (7.1)$$

Qualitative analysis.

To qualitatively estimate an uncertainty metric τ , we plot in Figure 7.2 the histograms distributions of bad/good predictions according to τ , on the Cornell dataset. Different estimations methods were compared against *TrustNet*.

1. *Dropout* and *Simple Ensemble* methods do not separate well good and bad predictions. Moreover, their respective distributions along τ look similar.
2. *Proper Scoring* methods show two partially shifted distributions which is a first step toward a clear separation. We can expect a good failure prediction capacity for those approaches, in particular for the *Ensemble Proper Scoring* method for which the drift is higher.

3. *TrustNet* also shows shifted distributions with more good grasps located at a low level of uncertainty. Distributions of good and bad predictions can not be clearly separated, but once again we can expect a good failure detection performance for this metric. We should note that *TrustNet* is the only metric proposing a bounding uncertainty metric between 0 and 1, making it easier to set an uncertainty threshold thereupon.

None of the tested uncertainty metrics clearly separate the good prediction distribution from the bad one. But still, *TrustNet* method shows promising behaviors for failure detection.

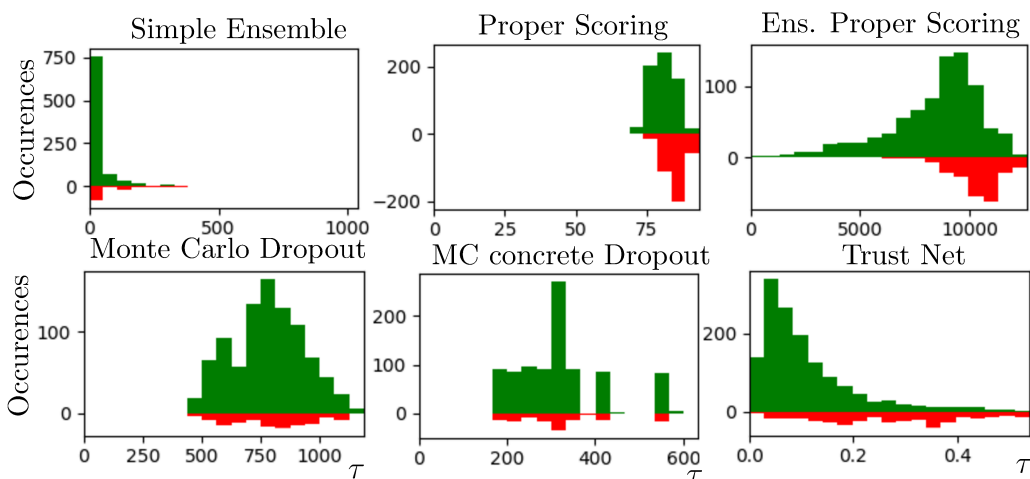


Figure 7.2: Histogram of τ for good (green) and bad (red) predictions for different uncertainty methods on Cornell grasping datasets. Bad predictions are represented by negative occurrences

7.2.3 Extension for active learning by demonstration

GraspNet/Trustnet methodology can serve as a base in the architecture to improve grasp quality and safety. We show, as illustration, in Figure 7.3, what we could obtain by integrating TrustNet with our learning from demonstration module. The learning from demonstration would output forbidden, neutral and authorized grasping area from which we derived grasping rectangles while TrustNet can be used to predict an uncertainty estimates for the authorized grasping area. This would allow safer prediction, as now, the IRL agent has a mechanism to decide **to act or not to act**. Moreover, it opens the gate for uncertainty driven active learning of grasping demonstration. In that direction, Figure 7.4 shows how this module could be integrated as a condition node of Belief Behavior Trees (BBT) presented in section 5.4.3. Given the grasping parameters and associated uncertainty, a Failure can

7.2. INTEGRATION OF AN UNCERTAINTY AWARE GRASPING MODULE

be generated according the uncertainty threshold τ_{thresh} . Uncertainty can be propagated back in the tree and if the agent is not certain, it can be solve in the BBT, leveraging active learning. Figure 7.5, presents the high-level integration of the module based on the methodology presented in section 5.3.

In the long term, we hope to reach a success rate of 100% for grasping trials while limiting the number of irrelevant requests. It can indeed generate a cognitive burden for humans and an economic cost if re-configuring the robot is required too frequently. Hopefully, as it increases its experience, the IRL agent will be less and less likely to ask for help.

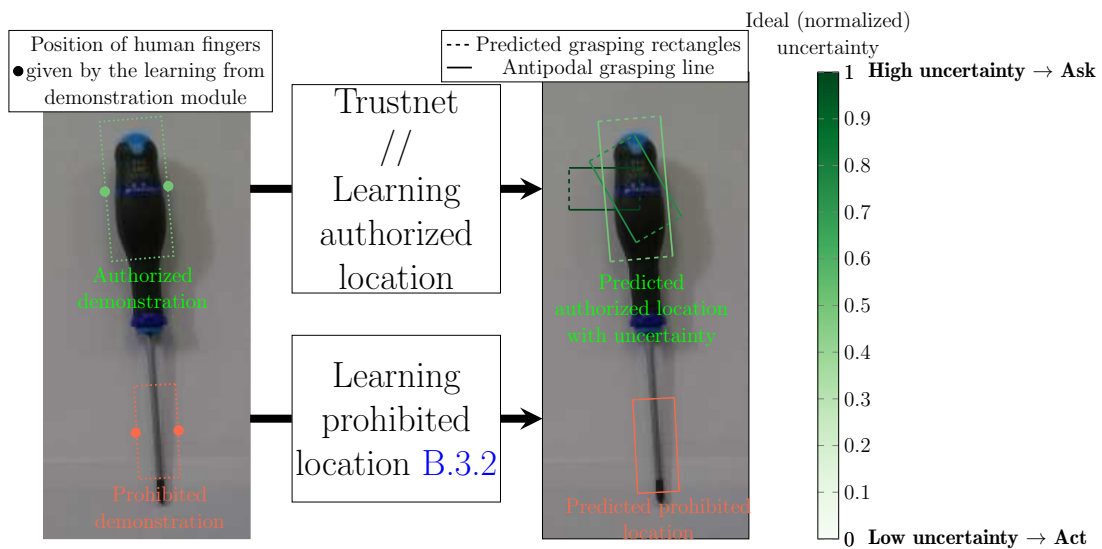


Figure 7.3: Graspnet and TrustNet integration can be used in a task oriented setting for active learning by demonstration.

7.2. INTEGRATION OF AN UNCERTAINTY AWARE GRASPING MODULE

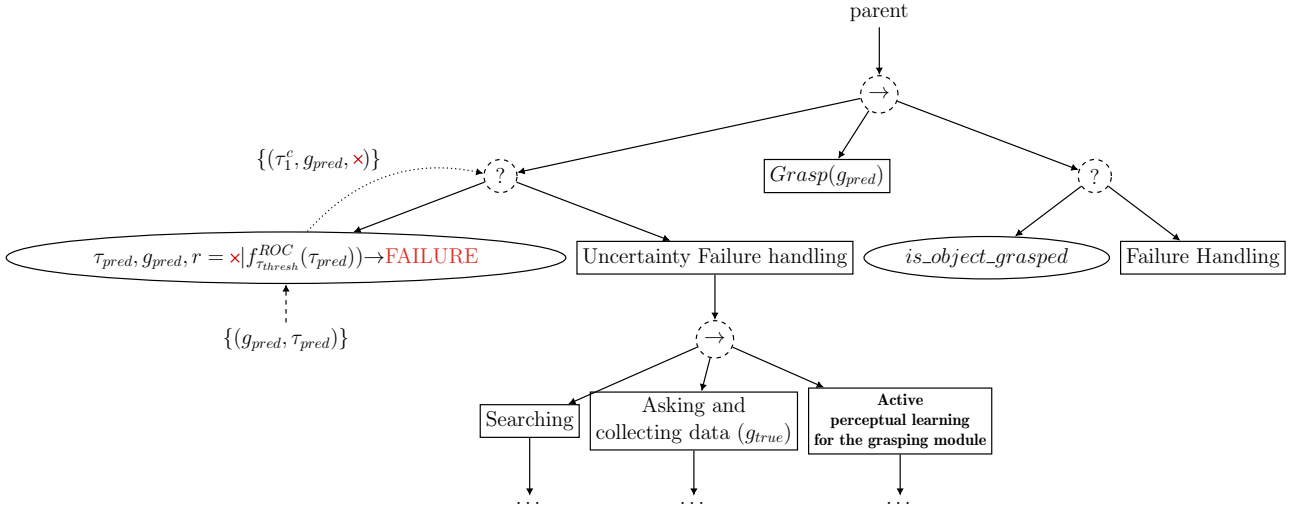


Figure 7.4: Possible integration of the module with BBTs. Uncertainty can be propagated backward and leveraged by the IRL agent for active learning.

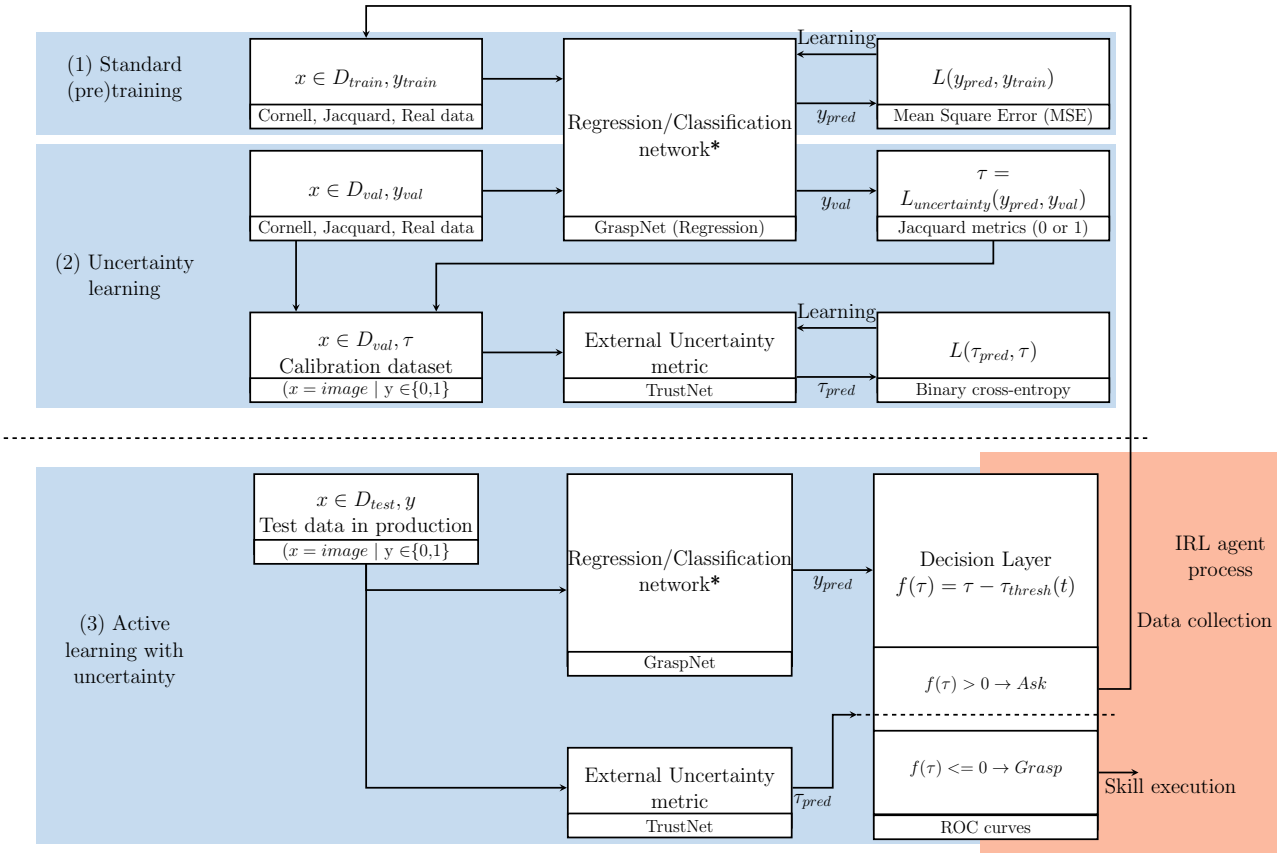


Figure 7.5: Integration and learning pipeline of the TrustNet/GraspNet module with the higher level processes of the IRL architecture.

7.3 Long terms perspectives

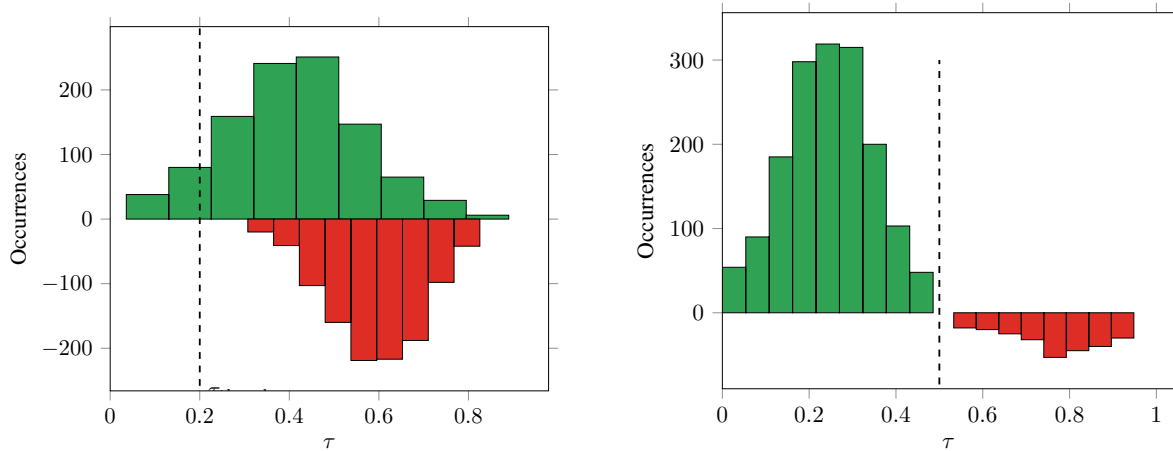
7.3.1 Continuous active learning setting

This active learning pipeline depends on the level of uncertainty of the system, τ_{thresh} (see figure 7.5) and on how datasets can be updated. During the IRL process, in first steps, we aim at drastically limit the number of missing alarms (FN). Therefore, we should start with a rather high $TPR(\tau_{thresh})$ corresponding to a low τ_{thresh} before progressively increasing it as the IRL agent predictive and uncertainty models improve. The idea is represented in Figure B.21 where we illustrate for an hypothetical model the proportion of good predictions (in green) and of wrong predictions (in red) given the uncertainty level. The agent will ask the operator for inputs labels of uncertain predictions. Training on few epochs from the current weights/knowledge will enable the agent to significantly improve its expertise over time. In order to not forget what was previously learned while avoiding the storage of unnecessary data, the IRL agent needs a way to select the most relevant data to keep, as new data are collected. Various techniques based on experience replay, a dataset of past experiences, [6, 7], commonly used in reinforcement learning, could be leveraged to select the most relevant data as suggested in [8].

Ideally, after a few iteration steps, the distribution of good and wrong predictions should be well separated among uncertainty τ (ACC tending to 1), with no missing alarms and less uncertain and wrong predictions. Expertise of the IRL will be seen when asking decisions becomes sparse with an NPV ratio equal to 1, and TN proportion very close to 1. Remaining asking decisions should happen with scarce unseen inputs (particular cases). This would allow to adjust τ_{thresh} as the agent becomes expert. However, defining the right way to do this automatically is open, as the level of acceptable uncertainty is likely to be task specific.

7.3.2 Reasoning about specific uncertainties

In terms of architecture integration, in the long term, thinking about distinction between aleatoric and epistemic uncertainty in the architecture seems important. Indeed, if the IRL agent has to deal with a task inherently stochastic, it should explore more, usually by asking a demonstration to a human, in order to reduce its epistemic uncertainty. Once the epistemic uncertainty is low, collecting more perceptual data by asking demonstrations to a human becomes meaningless and the agent can



(a) Histogram of predictions are not well separated at the beginning of learning

(b) Ideal histogram of predictions. As the agent increase its expertise, histograms should become more and more separated and pushed respectively towards minimum uncertainty for good predictions histogram and towards maximum uncertainty for wrong predictions histogram.

Figure 7.6: Illustration of τ_{thresh} usage to prevent wrong actions, given the uncertainty τ

then effectively compute the aleatoric uncertainty. For instance, if a visual sensor becomes very noisy in low light conditions, the aleatoric uncertainty of a classification model can become very high, even if the networks parameters have been adjusted and that epistemic uncertainty is low. Such model can then lead to specific strategies, such as ignoring vision sensing for decision making.

7.3.3 Multimodal fusion

We have seen that there are many modalities of perception. A complex collaborative application will probably need to combine even more sensors and perception modalities than we have done. Multimodal learning is based on fusing and relating information coming from those different sources. This could be helpful to provide various information and redundancy from different physical sources. Variety is likely to help in a better understanding of the environment. Many sources redundancy can help perform tasks more robustly. An ideal multimodal IRL agent should be able to exploit at least the same modality as humans such as vision, speech, textual data, force or haptic sensing [9, 10]. For instance, haptic could be especially useful in some context such as when the gripper is occluding the robot vision, or for better grasping prediction that could take object texture into account [11].

One of the challenges in multimodal processing is how to deal with data heterogeneity. To solve this difficulty in deep learning, several techniques exist. For example, in [12], authors categorize multimodal learning in deep learning according to three framework:

1. Joint representation: Each modality is encoded by a neural network. The joint representation consists in concatenating the last layer of the networks in order to force a joint representation during the learning process.
2. Constrained Coordinated Representation: The networks are trained in parallel for each modality and then a suitable loss function is used to update the networks. This involves establishing a similarity measure between the two modalities at the output of the parallel networks (called constraint).
3. Encoder-decoder system: An encoder encodes a modality in a latent vector then a decoder generates a sample of the targeted modality. At the end of the learning process, the network has learned a representation to switch from one modality to the other.

In recent works, cross-modality can also be leveraged in a sequential but related manner. For instance, in [13], authors proposed Cross-Modal Deep Clustering (XDC), where a firstly pretrained unsupervised model for a modality (audio) is used to supervise the training of another modality (video).

Sensors are likely to not have the same reliability in dynamic settings. For instance, when the sight is clear, vision based system is ideal, but in highly occluded setting such as insertion tasks, touch, force/torque sensing is likely to be more appropriate. To mitigate these limitations, use of uncertainty can represent a strong cue. That could help drive major improvements in sensing fusion, for active learning and for better decision-making. At high-level, we can distinguish two main views on multimodal sensing and perception in an online adaptive setting: when they *complement* and when they *contradict* each other.

When perceptions are complementary, they can provide redundant or additional information about the perceived concept. Perceptual uncertainty is a convenient tool that can help in this reasoning process. For instance [14], authors adapt uncertainty based techniques in specific audiovisual speech recognition task by exploiting Gaussian mixture model. According to the uncertainty estimation they

can rely on the different streams of data (vision of the mouth and speech).

When perceptions are contradictory, the problem becomes more complex. Indeed, for instance, if a human asks a robot "Give me the screwdriver" while pointing an object, whereas the IRL agent actually sees a wrench. In that case several interpretations are possible:

- The perception by the IRL agent of the pointing hand location, of the tool and of speech are true. As a consequence, the human made a mistake, this is not a screwdriver.
- Some perceptions are faulty. For instance, perceptions of the pointing hand location and speech could be true whereas perception of the tool is faulty. The object is a screwdriver but the IRL agent sees a wrench.

In the first case, the robot should infer, based on its confidence level and its knowledge of the current situation where the human was wrong. Assuming all perceptions are confident, it could ask to repeat the question or suggest to grasp the wrench instead.

The second case is the more complex but is interesting as this type of situation can be leveraged for a kind of multimodal self-active learning. In [15], authors use a Bayesian method, the Independent Opinion Pool, to reinforce or mitigate the overall uncertainty about human intent between several modalities (gesture, gaze, speech and objects recognition). In our example, we have two modules that predict the same information, the object recognition module and the speech recognition module. Assuming the object recognition module is uncertain and the speech recognition module is confident, we can exploit the fact they designate the same object. For example, we could update the object recognition module on behalf of the speech recognition module, without human intervention, by being optimistic and assuming that the speech recognition module must be right. Another approach, more conservative, could be to make the IRL agent asks a question to the human, based on the current situation understanding. The IRL agent could ask "Are you sure ? I don't see a screwdriver ? I see a wrench with uncertainty, should I learn that it is a screwdriver ?".

7.3.4 Improve preferences generalization

As the agent accumulates knowledge with several users preferences, some of them are likely to be redundant. For instance, it is likely that several users will share the same grasping location affordance.

Currently, a new neural network is learned for each new user which means learning and saving a lot of models which predict the same affordance. We need more general mechanisms for better scaling. For instance, when the IRL agent is given a new preference, it should have a mechanism to search in its perceptual memory for similar preference. If such preference has already been encountered, it can just point to this preference without relearning. Several strategies could be investigated:

- The IRL agent could propose a grasping affordance based on past demonstrations. This could simply be done by enumerating previously learned preferences and asking the human to select one of them. However, it could scale badly with the number of preferences. By computing statistical information about previously learned preferences, such as their frequencies, the enumeration could be done after a relevant ranking of preferences.
- On the other hand, the human could be asked to give a demonstration. In order to avoid learn again a similar demonstration, the IRL agent could leverage its prior perceptual knowledge stored as neural networks weights. We need a way to compare the demonstrated preference against previous one. This could be done, by exploiting clustering techniques [16]. The IRL agent would assign the closest known preference. Determining the best metric to compare preferences is likely to be task specific and should be discussed.

7.3.5 Improve behavior models in terms of learning and representations

Concurrency of behaviors: While classical BTs allow a parallel execution of perceptual modalities and actions, they can lead to unexpected behaviors when synchronization is not enforced between nodes as what is encountered in concurrent programming. For instance, controlling a bi-arm robot could require the parallel execution of both arms for a collaborative task. The BT model should be updated to take such setting into account as what was done in [17, 18] to formalize concurrency in Behavior Trees.

Multiparadigm learning with Hierarchical Reinforcement Learning and Learning From Demonstration: Hierarchical reinforcement learning (HRL) decomposes a reinforcement learning problem in several subtasks that can themselves be modeled as reinforcement learning problems. HRL has proved to be more data efficient than standard RL probably thanks to a better exploration ability. The

hierarchical nature of the framework, can help decompose tasks into simpler tasks for learning. Integration of reinforcement learning has been leveraged in several cognitive architectures such as Soar [19]. RL is also exploited in the IRL architecture in [20]. In this systems, the IRL agent can learn and coordinate new behaviors through model-based algorithm corresponding to goal directed behaviors, and model-free RL corresponding to habitual behaviors (a kind of reflexes).

Several extensions have demonstrated that it is possible to integrate RL in BTs framework, such as in video games with QL-BT : [21], BT-RL: [22] or in [23].

Another important point is the fact that, in the current architecture, we have mostly focused on learning *discrete* actions. Each action being a point to point continuous motion computed by a classical inverse kinematic motion planner. Learning lower level continuous action such as in [24] could be interesting and could be modeled with RL based learning modules.

Finally, while we first focused on the integration of learning from demonstration (LfD) techniques in the architecture, having a synergy between LfD modules and RL based modules seems a promising path where LfD would serve as a basis that could be refined by RL based approaches.

Knowledge update and repair: We are currently expecting that the human is an oracle, but in real-world, we can not always expect a perfect understanding between the human expectations in terms of task specifications and what the IRL agent understood. Actually, even human often fail when explaining a procedure to others. We need to integrate what is referred to as interactive misalignment repair [25] where the agent can be triggered by humans, if they see the robot doing the task wrong while it think the contrary. This could be done via specific keywords, such as "stop", "you are wrong", which would help in correcting a past misunderstanding. The structured nature of the procedural memory and episodic memory are likely to help for that types of introspection and for learning. During a repair phase, the agent could simulate what it did thanks to its skills representation and what it remembers thanks to the episodic memory while asking humans for correction.

Causal Learning: We have seen throughout this thesis that building an interactive robotic learning agent requires quite a cross disciplinary approach, merging works from several research area and built in a more general cognitive architecture. An approach, that we had not the time to investigate is *causal learning and discovery* literature. Causal research literature is gaining an increasing traction in the

7.3. LONG TERMS PERSPECTIVES

machine learning and deep learning communities [26]. While machine learning research has usually mostly focused on learning correlations between data, causal learning research aims at formalizing causality relations between data. Cognitive robotics and design of architecture, are likely to benefit from this research area, through relevant connections in terms of world modelling. Yet, it seems there has been only a limited set of work, *in robotics* [27], focusing on this cross disciplinary approach. It is likely that integrating concepts and results from this framework could help learn better representations for our cognitive system.

One of the most prominent framework, the *do causal framework*, was introduced by Pearl [28]. Reader can refer to [29], for a comprehensive introduction in the field. This framework entails to model causality in a hierarchical manner, through the lens of the "Ladder of Causality" [30]. This framework highlights the difference between three layers that intervene in causality: *seeing, doing, imagining* [29] which corresponds to different level of inferences from data observation to higher reasoning level. Table 7.1 presents Pearl's Causal Hierarchy, reproduced from [29] (Table 1.1), with an adapted example. This is done by building and leveraging a specific type of Bayesian Network graphical models called a Structural Causal Model (SCM) (see figure 7.7). In a classical Bayesian Network, we are given several random variables (observable or not observable) and their dependencies are represented through the graph edges. Assuming Markov properties, we can associate each variable with a certain conditional probability distribution (CPD) given its direct parents. In contrast with a classical Bayesian Network, SCM however, use *causal* dependencies in the graph. That means that the graph is built explicitly to represent real-world causal directions where edge direction implicitly encodes time. This nuance is important as we saw in chapter 5, that, the way a variable causes others is linked to dataset shift issues in deep learning.

For instance, let's imagine we want to teach an IRL agent, that a variation of illumination might change how well it perceives the world. Such a representation could help the agent better handle its uncertainty representation by explaining *why* it is uncertain. If its visual uncertainty increases, given the causal model, it might assume that its visual uncertainty was caused by a high illumination. Reciprocally, if the agent detects a change of illumination, but that its uncertainty remains low, it is possible that the predictive model for uncertainty prediction is ill-calibrated. In that context, the causal representation through an SCM, could help correct the model, such as the one in figure 7.7: the Sun causes light that affects objects visibility. There can be some hidden variables, also affecting

visibility, such as dust in the air or material properties of the object. In the non causal Bayesian Network, while probabilities distribution can be computed, they do not represent the true causal structure of the physical phenomena: light affect visibility and not the contrary. Only the SCM allows to represent directly the object visibility probability given the presence of light. Therefore, with the SCM, the agent is likely to build a better and more flexible understanding of the world. Then, the question is how to build such a SCM given observation data. In [29, 30] authors proved that a valid SCM could not be learned given *observational* data alone (seeing layer), but it requires what is called intervention data. An *intervention* (action layer) consists in an external action, which fixes some (usually one) variables that supersedes the current SCM representation. For instance the agent or the human can control the light. This is modeled via the *do* operator which defines a new *mutilated* model which can be used to infer more information about concept causal interaction. Finally *counterfactual* (imaging layer) aims at giving the agent the ability to make hypothesis (without intervention), to imagine what can happen in its model given this hypothesis. Here, if the agent sees nothing, it can make the hypothesis that it could see an object if light is on. By this way, it could decide to turn a the light or suggest a human to do it.

Humans have a natural intuition of causality that is learned throughout of their lives. In that context IRL could be exploited from the robotic agent perspective as a causal discovery mechanism where the human share its knowledge of the world to build the SCM. Conversely notions from causal learning is likely to help build more principled representations on the way we build and develop the different memories. In particular, it might impact procedural memory as precondition and postcondition already embed a notion of causality.

7.4. GENERAL CONCLUSION

Table 7.1: Pearl’s Causal Hierarchy, adapted from Table 1.1 [29]. Appearing correspondences within the IRL architecture design.

Layer Activity	Layer (Symbolic)	Typical Question	Example for an IRL agent	Machine Learning	IRL architecture correspondence
L_1 (Seeing)	Associational $p(y x)$	What is ? How would seeing X change my belief in Y ?	How does a change in light influence my vision ?	Supervised/Un-supervised Learning	Low level motor-perception module
L_2 (Acting)	Interventional $p(y do(x))$	What if I do X ?	What if illumination is low ? (Please show/tell me.)	Reinforcement Learning	Mixed initiative (natural) interaction
L_3 (Imagining)	Counterfactual $p(y_x x', y')$	Why ? What if I had acted differently ?	How should be the illumination if I want to recognize objects ?		Higher level reasoning processes and introspection

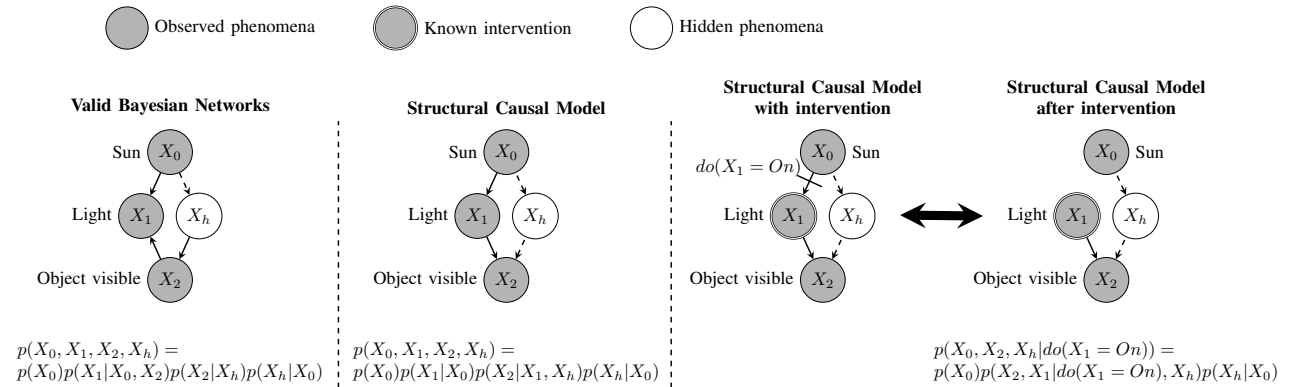


Figure 7.7: Different Bayesian Networks graphical models can represent the same data. But not all graphs represent *causal* relations. Gray variables are observed variables, white one is a hidden variable that could influence object visibility.

7.4 General conclusion

While industrial collaborative robots programming becomes more and more accessible, building adaptable and modular behaviors is still out of reach for most non programmer experts. Allowing any humans to teach cobots complex and personalized behaviors, with natural communications means, would likely ease their acceptability and long-term co-integration in industry of all sizes. We have presented a core architecture prototype focusing on integrating reactive planning, acting and incremental learning of skills in an interactive learning setting. The current development state of the architecture

could validate most of our specifications. Experimental validations were made in teaching task oriented grasping related tasks, with preferences handling to an industrial collaborative manipulator. The ILR agent is able to learn both high-level and low level representations of the task **online**, during a **mixed initiative interaction**. Use of **BTs** as behavior model help build **modular**, explicit task representations which help in **explainability** and **interpretability** of robot behaviors. The cognitive process exploit perceptual modules based on neural networks to learn complex perceptual features such as grasping location affordance on plan. **Learning is fast** thanks to pretrained networks, transfer learning to specific downstream tasks and augmentation techniques with specialized sub-networks. Moreover, as bigger neural networks are hardly interpretable, the use of specialized sub-networks is likely to help **interpret** and **correct** system failures. Thanks to modularity, if a sub network prediction fails, it can be possible to correct only this sub networks without affecting other modules. We have shown that it is possible through a mixed initiative scenario to teach a task and its variation with respect to most of our IRL specifications. Adaptation to **human preferences**, here validated on grasping affordance preferences and adaptation to dominant hand, is an important requirement in our architecture. Indeed, it offers personalized interaction which is likely to help in **acceptability** of robotic systems by the operators.

Our work is a first step towards a smart collaborative industrial robotics assistant. The architecture is opened and extensible to several improvements as integration of better perceptual abilities, language understanding and communication protocol in order to allow richer real world interaction. Finally, as the architecture matures to handle more complex tasks, it will become necessary to test it on more complex setting and with true non-experts, based on HRI metrics [31] and following standard HRI evaluation protocols (chapter 7 of [32]).

7.4. GENERAL CONCLUSION

Appendix A

Evaluation of grasping quality and uncertainty

A.1 Jacquard metric

To evaluate whether or not a grasping prediction $g = (x, y, \theta, w, h)$ is a success, one can use the Jacquard metric [1] with several grasping parameters $g_i \in g_{true}$ which are available for the same image in the dataset. For an image $x \in \mathcal{I}$, valid grasping representation g_i are converted to their rectangle representations R_i . Then, one can compute the area of intersection over union of the predicted grasping rectangle R_{pred} from g and R_i :

$$J(R_{pred}, \mathcal{R}) = \max_{R_i \in \mathcal{R}} \left(\frac{R_{pred} \cap R_i}{R_{pred} \cup R_i} \right) . \quad (\text{A.1})$$

As a rule of thumb, one can consider successful a grasping prediction if J is above 0.25 and the difference between θ_{pred} and θ_{true} is below 30° .

A.2 GraspNet/Trustnet experimental details

This section details the experimental work done on GraspNet/TrustNet. This is based on the work of Laurent Bimont.

A.2.1 Implémentation details

To perform computations, we used a computer with a Nvidia RTX 2080 8Gb graphic card and Intel®8 Core™ i7 9700K 3.6 GHz CPU. Implementation was done in Python 3.6 using Tensorflow

1.13. To train *GraspNet*, we used the adam optimiser with a learning rate set to 0.001. We reduced the learning rate by a factor 0.9 each time the validation loss did not decrease for 5 epochs, and stopped the learning process after 20 epochs without validation loss improvement. Training dataset is based on the Cornell and Jacquard dataset. We performed data augmentation on the Cornell dataset leading to a database of approximately ~ 6000 grasping examples. To implement the different uncertainty metrics τ , we used the following scheme:

- For *MonteCarlo Dropout* and *Concrete Dropout*, we performed $T = 100$ inferences with Dropout layers actives. For *Monte Carlo Concrete Dropout* we used the concrete keras version available on Yarin Gal’s GitHub.
- For *Ensemble Proper Scoring* network, we used adam optimizer with a learning rate of 0.001. We reduced the learning rate by a factor 0.9 after 5 epochs without validation loss improvements.
- For *Simple Ensemble* and *Proper Scoring Ensemble*, we used $T = 5$ networks
- For *Trust Net’s* $\%T$ training, we used a learning rate of 0.001 with adam optimizer. We used learning rate decay of 0.9 after 5 epochs without validation loss improvements. During training, VGG16’s weights were frozen.

Effects of architectural changes on *GraspNet* performance.

Changes on *GraspNet* architecture and training scheme lead to different grasping accuracy summed up in Table A.1. Performances stay at the same level for *Concrete Dropout*, and *Simple Ensemble*. However, the *Proper Scoring* method leads to a drop in performances (-24% for Cornell and -42% for Jacquard). Despite our effort to improve accuracy of this technique, we were not able to find a good set of hyper-parameters to reach an equivalent level of performance. We remark that for the Jacquard dataset, *Proper Scoring* networks had some difficulties to converge. We trained several identical networks leading to heterogeneous accuracy results (ranging from 0.4% to 20.6%). We believe that *Proper Scoring* networks need deeper architectural changes (like changing the number of hidden layers and neurons) to work well. However as we wanted a fair comparison between the different *GraspNet* models, we did not make any change for this study.

Table A.1: Accuracy for various *GraspNet* architectures and learning scheme

<i>GraspNet</i>	Original	<i>Concrete Dropout</i>	<i>Simple Ensemble</i>	<i>Proper Scoring</i>	<i>Ensemble Proper Scoring</i>
Cornell	86.6%	85.7%	87.3%	62.5%	78.8%
Jacquard	62.5%	63.5%	63.1%	20.6%	0.8%

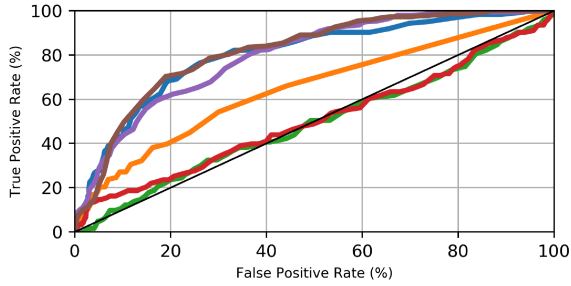
Quantitative analysis

The quantitative analysis is made through the curves and scores proposed in Section 5.3.2. We plot the ROC and $S - \beta\% - FPR$ curves for both the Cornell and the Jacquard dataset in Figure A.1. *Dropout methods* do not perform better than a random detector, since ROC curves follow the random performance and grasping accuracy is not improving with FPR (Figure A.1-(c)-(d)). This poor performance does not seem to be related to fixed Dropout rates p since the results are equivalent for *Concrete Dropout*. Such poor performances of *Dropout methods* have already been highlighted in some papers as mentioned in section 5.2. *Simple Ensemble* has results with ROC curves better than the random detector and an increase in grasping accuracy, however it is outperformed by other methodologies. *Proper Scoring* methods and *TrustNet* show equivalent good ROC curves. However, for the *Proper Scoring* on the Jacquard dataset, the ROC curve is obtained with so few number of good predictions (as shown in Table A.1 and recovered on the S-FPR curve) that it does not allow us to draw a conclusion.

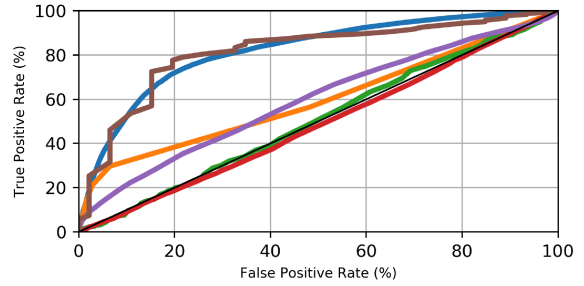
The S-FPR curve of *TrustNet* is much better and relevant for an industrial use case. Indeed for an FPR (\sim economic cost) of 15% we reach a grasping success rate up to 93.7% for the Cornell dataset. We should note that despite starting from a lower grasping success rate, after 20% FPR, *Ensemble Proper Scoring* outperforms other methodologies (except *TrustNet*) for tempted grasp success rate for the Cornell dataset, showing the relevance of *Proper Scoring* as predictive uncertainty.

On Figure A.1-(d), we can notice that *TrustNet* is the only one for which the grasping success at 0%FPR (71.1%) is better than the initial *GraspNet* accuracy (62.5%). This result is particularly interesting as it shows that *TrustNet* can improve grasping accuracy without increasing the number of false alarms.

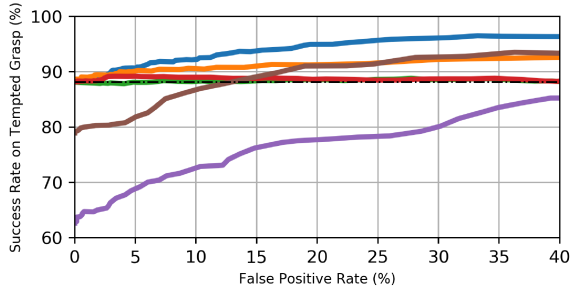
A.2. GRASPNET/TRUSTNET EXPERIMENTAL DETAILS



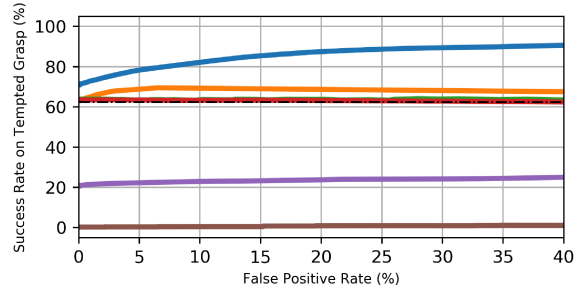
(a) ROC curve for Cornell dataset



(b) ROC curve for Jacquard dataset



(c) S-FPR curve for Cornell dataset



(d) S-FPR curve for Jacquard dataset

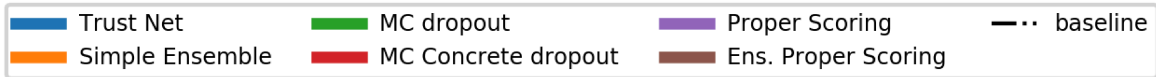


Figure A.1: ROC and S-FPR curves for different uncertainty metrics τ

Table A.2 gathers the different scores defined in chapter 5.2. AUROC scores of *Ensemble Proper Scoring* and *TrustNet* are both very similar ($\sim 80\%$ for Cornell and Jacquard) showing an equivalent relevance between those two metrics τ . For $FPR - 100\% - TPR$, none of the methods achieve relevant results for the Jacquard dataset (100% of FPR is equivalent to asking for help all the time) making them useless for a very cautious use. However, we can spot promising results for *Ensemble Proper Scoring* and *TrustNet* as their value is significantly lower than the baseline 100% for Cornell dataset. Relaxing the security level by 5% ($FPR - 95\% - TPR$) decreases a lot the FPR rate (down to 58.2%) showing that a trade off between security and economic cost may be found. Improving those scores is equivalent to separate the distributions in Figure A.2. We believe that this can be done by continuously learning from an external oracle in an active learning setting.

A.2. GRASPNET/TRUSTNET EXPERIMENTAL DETAILS

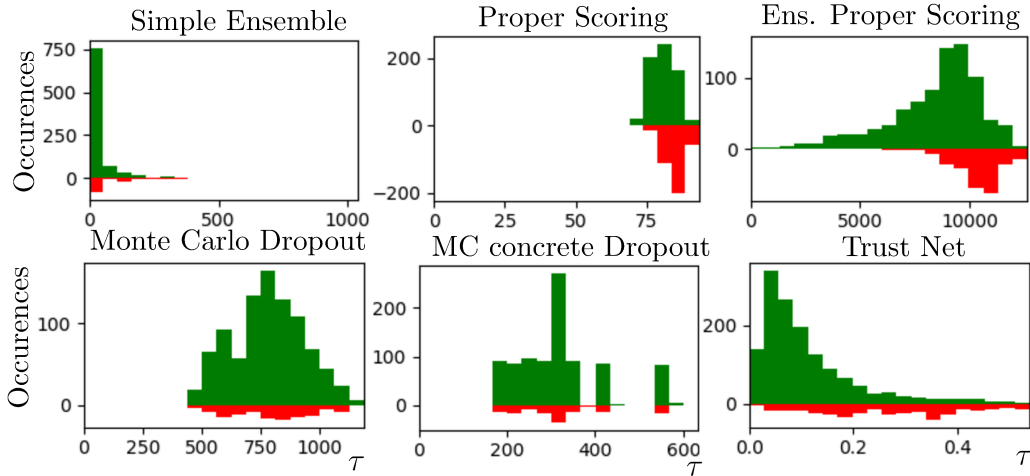


Figure A.2: Histogram of τ for good (green) and bad (red) predictions for different uncertainty methods on Cornell grasping datasets. Bad predictions are represented by negative occurrences

Table A.2: Uncertainty metric performance for grasping cornell|jacquard datasets

Method	AUROC		GS-15%-FPR		FPR-100%-TPR		FPR-95%-TPR	
	corn.	jacq.	corn.	jacq.	corn.	jacq.	corn.	jacq.
<i>Simple Ensemble</i>	64.6%	59.7%	90.1%	68.9%	100.0%	100%	92.4%	94.1%
<i>Proper Scoring</i>	79.5%	59.5%	72.4%	23.2%	97.2%	100%	60.0%	75.3%
<i>Ens. Proper Scoring</i>	81.4%	81.5%	89.0%	0.8%	86.9%	100%	58.2%	72.3%
<i>MC Dropout</i>	52.8%	50.5%	88.8%	63.4%	100%	100%	94.8%	94.7%
<i>MC Concrete Dropout</i>	50.4%	48.6%	89.3%	63.3%	99.9%	100%	95.0%	95.3%
<i>Trust Net</i>	80.0%	82.0%	93.7%	85.4%	94.4%	100%	66.5%	69.9%

A.2.2 TrustNet improvement study

We have shown that, besides outperforming uncertainties metrics baselines, our approach *TrustNet* revealed an interesting potential to separate good predictions distribution from bad one, along uncertainty measure τ . It is a first step toward the goal of reaching a perfect grasps success rate on tempted grasps ($FN = 0$), while limiting the number of irrelevant requests FP . In this section, we experiment different configurations of *TrustNet* to improve its behavior.

Different features representations and Ensemble TrustNet experimentations, done with Cornell dataset, have not led to significant improvements:

- VGG16 with frozen weights remains the most appropriate features extractor for *TrustNet* with an AUROC score of 80.0%, against respectively 74.0%, 73.9%, 56.9% and 55.5% for Xception, MobileNet, Resnet50, and Densenet121.
- We trained an ensemble of $T = 10$ *TrustNet* Networks and computed an uncertainty metric in the same way than for *Deep Ensemble* methods. The AUROC score slightly increases from 80.0% to 81.2%, however inference computation increases by a factor T .

Weighted loss. We also studied the impact of a weighted loss function to tune our failure predictor based on our risk appetite :

$$\mathcal{L}(\tau, \hat{\tau}) = \omega_0 \times \tau \log \hat{\tau} + \omega_1 \times (1 - \tau) \log \hat{\tau} . \quad (\text{A.2})$$

Figure A.3 shows the evolution of FPR-95%TPR score and histogram distributions for different ratios $\gamma = \omega_1/\omega_0$, on Cornell dataset. Those histograms distributions are very different according to the ratio, leading to different behavior of failure predictor. Depending on this ratio, we can set the sensibility of *TrustNet* depending on our risk appetite. We would like to highlight that this fine tuning of the τ metric according to risk appetite of the user is not possible for other tested existing uncertainty methods.

For a small γ around 1, most of the good predictions are concentrated around low τ values while bad predictions have a more uniform distribution. For high γ values (around 100), we highlight an opposite behavior. Therefore, we tried to benefit from this complementary behaviors by considering an ensemble of two *TrustNet* networks respectively trained with $\gamma = 1$ and 100 (we called it *Ensemble Weighted Trustnet*). The resulting uncertainty metric outperforms the original *TrustNet* (Figure A.4) with an AUROC score increasing by 4% (84.1% > 80%). Moreover, FPR-100%-TPR and FPR-95%-TPR scores obtained are 76.5% and 50.4%, which is a major improvement compare to those of the original *TrustNet* (94.4% and 66.5% respectively).

A.2. GRASPNET/TRUSTNET EXPERIMENTAL DETAILS

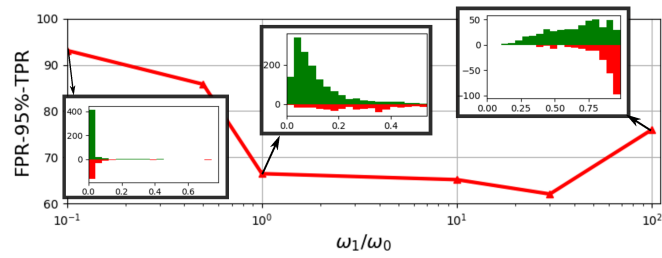


Figure A.3: FPR-95%-TPR evolution for different γ . For some γ , we also plot similar histograms as presented in the qualitative analysis.

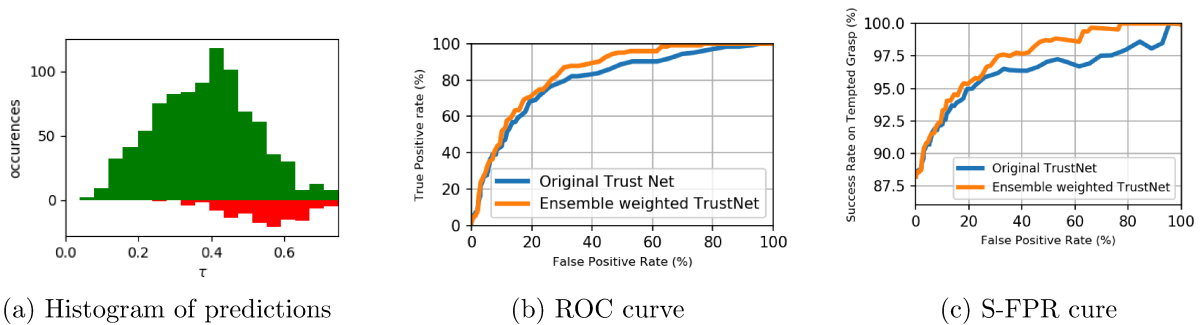


Figure A.4: Performances comparison between *Ensemble Weighted Trustnet* and the original *TrustNet* (Cornell dataset)

Appendix B

Résumé étendu en français

Contents

Chapter 1: Introduction	187
Chapter 2: State of the art on cognitive systems	189
Chapter 3: Design of a cognitive architecture for Industry 4.0	196
Chapter 4: Complementary ML approaches for IRL on planar grasping use cases	198
Chapter 5: Learning under uncertainty	202
Chapter 6 : Implementation and validation on a planar pick and place learning task	206
Chapter 7: Conclusion and perspectives	209
Résumé étendu en français	211

Cette annexe propose un résumé étendu en français du manuscrit de thèse. Les travaux présentés ont été réalisés au sein du LISPEN - Campus de Lille (Laboratoire d'Ingénierie des Systèmes Physiques et Numériques).

B.1 Introduction: vers un Assistant Robotique Intelligent

B.1.1 Motivations

Ces dernières années, les robots industriels ont quitté leur cage pour devenir plus collaboratifs grâce à des robots plus sûrs (plus légers avec des cadences plus lentes, et équipés de capteurs d'effort), à de meilleurs capteurs et à des bibliothèques de programmation de plus haut niveau (voir Figure B.1). Pourtant, dans les scénarios du monde réel, la flexibilité et les capacités d'interaction des robots restent éloignés de l'interaction naturelle attendue entre deux collègues humains. Ce nouveau paradigme nécessite des robots avec de meilleures capacités matérielles et logicielles. Dans ce dernier cas en particulier, l'intelligence artificielle joue un rôle croissant pour faire face à la variabilité de l'environnement et à la complexité des interactions avec les humains.

Dans l'Industrie du Futur idéale, les robots collaboratifs travailleront main dans la main avec les humains et occuperont une place prépondérante centrée sur l'humain. C'est donc au robot de s'adapter

B.1. INTRODUCTION: VERS UN ASSISTANT ROBOTIQUE INTELLIGENT

à la diversité de chaque personne, chaque tâche, chaque situation. Ce sera la génération des assistants robotiques intelligents (SRA pour "Smart Robotic Assistant"). La route est encore longue et nécessite un changement de paradigme et le développement de solutions permettant au robot d'acquérir des capacités d'adaptation d'une manière qui soit à la fois généralisable et explicable.

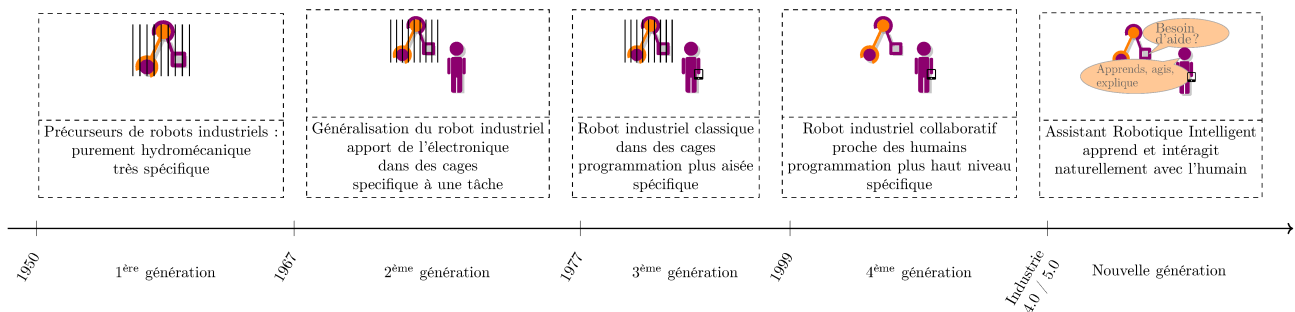


Figure B.1: Evolution des robots industriels.

B.1.2 Positionnement

Ce changement de paradigme nécessite un système cognitif définissant un assistant robotique intelligent SRA [1]. Il doit être capable d'interpréter et de réagir aux interactions naturelles des humains pour un apprentissage incrémental, en particulier au contact d'un instructeur humain non nécessairement expert en robotique et programmation. Il est essentiel que cette interaction soit la plus naturelle et intuitive possible pour l'humain. Cet apprentissage incrémental doit améliorer et exploiter une base de connaissances et de compétences modulaires qui peuvent être utilisées, associées et transférées à un large éventail de tâches, avec une adaptation aux préférences et aux caractéristiques individuelles pour une interaction personnalisée. Ces capacités devraient être intégrées dans un processus de prise de décision rendu aussi explicable que possible pour les non experts en programmation, avec des explications de haut niveau. Un tel SRA aurait un impact important sur la prochaine génération de robots industriels collaboratifs, et est également au cœur d'enjeux sociétaux tels que la réduction des troubles musculosquelettiques (TMS) et l'handicap au travail.

Pour gérer une telle complexité de manière significative et compréhensible, nous devons mettre en oeuvre un agent artificiel en tirant parti de plusieurs paradigmes d'intelligence artificielle et d'interaction homme-robot.

Une mesure de l'intelligence peut être considérée comme la capacité d'apprendre comment accomplir des tâches et de tirer parti de ce qui a été appris pour transfert à de nouvelles tâches cibles. Une autre propriété intéressante de l'intelligence humaine est que les connaissances sont construites de manière incrémentale tout au long de la vie. La généralisation et l'adaptation pourraient prendre racine dans cette capacité d'apprentissage continu de manière supervisée ou autonome et dans l'échange d'informations avec les autres. Cela nécessite aussi un certain niveau de connaissances préalables et de sens commun.

Bien que l'IA de niveau humain soit probablement encore loin d'être atteinte, cela encourage le développement d'agents qui apprennent tout au long de leur vie, au fil de temps, de manière incrémentale et interactive comme le font les humains depuis leur enfance. Pour le développement et

la mise en œuvre de tels agents robotiques, la conception d'architectures robotiques cognitives est un point central.

B.1.3 Objectifs

Globalement, la conception d'un SRA nécessite une approche multidisciplinaire. Dans cette thèse, nous avons visé la conception d'un prototype de base d'architecture cognitive dans le contexte de l'Industrie 4.0. Nous avons fixé les spécifications à remplir par un SRA dans un contexte industriel :

- **Perception et adaptation robuste, flexibilité.** Un SRA doit pouvoir percevoir, interpréter, et réagir en temps réel à la complexité d'un environnement dynamique, de manière flexible et robuste. Cette complexité inclut bien entendu les interactions humaines. Des modules à base d'apprentissage automatique seront intégrés pour l'adaptation à la variabilité des situations ne pouvant être traitée de manière discrète et combinatoire.
- **Interaction naturelle, intuitive, et personnalisée. Reconfigurabilité aisée.** Un SRA doit interagir intuitivement avec des non-experts (non-programmeurs et non experts en robotique). Il doit donc comprendre spécifiquement les moyens de communication naturels des humains tels que la vision, la parole, le regard, le toucher, mais aussi fournir durant l'interaction des réponses/demandes compréhensibles par des non-experts et non ambiguës. Un SRA doit également s'adapter aux préférences et spécificités des utilisateurs tels que la main dominante, un handicap. Pendant que l'agent SRA apprend de nouvelles tâches, il doit être capable d'adapter avec un certain niveau d'automatisation ses comportements en fonction des préférences et des caractéristiques de chaque individu, pour une interaction personnalisée. L'acceptabilité sera d'autant facilitée et des utilisateurs non experts pourront intuitivement reconfigurer le système à leur guise pour une autre tâche.
- **Apprentissage incrémental modulaire.** Un SRA doit avoir la capacité d'apprendre rapidement et de manière incrémentale une nouvelle tâche, du bas niveau aux abstractions de haut niveau. L'exécution d'une tâche requiert à la fois des connaissances de haut niveau pour la compréhension générale et de bas niveau pour la perception et l'exécution dans le monde réel. Cela peut se faire par transfert de connaissances et nécessite des représentations et des processus qui favorisent la modularité dans tout le système. Le SRA pourra ainsi exploiter une base de connaissances antérieures pour l'exécution des tâches et l'apprentissage en ligne. Nous ne voulons pas tout apprendre à partir de zéro à un robot. Par conséquent, un agent SRA doit être capable de tirer parti de certaines connaissances antérieures tout en exécutant et en apprenant des compétences modulaires pour résoudre des tâches.
- **Comportements explicables et sûrs.** Un SRA doit être capable de raisonner et d'avoir des capacités d'explication aux moins partielles, pour fournir des explications sur ses prédictions et ses comportements. Un comportement erroné peut générer des problèmes de sécurité dans un contexte industriel, pouvant porter intégrité à des objets, à l'agent robotique lui-même, voir plus grave aux opérateurs. Pour avoir un comportement sûr, le SRA doit savoir gérer l'incertitude dans la moto-perception, ses représentations internes, ses prédictions. Nous voulons que l'agent SRA sache ce qu'il ne sait pas. Pour cela, la notion d'incertitude est importante. En tant que mesure de confiance et de curiosité, elle donne à l'agent SRA la capacité de raisonner sur ses

propres prédictions afin de décider d’agir ou de ne pas agir (questionner). L’agent devient alors acteur de son propre apprentissage, et de plus en plus expert.

B.1.4 Contributions et organisation

Cette thèse a pour but de développer et d’intégrer les briques principales afin de créer une architecture robotique cognitive pour la robotique collaborative permettant d’intégrer l’ensemble des spécifications requises décrites en section B.1.3. Dans ce résumé, la section B.2 (correspondant au chapitre 2 et 3) détaille l’architecture actuelle que nous avons proposée, en termes de blocs principaux, de processus d’apprentissage interactif et d’organisation des modules, dans une vue d’ensemble de haut niveau et au regard de l’état de l’art. Nous détaillons ensuite dans la section B.3 (chapitre 4), les différents paradigmes d’apprentissage qui ont été étudiés au cours de la thèse. L’apprentissage d’une préhension orientée avec des robots réels a été utilisé comme application pour une validation de l’intégration des spécifications. Plus précisément, une contribution à l’apprentissage de la préhension orientée par démonstration a été apportée par le développement d’un module spécifique. La section B.4 (chapitre 6) décrit la mise en oeuvre de l’architecture ainsi que des modules intégrés en validant l’approche globale sur un robot réel. Enfin, la section B.5 (chapitre 5 et 7) présente les perspectives de développement. L’importance de la gestion des incertitudes y est particulièrement abordée.

Les travaux réalisés au cours de cette thèse ont été valorisés par des publications et soumissions internationales et la réalisation de vidéos de présentations : 1¹, 2², 3³, déposées sur la chaîne youtube du laboratoire LISPEN⁴.

- Contribution à l’apprentissage par démonstration et à la préhension orientée : [2]
- Contribution sur l’architecture cognitive pour l’apprentissage interactif pour les robots collaboratifs industriels : [3]
- Un article de revue internationale regroupant et mettant à jour les contributions avec les derniers développements et validations de notre architecture : soumis, processus de révision en cours.

¹<https://www.youtube.com/watch?v=T592ye7RPxQ>

²<https://www.youtube.com/watch?v=1UjehV1RCLc>

³<https://www.youtube.com/watch?v=EAuLMnQULB0>

⁴<https://www.youtube.com/channel/UC-PsC0eg4uLbF35vW-Sa7zw>

B.2 Conception d'une architecture cognitive inspirée de l'état de l'art

Cette section décrit les principaux composants nécessaires à la construction d'une architecture cognitive [4] vis à vis de l'état de l'art. L'accent a été mis sur les différents compromis entre la vision connexionniste et la vision symbolique en intelligence artificielle afin de justifier le développement d'une architecture hybride pour l'apprentissage de tâches. Les architectures existantes n'étant pas totalement adaptées à nos spécifications, nous avons développé notre propre architecture. En s'inspirant de l'état de l'art en matière de SRA pour l'apprentissage interactif de tâche, d'architecture cognitive et des avancées récentes dans les architectures d'apprentissage profond, plusieurs choix de conception ont été faits pour construire un prototype d'architecture cognitive hybride permettant d'intégrer l'ensemble de nos spécifications.

La Figure B.2 fournit une vue d'ensemble de haut niveau de l'architecture hybride. Elle présente les différentes représentations et la manière dont elles interagissent afin de construire des comportements complexes. Pour plus d'informations sur les différents blocs et leur interaction, le lecteur peut se référer à 3.

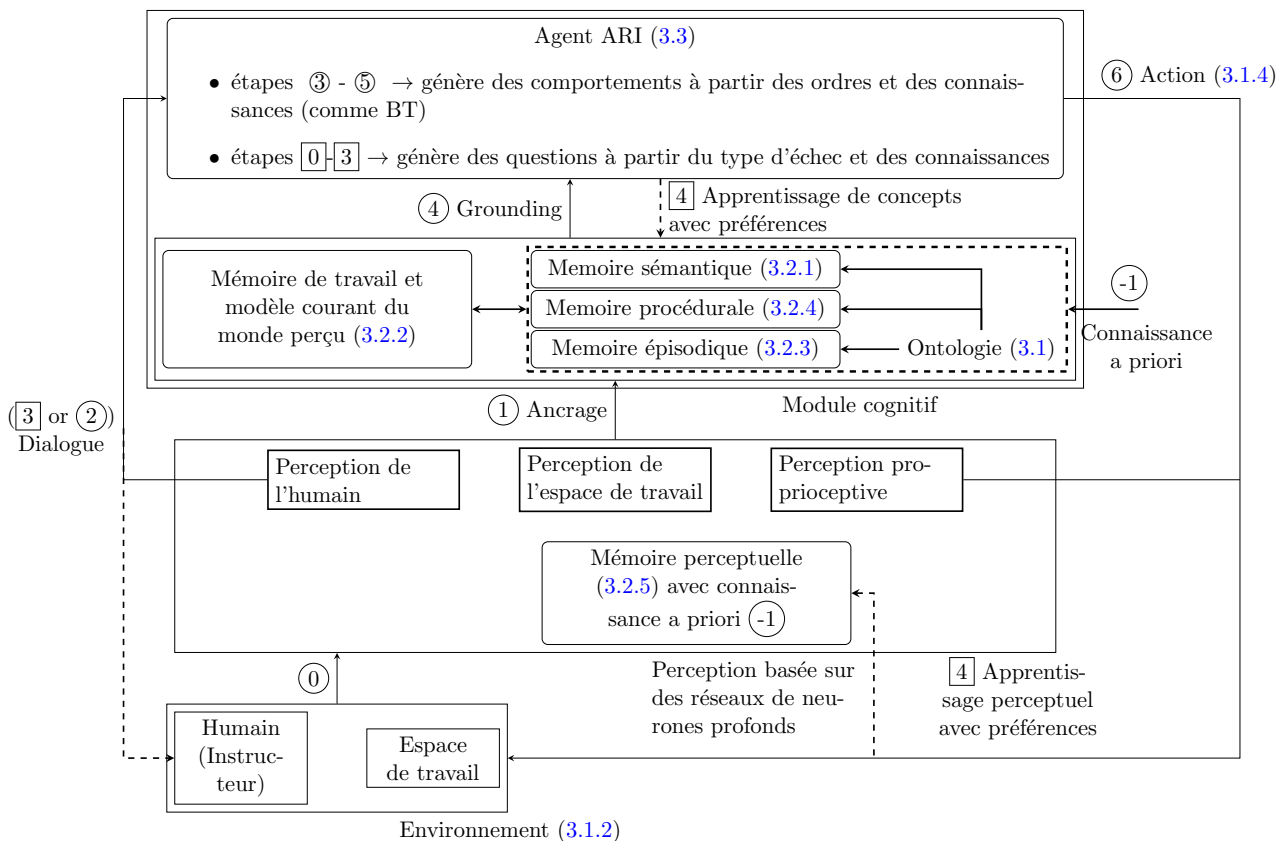


Figure B.2: Aperçu de haut niveau de l'architecture. L'architecture se compose de modules perceptifs basés sur des approches connexionnistes, de représentations relationnelles symboliques et d'un processus délibératif pour l'apprentissage en interaction avec l'humain.

B.2.1 Approches connexionnistes et symboliques

Les systèmes d'IA peuvent être abordés de deux points de vue : un point de vue descendant où l'on part d'une connaissance abstraite, relationnelle et de haut niveau, où la plupart des raisonnements et des planifications ont lieu (parfois appelé système 2 dans la littérature), jusqu'aux capacités sensori-motrices de l'agent. C'est généralement le territoire de l'IA symbolique qui exploite des symboles pour ses représentations internes. À l'opposé, une vision ascendante est liée à l'IA connexionniste et vise à exploiter l'interaction de plusieurs modèles simples à partir desquels des comportements complexes émergent (au niveau du système 1). Les deux approches ont leurs avantages et leurs inconvénients pour la construction d'une architecture cognitive pour la robotique. Nous illustrons, Figure B.3, l'existence d'un compromis, en termes de facilité d'implémentation et de représentations, entre les architectures symboliques et connexionnistes en ce qui concerne les représentations abstraites, l'efficacité des données pour l'apprentissage et l'explicabilité du système. Cela à justifier une approche hybride au sein de notre architecture.

En particulier, l'approche symbolique sera utilisée pour permettre une représentation modulaire et explicable de la connaissance, mais combinatoire donc rigide. La flexibilité nécessaire pour l'adaptation à la variabilité/complexité d'un environnement dynamique sera apportée grâce à l'approche connexionniste (en particulier des réseaux de neurones) au détriment d'un manque d'interprétabilité. C'est pourquoi l'intégration de l'incertitude au regard des prédictions des modules sera d'autant plus primordiale.

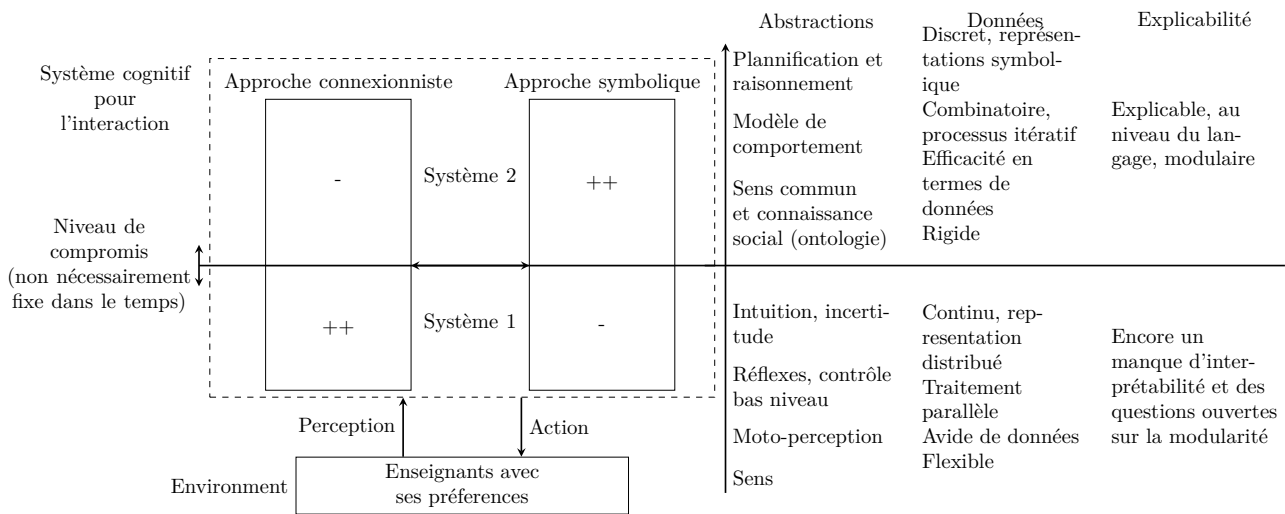


Figure B.3: Conception générale de l'architecture hybride

Les principaux éléments constitutifs d'une architecture cognitive reposent sur une ontologie et un modèle de comportement. Une ontologie joue le rôle d'une structure explicable pour l'interopérabilité des composants de l'architecture [5]. C'est une représentation conceptuelle orientée objet construite autour de classes, d'attributs/de propriétés et de relations entre ces concepts. Les systèmes cognitifs sont souvent accompagnés d'une ontologie qui fournit une structure symbolique de base facilitant la compatibilité entre les différents modules et sous-systèmes, ou même entre différents systèmes indépendants (tels que d'autres robots) [6]. Les robots étant des agents agissant, nous avons également besoin d'un modèle de comportement capable d'exploiter l'ontologie. La figure B.4 illustre une ontologie sim-

B.2. CONCEPTION D'UNE ARCHITECTURE COGNITIVE INSPIRÉE DE L'ÉTAT DE L'ART

ple. Des ontologies plus complexes existent telles que KnowRob [7] utilisée dans l'architecture CRAM [8].

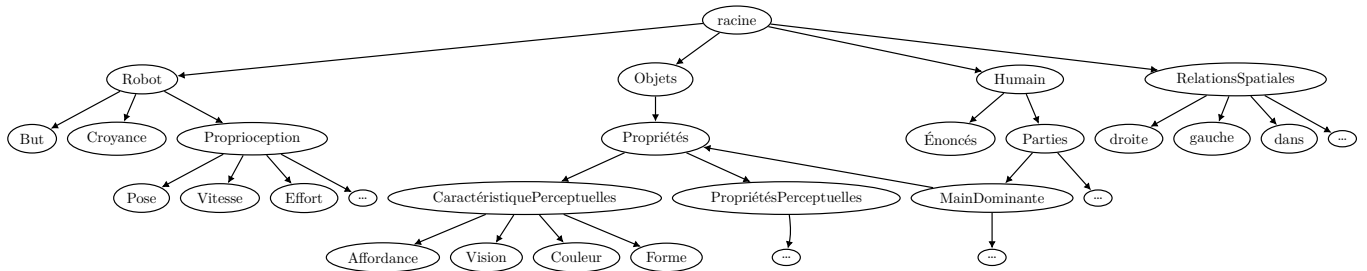


Figure B.4: Exemple haut niveau d'une ontologie

Compte tenu de notre ontologie, nous exploitons différents types de mémoires, détaillées dans la section 3.2, et utiles pour différents aspects de l'apprentissage des tâches : une mémoire sémantique, une mémoire de travail, une mémoire épisodique, une mémoire procédurale, et une mémoire perceptuelle.

Ces représentations sont sous formes de graphes relationnels pour leur caractère hiérarchique, modulaire, leur explicabilité et pour faciliter l'apprentissage par rapport à nos spécifications.

Les composants connexionnistes d'apprentissage peuvent ensuite être exploités pour ancrer ces représentations symboliques dans le réel via une collection de données d'apprentissage. L'agent SRA peut alors apprendre les propriétés des objets telles que les caractéristiques visuelles et l'affordance (comment prendre un objet pour une tâche donnée), compte tenu du contexte de la tâche, des préférences et caractéristiques humaines. Un exemple d'intégration technique de tels modules connexionnistes est donné dans la section de validation expérimentale (chapitre B.4).

B.2.2 Choix du modèle de comportements

Les robots étant des agents agissant, nous avons spécifiquement besoin d'un modèle de comportement capable d'exploiter l'ontologie. Pour agir dans le monde réel, un agent SRA doit être capable de générer des comportements complexes pertinents pour une situation donnée, même nouvelle. Pour cela, il doit être capable de planifier et de réagir tout en apprenant en ligne auprès de l'humain.

On distingue en générale dans les architectures trois couches principales [9, 10] (voir Figure B.5) : une couche fonctionnelle adaptée à l'action, à la perception et à l'apprentissage ; une couche de décision adaptée à la planification ou à la supervision ; et une couche d'exécution où le modèle de comportement intervient, pour interfacer et coordonner les autres couches du système en fonction des exigences de la tâche en cours.

B.2. CONCEPTION D'UNE ARCHITECTURE COGNITIVE INSPIRÉE DE L'ÉTAT DE L'ART

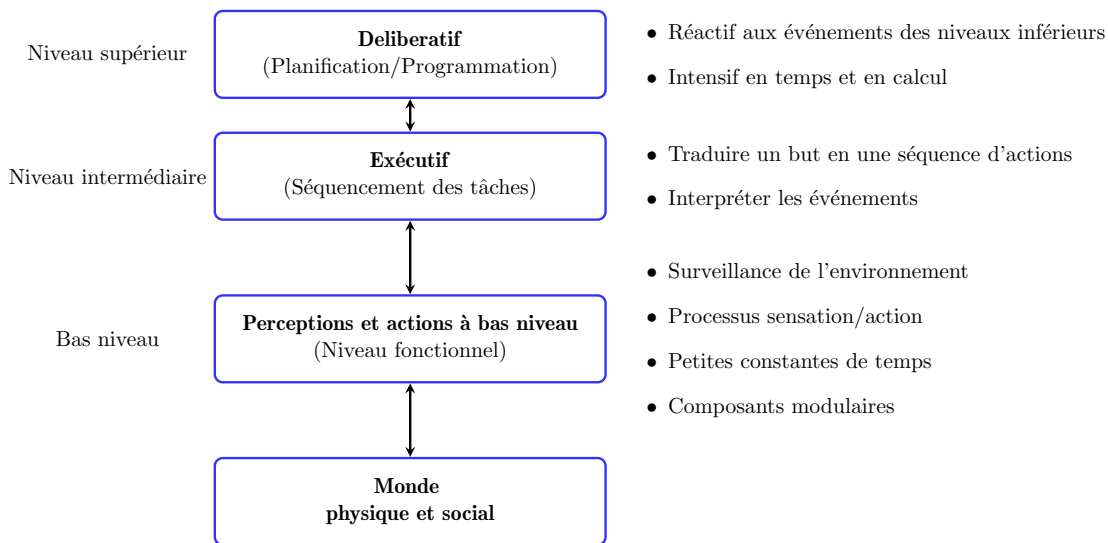


Figure B.5: L'architecture robotique cognitive peut être représentée par trois couches de contrôle.

Au niveau d'un Assistant Robotique Intelligent, le modèle de comportement doit pouvoir répondre à un certain nombre d'exigences pour obtenir un robot interactif. Le modèle de comportement doit :

- être explicable. Le comportement du robot doit être au moins partiellement compréhensible. Par conséquent, le comportement doit être suffisamment complexe pour permettre à l'agent d'exécuter des tâches, mais aussi suffisamment simple et interprétable pour être compris par des utilisateurs non experts.
- décrire à la fois les parties descriptives et exécutives des comportements. L'agent SRA doit notamment être capable de décrire le *quoi*, le *pourquoi* et le *comment* de ses actions.
- s'interfacer avec des bases de données de connaissances préalables (telles que les préférences des utilisateurs). Le modèle de comportement doit permettre l'intégration de connaissances préalables de différents types, comme les connaissances basées sur des règles, les spécificités et les préférences des utilisateurs. Par exemple, il peut s'agir de la main dominante, ou d'un handicap nécessitant une adaptation de l'agent à l'opérateur.
- être interopérable avec le modèle du monde. Lorsque l'agent construit un modèle du monde, le modèle de comportement doit être capable de l'utiliser en respectant l'ontologie.
- interfacer les compétences de bas niveau et de haut niveau de manière multimodale. Les compétences d'apprentissage nécessitent à la fois des informations de haut et de bas niveau et le modèle de comportement sert donc de passerelle entre les deux.
- s'interfacer avec des techniques d'apprentissage telles que les réseaux neuronaux profonds. Les techniques d'apprentissage profond étant devenues des outils d'apprentissage très puissants, le modèle comportemental doit s'interfacer facilement avec ces systèmes.

- permettre un apprentissage rapide et une forte généralisation grâce à la réutilisation de compétences avec des représentations composables et paramétrables. Dans le cadre de l'enseignement interactif de tâche, l'apprentissage se fait en ligne en interagissant avec un enseignant humain. Par conséquent, nous voulons que l'apprentissage soit rapide tout en conservant de bonnes capacités de généralisation. Ceci peut être fait en exploitant des comportements modulaires paramétrés. Cela permet, en effet, de réutiliser un comportement appris dans plusieurs tâches connexes avec un minimum de mises à jour.
- permettre de raffiner les actions de manière réactive. Comme l'environnement est dynamique et peut changer en fonction des actions du SRA ou d'autres agents, le robot doit alterner en permanence entre la perception provenant des flux sensoriels, l'apprentissage, la planification et l'action dans une boucle délibérative prenant en compte l'humain.

Afin de choisir le paradigme à mettre en oeuvre, nous avons passé en revue la littérature sur les architectures pour l'enseignement interactif de tâches et les modèles de comportement qu'ils utilisaient. Nous avons basé notre comparaison en mettant à jour l'état de l'art dans [11] (partie 2.) par rapport à nos besoins. Les différents modèles sont détaillés dans le chapitre 2 de la thèse. Dans notre architecture, nous avons choisi les arbres de comportement (BTs pour Behavior Trees) comme modèle de comportement. Les BTs sont des modèles basés sur des arbres qui permettent une séparation claire entre la structure de l'arbre (la partie descriptive en tant que flux de contrôle des comportements) et l'implémentation des noeuds (la partie exécutive). Ils sont largement utilisés dans l'industrie des jeux vidéo au lieu d'autres modèles que les machines à états qui sont enclins à l'explosion d'états lorsque les comportements deviennent complexes. L'utilisation de noeuds parallèles facilite également l'exécution de processus parallèles, comme cela est nécessaire dans un cadre d'une interaction multimodale. La gestion des échecs est aisée et est au coeur du processus d'apprentissage dans notre architecture. La nature hiérarchique des BTs facilite l'implémentation de méthodes de raffinement : étant donné des actions de haut niveau, il est possible, en fonction des changements de l'environnement, de ramifier l'arbre vers des sous-actions de plus bas niveau. Ces propriétés attrayantes en termes de modularité du comportement en font une alternative pertinente à d'autres modèles de comportements. De plus, des sous-arbres peuvent être ajoutés ou supprimés n'importe où dans le BT sans modifier les autres composants ce qui augmente la flexibilité de ces modèles. Enfin, il est possible d'étendre les BTs standards avec des noeuds de préconditions et postconditions [12], ce qui aide à construire des représentations pour la planification.

Plus précisément, les arbres de comportement (BT) sont composés de plusieurs (généralement six) types de noeuds illustrés dans le tableau B.1 : un ensemble de noeuds de *contrôle* qui aide à gérer le flux de décision, un ensemble de noeuds d'*exécution* qui exécute les actions, un noeud le *décorateur* qui aide à construire des noeuds de contrôle plus complexes, comme réessayer une action ou un sous-arbre jusqu'au succès. Chaque noeud peut renvoyer un état, généralement *succès* ou *échec*. Les noeuds de contrôle renvoient *succès* ou *échec* en fonction du statut de retour de leurs enfants et des règles définies dans le tableau B.1.

B.2. CONCEPTION D'UNE ARCHITECTURE COGNITIVE INSPIRÉE DE L'ÉTAT DE L'ART

Table B.1: Flux de contrôle et noeuds d'exécution dans le cadre standard des arbres de comportement

	Noeuds d'exécution	Symbole	Succès	Échec
	Action	□	L'exécution est effectuée	Exception pendant l'exécution
	Condition	○	Condition est vraie	Condition est fausse
	Noeuds de contrôle			
	Séquence	→	Tous les enfants doivent réussir	Un enfant échoue
	Parallèle	→	Plus de $M \in \mathbb{N}^*$ enfants réussissent	Plus de $N \in \mathbb{N}^*$ enfants échouent
	Repli	?	Un seul enfant réussit	Tous les enfants échouent
Décorateur	◇	Défini par l'utilisateur	Défini par l'utilisateur	Défini par l'utilisateur

Afin de définir des comportements modulaires complexes, compatibles avec des objectifs de planification et de raisonnement, nous représentons les compétences avec des BT en utilisant le modèle traditionnel préconditions, exécution, postconditions (également appelé effets dans la littérature) (voir Figure B.6). Dans [13], les auteurs fournissent un aperçu formel détaillé des BT et de leur utilisation en robotique. Les postconditions permettent de vérifier si l'exécution de la compétence est un succès, tandis que les préconditions déterminent si l'agent possède les connaissances nécessaires pour exécuter la compétence, telles que par exemple l'existence de préférences dans l'exécution d'une tâche. Une autre propriété intéressante de la modularisation des compétences avec des conditions est le fait qu'elle aide l'agent à faire de la perception active : l'agent ne vérifie que les conditions qui sont pertinentes pour la tâche, selon les compétences précédemment apprises.

Une *action primitive* est une feuille dans le modèle de comportement global. Par conséquent, elle est directement exécutée sans raffinement supplémentaire par le robot. Des exemples d'actions primitives sont l'ouverture ou la fermeture d'une pince, un mouvement point à point, l'envoi de questions à l'humain.

Plus de détails sont disponibles dans la section 3.1.4 du manuscrit et dans [13] pour les aspects théoriques.

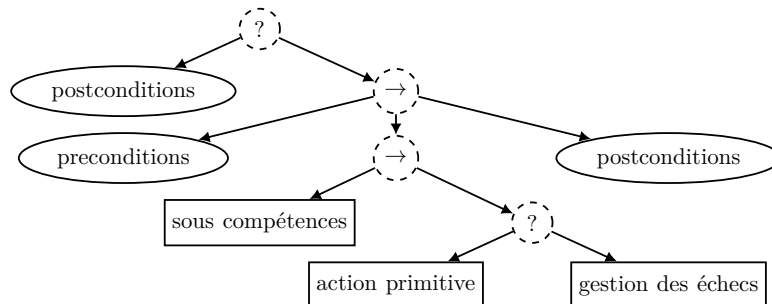


Figure B.6: Modèle de base pour une compétence

B.2.3 Processus décisionnel du SRA prenant en compte les préférences

Dans notre architecture, nous exploitons spécifiquement les mécanismes d'échec des BT comme un signal de haut niveau à des fins d'apprentissage. Cela permet d'affiner ou d'étendre l'arbre de manière incrémentale pendant l'interaction du SRA, de manière similaire à [14]. La transparence, la modularité et l'efficacité des BT apparaissent naturellement puisque la réutilisation, la mise à jour et la composition des comportements peuvent être effectuées en exploitant les graphes. Le processus d'apprentissage interactif est basé sur la gestion des erreurs (*échecs* ou *impasses*) pendant le flux d'exécution du programme. La figure B.7 illustre plus spécifiquement le processus décisionnel qui se produit pendant l'exécution d'une compétence. Ici, nous nous concentrons principalement sur le processus d'interaction et ses aspects délibératifs. En reprenant la figure B.2, on peut observer qu'il y a différents chemins. Le chemin en trait plein représente ce qui se passe lorsque l'agent a toutes les connaissances pour agir, l'ordre des étapes est représenté par des chiffres encadrés. Le chemin en pointillé représente ce qui se passe lorsqu'un échec se produit, les étapes sont représentées par des nombres encadrés. L'exécution entrelacée de ces deux chemins est au coeur du cycle d'interaction, au cours de laquelle l'agent robotique agit selon les instructions de l'humain ou apprend de ses échecs à la fois au niveau symbolique et connexionniste.

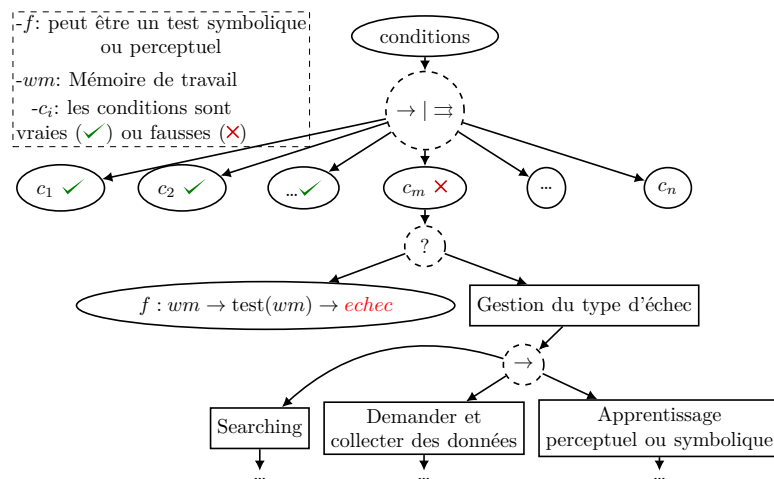


Figure B.7: Le processus de résolution des échecs déclenche l'apprentissage de représentations symboliques ou perceptuelles grâce à l'interaction avec l'humain.

B.2.4 Conclusion

L'exploitation des architectures existantes était difficile, car la plupart des SRA étaient soit spécifiques à une plateforme, soit non open source, soit non maintenus. De plus, la plupart des agents ne pouvaient pas valider toutes nos spécifications dans une même et unique approche. Cela a motivé l'exploration du développement de notre propre architecture cognitive hybride pour nos cas d'utilisation industrielle collaborative. Finalement, notre principal objectif et contribution est d'intégrer plusieurs idées complémentaires de la littérature dans une seule architecture, pour construire un SRA présentant des capacités d'apprentissage interactif et incrémental, non seulement à haut niveau mais aussi à bas niveau, avec des modules d'apprentissage profond, tout en étant capable d'être rapidement reconfig-

uré selon les préférences humaines. Cela a conduit à des représentations symboliques de haut niveau, sémantiquement compréhensible, pour la gestion de la complexité et une meilleure explicabilité du comportement du système. Le processus délibératif du SRA exploite ces représentations pour piloter des modules d'apprentissage connexionnistes, par le biais d'un processus interactif humain/robot axé sur des objectifs et le dialogue. Le chapitre suivant (chapitre B.3) se concentre sur les approches complémentaires d'apprentissage automatique qui ont été utilisées au cours de la thèse afin de développer des modules connexionnistes exploitable dans le contexte de l'enseignement interactif de tâches.

B.3 Approche ML pour la préhension planaire

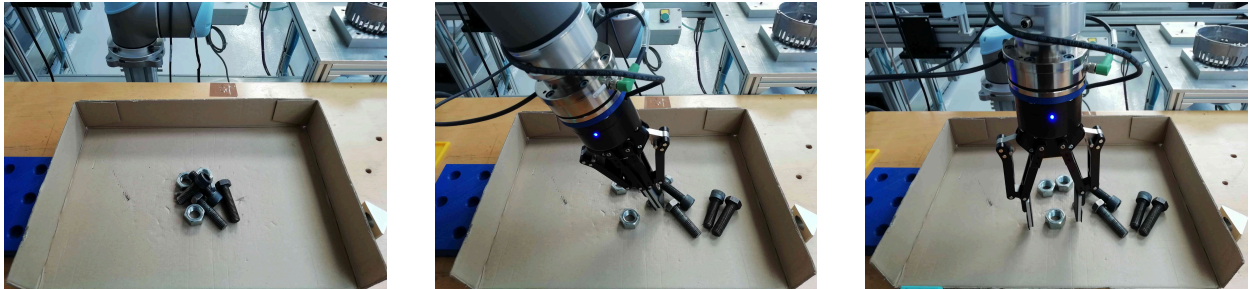
Étant donnée que le SRA apprend de manière interactive une tâche et la structure des compétences associées, nous avons vu qu'il doit fonder ses représentations en apprenant à partir de données du monde réel. Nous détaillons à cette fin la manière dont l'agent peut exploiter des paradigmes complémentaires d'apprentissage machine (ML) dans une approche connexionniste. En suivant nos spécifications, nous voulons exploiter des modules qui permettent un apprentissage rapide en ligne, à partir de jeux de données construits à la volée, pendant l'interaction. En raison de l'importance des tâches liées à la préhension dans de nombreuses applications industrielles, nous nous sommes concentrés sur des tâches liées à la préhension orientée dans le plan. Nous présentons les résultats principaux pour deux modules d'apprentissage adaptés à l'acquisition de compétences liées à la préhension. Le premier s'appuie sur une approche d'apprentissage profond par renforcement, adaptée de [15], pour l'apprentissage autonome d'une tâche de devracage. La deuxième et plus importante contribution dans le cadre de la thèse, présente un module d'apprentissage d'affordance de préhension orientée à la tâche, à partir de quelques démonstrations humaines, respectant nos spécifications d'SRA. Les modules ont été développés en collaboration avec un co-auteur, Laurent Bimont.

B.3.1 Apprentissage autonome du devracage

B.3.1-I Module de devracage: Le cas du devracage est un exemple typique de tâche pour laquelle il est difficile d'expliquer de manière procédurale comment l'exécuter, même pour un humain. En effet, nous pourrions difficilement expliquer pourquoi nous répartissons le tas d'une certaine manière plutôt que d'une autre et l'ordre des actions effectuées.

L'apprentissage par renforcement convient bien pour apprendre une telle tâche de manière autonome. Par conséquent, nous avons reproduit [15] et l'avons étendue expérimentalement à notre contexte industriel. Nous avons abordé les problèmes de devracage comme une stratégie d'apprentissage par renforcement autonome où l'agent robotique apprend les synergies entre poussée et préhension, comme illustré dans la figure B.8. En outre, l'utilisation de l'apprentissage par renforcement et de l'apprentissage profond permet au robot d'apprendre à prendre des pièces sans avoir besoin d'un modèle CAO. Cet aspect est important car on peut s'attendre à ce que les robots collaboratifs travaillent avec des pièces qui n'ont pas été modélisées par des spécialistes de la CAO, notamment dans l'industrie à petite échelle.

B.3. APPROCHE ML POUR LA PRÉHENSION PLANAIRE



(a) Tas de pièces

(b) Actions de poussée pour séparer des pièces

(c) Préhension de pièce isolée

Figure B.8: Exemple de synergie entre actions de poussée et de préhension. On présente une pile d'objets dont aucun ne peut être récupéré par préhension directe (a). Le robot va d'abord pousser la pile pour séparer des objets (b) afin de saisir un premier objet isolé (c).

Ce travail a été valorisé par une démonstration lors de la journée de clôture du projet européen Col-Robot ⁵ en présence de membres de la Commission européenne, de différents partenaires académiques et de partenaires industriels (Renault et Thalès). Une vidéo de ce travail peut être trouvée [ici](#)⁶.

Le dispositif expérimental était le suivant (Figure B.9) : nous avons installé un capteur de profondeur haute définition de qualité industrielle (une caméra photoneo3D ⁷) au sommet d'un robot collaboratif UR5 équipé d'une pince deux doigts Robotiq. Pour la validation, nous avons collecté des vis et des boulons pour l'opération de devracage et avons réalisé l'expérience sur une plateforme industrielle.

⁵<https://colrobot.eu/>

⁶<https://www.youtube.com/watch?v=T592ye7RPxQ>

⁷<https://www.photoneo.com/products/phoxi-scan-1/>

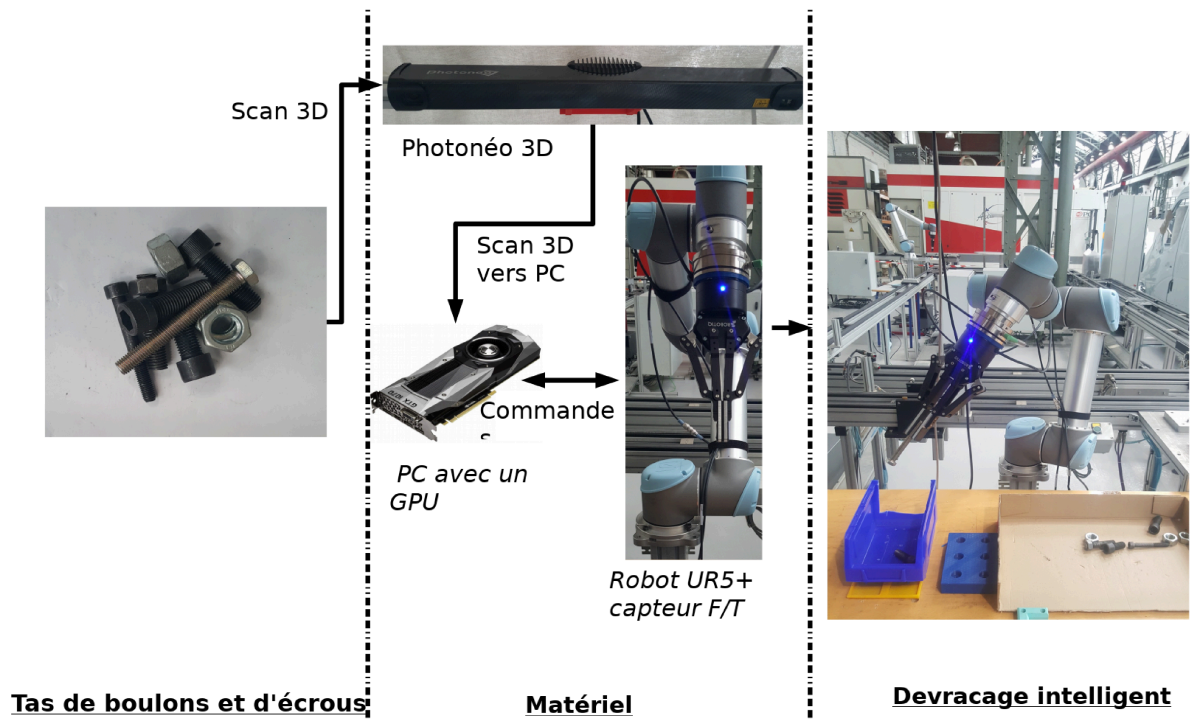


Figure B.9: Dispositif expérimental

B.3.1-II Méthodologie et résultats: Nous avons reproduit l'architecture du réseau en implémentant un algorithme DQN (Deep Q network), illustré Figure B.10. L'algorithme DQN permet d'exploiter des réseaux neuronaux pour inférer des Q-maps (voir les sorties sur la Figure B.10) à partir de l'état de l'espace de travail. Une Q-map associe à chaque pixel de l'image d'entrée une valeur prédisant la récompense attendue de l'action (respectivement poussée et préhension) à l'endroit considéré, compte tenu des récompenses des actions précédentes. L'action décidée est alors celle qui maximise la récompense attendue. A chaque action, le système a été récompensé de la manière suivante :

- pour une action de poussée, la récompense attribuée dans $]0,1[$ est fonction de la différence de dispersion des pièces entre le nouvel état et l'état précédent (modification apportée par rapport à l'article [15]).
- pour une action de préhension la récompense est de type "tout ou rien" à savoir 1 si la préhension a réussi et 0 si elle a échoué.

En effet, une action de poussée est considérée comme bonne si elle augmente la dispersion entre les pièces. Par contre, la récompense d'une telle action doit rester inférieure à celle d'une bonne action de préhension pour privilégier une action de préhension à une action de poussée, sinon le système pourrait développer la stratégie de pousser indéfiniment.

Après quelques heures d'apprentissage par renforcement, le robot a acquis la capacité de prendre et de ranger un tas de vis et de boulons sans aucun modèle CAO des pièces.

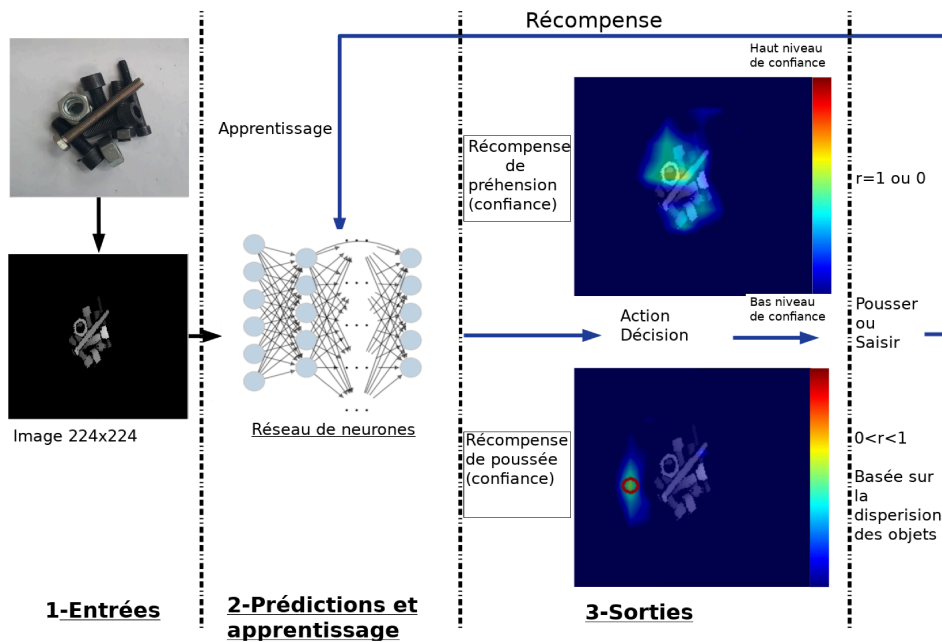


Figure B.10: Architecture de l'algorithme

B.3.1-III Conclusion du module: Au final, l'agent robotique a été capable d'améliorer ses performances au fil du temps et est capable de s'adapter de manière autonome à des objets qu'il n'a jamais vus. Cependant, il existe certaines limites. Dans un contexte industriel, les objets peuvent avoir des spécifications et doivent être capturés de manière très spécifique en fonction de l'objet et de la tâche à accomplir (préhension orientée). Cela a motivé le fait que ce type d'apprentissage seul n'est pas suffisant. Nous avons en outre besoin de plus d'interaction entre l'opérateur et le robot pendant le processus d'apprentissage, afin que le robot puisse apprendre à s'adapter aux besoins spécifiques de la tâche et des opérateurs, en gardant toujours à l'esprit nos spécifications. Cela a motivé le développement d'un deuxième module (prochaine section B.3.2) qui a été intégré à l'architecture plus globale dans le cadre de sa validation.

B.3.2 Apprendre l'affordance de la position de saisie à partir de quelques démonstrations

Avec ce module, nous avons étudié le problème d'un opérateur qui souhaite configurer un robot pour saisir un objet industriel par une zone spécifique. Notre motivation a été de créer un système d'apprentissage rapide de la préhension et qui ne nécessite pas de bases de données, de modèles CAO ou de simulateurs. Ce système doit être simple et intuitif pour pouvoir être facilement reconfiguré par l'opérateur lui-même sans expertise en programmation ou robotique. Le transfert de connaissances de l'opérateur vers le robot se fait par des interactions naturelles : la démonstration à la main des emplacements de préhension autorisés et interdits (Figure B.11).

Ce travail a été valorisé dans l'article de conférence suivant [2]. Une vidéo synthétique de présentation peut être trouvée [ici](#)⁸.

⁸<https://www.youtube.com/watch?v=1UjehV1RCLc>

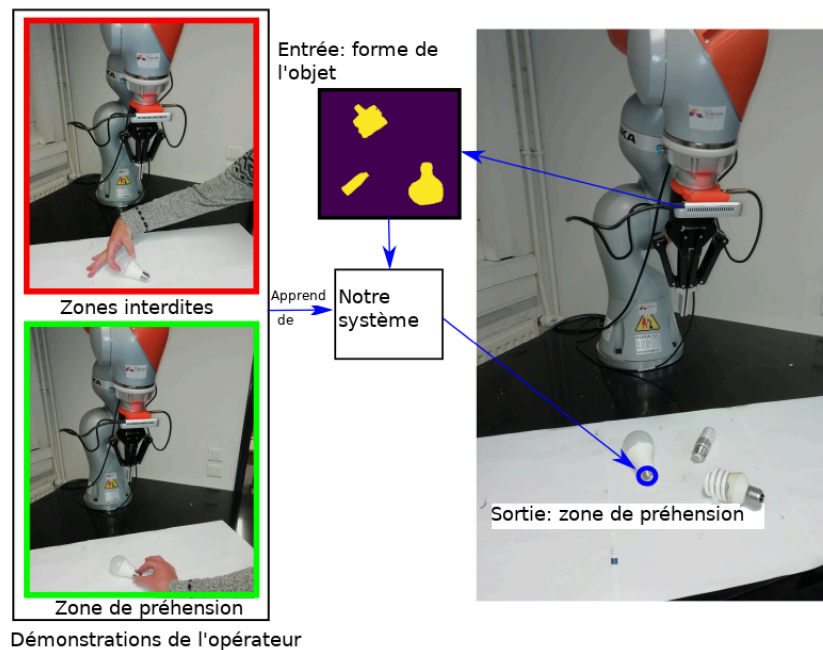


Figure B.11: Tout d’abord, pour un nouvel objet, le système apprend à partir de la démonstration de l’opérateur. Après quelques minutes d’entraînement, le système est capable de retrouver la zone désignée sur une carte de profondeur.

B.3.2-I Méthodologie: Nous avons défini une pipeline de segmentation sémantique par pixel, où l’entrée est une image de profondeur de la scène et la sortie, les paramètres de préhension $\mathbf{g} = (x, y, z, \theta)$ pour saisir l’objet sur la zone désignée. Les coordonnées (x, y, z) représentent le centre de l’outil de préhension, et θ est l’angle de la préhension dans le plan. Les paramètres de préhension sont directement dérivés de l’image de sortie. Afin d’assurer un apprentissage rapide et de généraliser avec peu de démonstration, plusieurs axes ont dû être exploités.

Pour ce faire un jeu de données d’apprentissage est généré directement à partir d’une ou plusieurs démonstration(s) de l’opérateur de zones autorisées et de zones interdites (Figure B.12). La constitution du jeu de données se fait directement à partir de la démonstration d’un opérateur sans utiliser de bases de données externes. Les gestes de préhension sur les zones autorisées et/ou interdites sont mémorisés en enregistrant les coordonnées des doigts (Figure B.12 -a). Ensuite (Figure B.12 -b), une forme 2D de l’objet est obtenue à partir de la binarisation de l’image de profondeur. Des étiquettes sont générées sous la forme d’images où les pixels autorisés ont une valeur de 1 pour les zones autorisées, 0 pour les zones neutre -1 pour les pixels des zones interdites. L’utilisation de la forme 2D réduit la taille de l’espace d’entrée et contribue à notre objectif de généralisation. Si nécessaire, plusieurs démonstrations du même objet à différents endroits sont collectées. Nous avons choisi d’utiliser les doigts de l’opérateur pour bénéficier directement de l’interaction homme-robot (la démonstration pourrait également se faire intuitivement via une interface graphique où l’opérateur renseigne les zones sur la forme 2D). Le nombre de couples (**images, étiquettes**) est ensuite augmenté suivant plusieurs types de transformations afin d’augmenter la variabilité du jeu d’entraînement (Figure B.12 -c) avant exploitation par un réseau de neurones profond.

B.3. APPROCHE ML POUR LA PRÉHENSION PLANAIRE

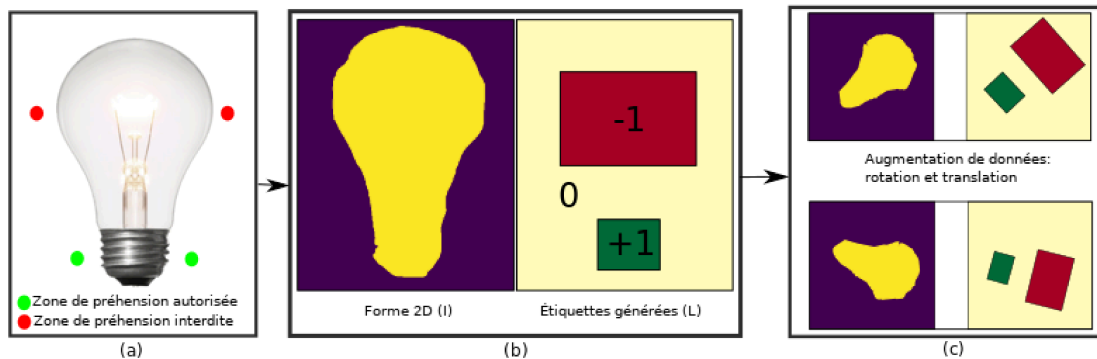


Figure B.12: Capture de données et augmentation

L'architecture à base de réseau de neurones est présentée dans la Figure B.13 -b et exploite la notion de transfert d'apprentissage. Nous utilisons un CNN (réseau de neurones convolutif) composé d'un réseau Densenet 121 [16] pré-entraîné et un réseau CNN léger spécifique. L'approche utilisée est une régression, la fonction d'activation de la dernière couche du CNN léger étant une "tanh". Le CNN léger est entraîné à produire une représentation pixelisée de l'affordance de saisie (Figure B.13 -c).

Enfin cette carte d'affordance est post-traitée afin de déterminer les paramètres de préhension, localisation et orientation de l'objet (Figure B.13 -d), avant expression dans le repère robot des paramètres (x, y, z, θ) ainsi prédits (la préhension s'effectue alors par le robot muni d'une pince 2 doigts centrée en (x, y, z) dont l'orientation θ est perpendiculaire à celle de l'objet). La petite taille du CNN ainsi que la dimension des entrées permet un apprentissage très rapide de l'affordance en quelques minutes.

Par ailleurs, étant donné que les zones autorisées/neutres/interdites n'occupent pas la même surface, une fonction de coût particulière a dû être développée afin que le réseau puisse prendre ce déséquilibre en compte au cours de son apprentissage.

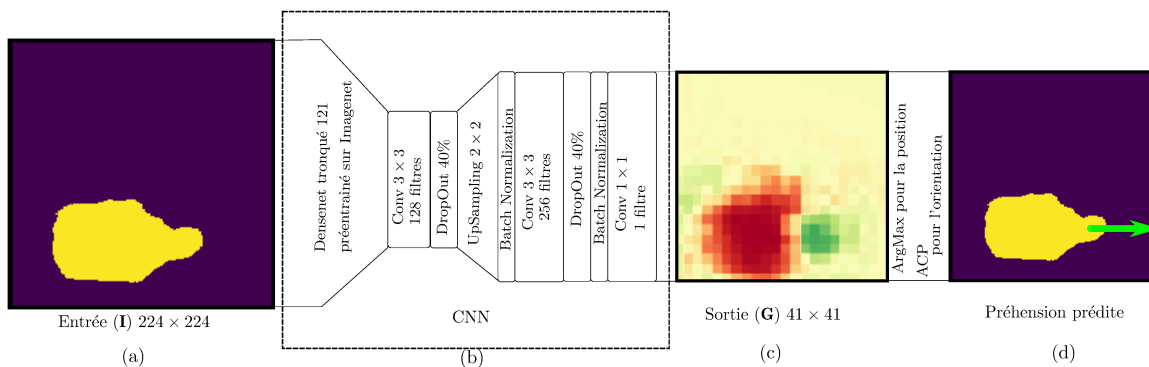


Figure B.13: Aperçu de notre pipeline CNN.

Le lecteur peut se référer à la section 4.3.2 pour plus de détails sur la méthodologie.

B.3.2-II Résultats expérimentaux: Pour évaluer la méthode proposée, nous avons réalisé une vaste série d'expériences sur un robot industriel collaboratif et divers objets (illustrés en B.14) . Nous avons étudié les performances de notre approche suivant différents paramètres liés à l'apprentissage :

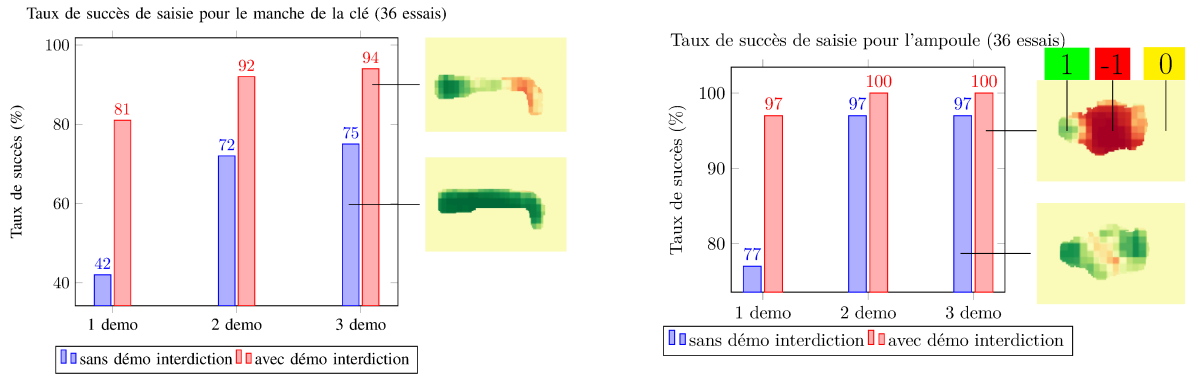
1. la saisie d'objets *référéncés* à la bonne zone dans différentes positions
2. les avantages de la pondération dans notre fonction de coût
3. les avantages de l'utilisation de démonstrations autorisées et interdites
4. la capacité de l'algorithme à généraliser à des objets similaires non référéncés.
5. la réalisation de la saisie dans un environnement composé de plusieurs objets *non référéncés* similaires



Figure B.14: Objets utilisés pour notre expérience, avec le nom des zones de préhension. La couleur verte (resp. la couleur rouge) indique la zone de préhension autorisée (resp. interdite). La couleur jaune est utilisée pour illustrer la zone autorisée/interdite et vice versa, en fonction de la tâche. Pour les pinces, deux zones autorisées différentes sont testées séparément

Le détail des expériences et leurs résultats est disponible dans le manuscrit (voir section 4.3.3). Dans la figure B.15, nous présentons le taux de réussite de saisie pour différents objets après entraînement, en fonction du nombre de démonstration(s). On remarque que le réseau est capable de généraliser à de nouvelles données avec seulement quelques démonstrations et que le taux de succès augmente avec le nombre de démonstrations de façon plus ou moins importante en fonction de la complexité de l'objet. Pour la clé par exemple, la similitude entre le manche et la tête rend plus difficile une préhension orientée, en apprenant uniquement à partir de la zone autorisée. Ceci montre la pertinence d'indiquer à la fois une zone autorisée et une zone interdite pour améliorer la précision. Par ailleurs, l'indication des zones interdites est aussi une garantie de sécurité puisque si le réseau réalise une mauvaise prédiction, le robot aura tendance à attraper un objet dans une zone neutre (espace de travail ou zone neutre de l'objet) plutôt que dans une zone interdite.

B.4. MISE EN OEUVRE ET VALIDATION SUR UNE TÂCHE D'APPRENTISSAGE DE TYPE "PRENDRE ET PLACER"



(a) Résultats pour le manche de la clé

(b) Résultats pour les ampoules

Figure B.15: Illustration des résultats de préhension suivant le nombre de démonstrations

B.3.2-III Conclusion du module: Dans ce travail, nous avons montré qu'une reconfiguration rapide d'un robot de préhension est possible avec une (ou quelques) démonstration(s). Notre méthode combine un espace d'état réduit, un CNN léger et une fonction de perte pondérée. Le réseau est capable d'apprendre rapidement à partir de quelques démonstrations sans nécessiter de jeux de données, de modèles de CAO ou de simulations répondant à notre motivation initiale de créer un système de préhension orientée qui puisse être rapidement et facilement reconfiguré par un opérateur. De plus, l'apprentissage des zones interdites rend ce processus plus sûr. Ainsi, il montre un bon potentiel d'intégration dans un contexte industriel.

Cependant, il présente des limites qui suggèrent des travaux supplémentaires. L'espace d'entrée choisie limite notre algorithme à des formes simples en 2D. Travailler directement avec la carte de profondeur de la caméra RGB-D permettrait de considérer des objets 3D plus complexes et/ou des objets en contact, au détriment du temps d'entraînement et donc de reconfiguration qui augmenterait en conséquence. De plus, la segmentation des objets peut être erronée dans certains cas. La détection de ces erreurs permettrait de demander de l'aide à l'opérateur en cas de besoin dans un scénario d'apprentissage continu.

B.4 Mise en oeuvre et validation sur une tâche d'apprentissage de type "prendre et placer"

L'organisation générale en terme d'implémentation, de modules, de modalité d'interaction et de choix de capteurs est illustrée Figure B.16. L'architecture a été validée sur un problème de type "prendre et placer".

B.4. MISE EN OEUVRE ET VALIDATION SUR UNE TÂCHE D'APPRENTISSAGE DE TYPE "PRENDRE ET PLACER"

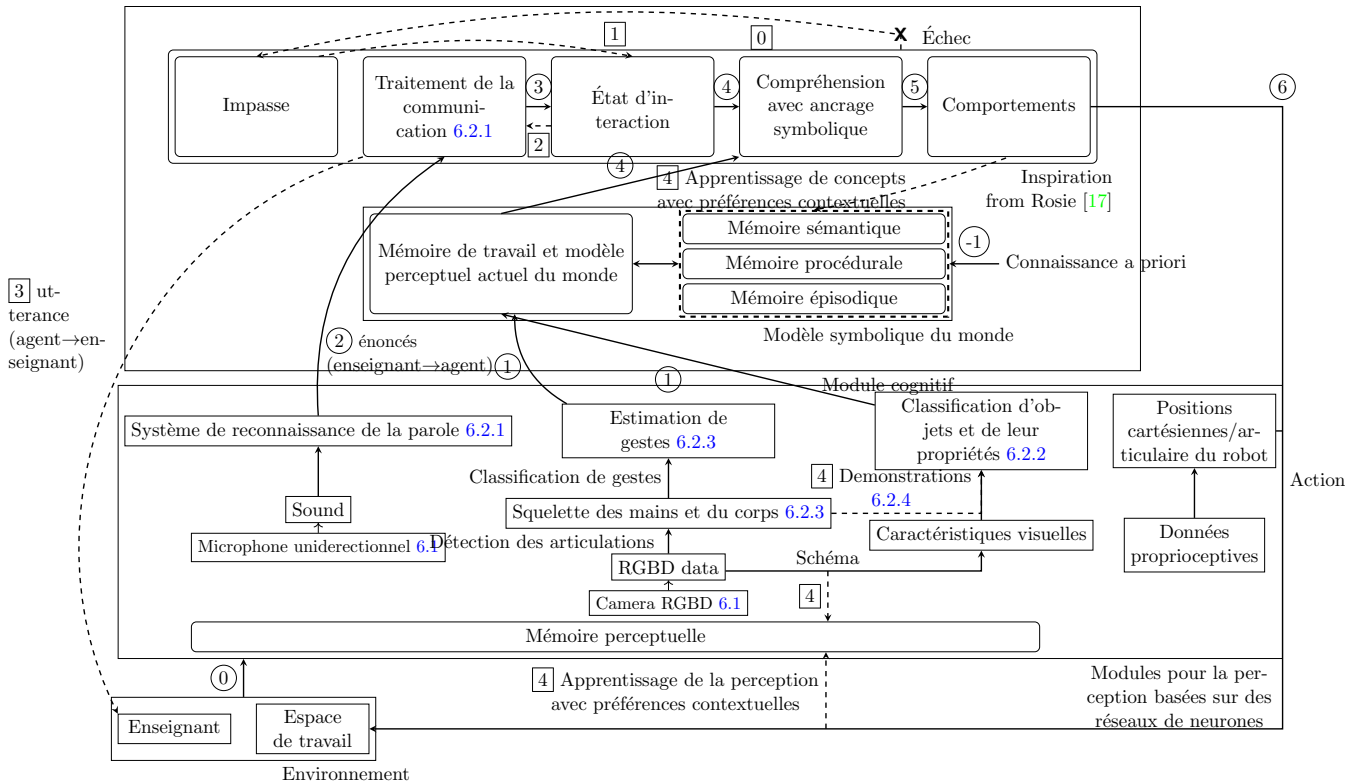


Figure B.16: Vue détaillée de l'architecture en termes de modules intégrés.

B.4.1 Choix des capteurs pour la perception et l'interaction SRA

Nous avons voulu que l'agent SRA s'adapte à l'humain et, par conséquent, que les capteurs soient les moins gênants possibles pour les travailleurs tout en permettant de détecter et de percevoir des actes de communication. Ceci a conduit à une liste non exhaustive de types de capteurs, qui pourraient être utilisés pour différentes modalités d'interaction. Nous pouvons distinguer les moyens de communication verbaux et non verbaux. Les deux sont importants lors de la définition d'une tâche, car certaines informations peuvent être plus facilement partagées par des mots ou par une interaction non verbale. Après avoir listé les principaux capteurs et les modalités d'interaction associées, nous avons sélectionné ceux qui semblent les plus utiles pour une première intégration et validation dans l'architecture. Nous avons voulu assurer une communication la plus naturelle possible entre l'homme et l'agent SRA. Les principaux critères considérés pour remplir nos spécifications d'interaction ont été les suivants :

1. Les perceptions potentielles après traitement des signaux doivent faire sens et être naturelles pour un humain.
2. Les capteurs doivent être aussi non-invasifs que possible pour être acceptés par les personnes interagissant avec le robot.
3. La programmation et le traitement doivent être simples et rapides pour permettre une communication quasi temps-réel.

B.4. MISE EN OEUVRE ET VALIDATION SUR UNE TÂCHE D'APPRENTISSAGE DE TYPE "PRENDRE ET PLACER"

4. Les capteurs doivent être suffisamment robustes par rapport aux perturbations de l'environnement.

Le choix des signaux étant centré sur l'homme, nous avons privilégié des critères de moyens de communication naturels et non invasifs.

L'un des moyens les plus naturels de communication non verbale est le geste, qui peut être facilement détectée par les capteurs de vision, à condition qu'il n'y ait pas d'occlusion. Pour leur polyvalence et leur facilité d'utilisation pour la compréhension de l'environnement, nous avons envisagé d'utiliser une caméra RGBD.

La parole reste notre principal moyen de communication. Il est donc essentiel d'équiper les robots de capacités de traitement du langage parlé afin d'assurer une communication naturelle pour les composants de haut niveau de l'architecture. Comme pour la vision, les microphones sont des capteurs relativement peu invasifs. Pour limiter les phénomènes de bruit ambiant, nous avons choisi un casque unidirectionnel.

La section 6.1 du manuscrit détaille de manière plus significative les capteurs envisagés avec une comparaison qualitative dans le tableau 6.1.

B.4.2 Intégration des modules de perception et d'action

Les capteurs choisis sont ensuite exploités par l'agent SRA grâce à plusieurs modules perceptifs que nous avons intégrés dans l'architecture pour la compréhension de l'espace de travail et l'interaction homme/robot. Plus précisément, nous avons adapté et intégré des modules pour la reconnaissance vocale (basée sur Google Speech), le traitement sémantique du langage (basé sur FLAIR), l'estimation de la pose et des gestes (basé sur OpenPose), et l'enseignement à partir de démonstrations (basé sur notre module d'apprentissage par démonstration). Plus d'informations sur cette intégration sont disponibles dans la section 6.2 de la thèse.

B.4.3 Scénario de validation

L'architecture actuelle a été validée sur un robot collaboratif UR10e 6 DOF pour l'apprentissage de la tâche "donner". Le cobot exploite des connaissances antérieures pour apprendre les variations de la tâche à effectuer, pour différents objets et suivant les préférences de préhension de différents opérateurs. Ces connaissances préalables sont intégrées à différents niveaux et sont présentées dans le tableau B.2. Comme indiqué dans nos spécifications, l'apprentissage se fait par un processus en ligne, mixte, incrémental et prenant en compte les préférences humaines. Le tableau B.3 présente les principales inconnues et les connaissances acquises à la fin du scénario d'enseignement interactif. Le tableau B.4 détaille le scénario et montre le processus d'apprentissage incrémental et interactif qui exploite à la fois les connaissances antérieures, les informations apprises et les démonstrations ou instructions humaines.

Le scénario d'enseignement interactif utilisé pour la validation peut être décomposé en deux phases principales :

- Une interaction avec un humain H_1 pour valider la capacité du SRA à exploiter les connaissances, à demander uniquement les connaissances manquantes et à apprendre de manière incrémentale

B.4. MISE EN OEUVRE ET VALIDATION SUR UNE TÂCHE D'APPRENTISSAGE DE TYPE "PRENDRE ET PLACER"

des variations de la tâche "donner" pour différents objets.

- Une interaction avec un humain H_2 qui est inconnu du SRA et qui a des caractéristiques et des préférences différentes de celles de H_1 . La tâche "donner", qui est maintenant une tâche connue, est demandée par H_2 pour un objet appris avec H_1 . Cette phase, durant laquelle le SRA demande et apprend naturellement les préférences de H_2 , valide l'adaptation à l'individu.

La figure B.17, illustre comment le module d'apprentissage par démonstration est exploité au sein de l'architecture pour que l'agent puisse adapter l'affordance de préhension en fonction de la personne interagissant avec le robot.

Le détails des phases est disponible dans la section 6.3 du manuscrit. Une vidéo de démonstration de cette validation est également disponible [ici](#)⁹.

Table B.2: Aperçu des connaissances *a priori*

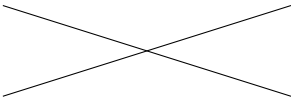
Représentation	Connaissances préalables construites dans l'architecture (étape -1)
Perceptions	Extraction de caractéristiques à partir de réseaux neuronaux pré-entraînés. Capacités de segmentation sensorielle : suppression de l'arrière-plan, segmentation des proto-objets Reconnaissance de la pose humaine Détection de tag ArUco. Reconnaissance de mots et de la parole (STT). Analyse sémantique avec un protocole de communication de base.
Action-s/tâches	prendre(objet) sous la forme d'un arbre de comportement (BT). placer(position) comme un BT
Préférences	Les humains ont des préférences et des caractéristiques, H_1 est connu et est droitier

Table B.3: Synthèse de ce qui sera appris au cours du processus d'apprentissage interactif incrémentale de la tâche inconnue donner

Représentation	Inconnues	Connaissances apprises
Perception	clé et tournevis	caractéristiques perceptives et capacité de préhension de la clé et du tournevis.
Action-s/tâches	donner(objet)	donner(objet) comme un BT
Préférences	Affordances et préférences d'action	affordance de préhension préférée (clé prise par la tête ou la poignée) Adaptation pour donner l'objet dans la main dominante

⁹<https://www.youtube.com/watch?v=EAuLMnQULBO>

Table B.4: Détail du processus d'apprentissage pendant le processus d'apprentissage interactif incrémentiel de la tâche inconnue "donner"

Étapes de l'interaction	Inconnues \rightarrow Connaissances apprises	Connaissances exploitées	Intervention humaine
Étape 0 : H_1 initie l'interaction		<ul style="list-style-type: none"> • Connaissances a Priori (CP) intégrées avec la vision et la parole voir tableau B.2 • Base de données contenant caractéristiques et préférences connues de H_1. 	Identifiant ArUco
Étape 1 : H_1 demande : "donner tournevis"	<ul style="list-style-type: none"> • donner(objet) \rightarrow nouveau but G1 : donner = Dans(tournevis, main) 	<ul style="list-style-type: none"> • CP intégrée avec vision et parole 	Reconnaissance de la parole et analyse sémantique
Étape 2 : H_1 demande : "L'objectif est le tournevis dans la main"	<ul style="list-style-type: none"> • tournevis \rightarrow caractéristiques perceptuelles 	<ul style="list-style-type: none"> • CP intégrée avec vision et parole, • G1 	Démonstration de pointage avec validation vocale
Étape 3 : H_1 explique : "prendre le tournevis"	<ul style="list-style-type: none"> • donner(objet) \rightarrow prendre(objet) + ... 	<ul style="list-style-type: none"> • CP • G1 	Reconnaissance vocale et analyse sémantique
Étape 4 : H_1 montre sa préférence	<ul style="list-style-type: none"> • donner(obj) \rightarrow prendre(obj) + ... • préférences de H_1 pour l'affordance du tournevis \rightarrow poids de réseaux spécifiques 	<ul style="list-style-type: none"> • CP • G1 • prendre(obj) a besoin d'une affordance de préhension 	Démonstration des zones autorisées et interdites avec validation vocale
Étape 5 : H_1 explique : "placer dans main (tournevis)"	<ul style="list-style-type: none"> • donner(objet) \rightarrow donner(objet) = prendre(objet) + placer dans(main) 	<ul style="list-style-type: none"> • CP • G1 • placer dans(loc) • main dominante H_1 	Reconnaissance vocale et analyse sémantique
Step 6: H_1 demande "donner clé"	<ul style="list-style-type: none"> • clé \rightarrow caractéristiques perceptuelles de la clé 	<ul style="list-style-type: none"> • CP • nouveau but G2 = donner(clé) 	Démonstration en pointant avec validation par la voix
Step 7: H_1 montre sa préférence	<ul style="list-style-type: none"> • préférence d'affordance pour H_1 pour clé \rightarrow préférence de H_1 	<ul style="list-style-type: none"> • CP • but • donner(obj) dont la sous action prendre requière une affordance • main dominante H_1 	Zones autorisées et interdites
Étape 8 : H_2 initie l'interaction	<ul style="list-style-type: none"> • nom H_2 • main dominante H_2 \rightarrow caractéristiques de H_2 dans la base de données 	<ul style="list-style-type: none"> • CP 	entrées clavier
Étape 9 : H_2 montre sa préférence	<ul style="list-style-type: none"> • préférence d'affordance de H_2 pour clé \rightarrow affordance de la clé comme préférence de H_2 	<ul style="list-style-type: none"> • CP • nouveau but G3 = donner(clé) • donner(objet) + besoin de l'affordance de préhension pour prendre • main dominante H_2 	Démonstration des zones autorisées et interdites avec validation vocale

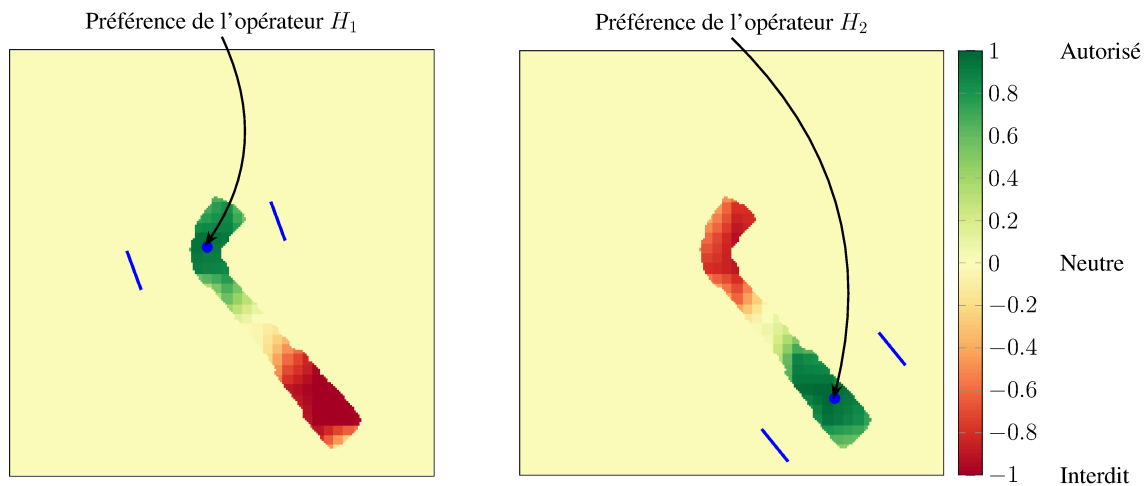


Figure B.17: L'agent SRA peut adapter dynamiquement sa prédiction d'affordance en fonction de l'humain avec lequel il interagit, comme le montre l'exemple de la clé. À gauche, H_1 a appris à l'agent SRA à saisir la clé par la tête. À droite, H_2 a appris à l'agent à saisir la clé par le manche. En fonction de l'opérateur avec lequel le SRA interagit, le robot apporte alors la clé dans sa main dominante et selon sa préférence.

B.4.4 Conclusion

Dans ce chapitre, nous avons fourni une implémentation et une validation de plusieurs des principales spécifications de l'architecture. L'architecture de base a été développée en python3 pour être facilement liée à différents modules (perception, apprentissage, ...). Nous avons fait en sorte qu'un robot réel apprenne une tâche inconnue, en construisant des connaissances procédurales et perceptuelles prenant en compte les préférences et les caractéristiques des humains. L'utilisation d'arbres de comportement fournit un moyen efficace de gérer la représentation des connaissances, l'apprentissage et l'exécution des séquences de tâches. En effet, de cette façon, lorsque l'agent fait face à des échecs lors de l'exécution de comportements, il n'a qu'à demander les caractéristiques manquantes. Les informations sont fournies de manière interactive par l'humain dans un contexte d'initiative mixte enseignant/apprenant. En utilisant la parole et les gestes comme moyens de communication et en utilisant des modules intégrés tels que notre module d'apprentissage par la démonstration, les personnes sans expertise en programmation peuvent plus naturellement reconfigurer les robots.

B.5 Vers plus de flexibilités

Ce travail est une première étape dans le développement d'un système cognitif permettant à un humain non programmeur d'enseigner plus naturellement des comportements flexibles à un robot in-

dustriel collaboratif. Dans l'ensemble, nous avons construit l'architecture en gardant à l'esprit la modularité, car l'utilisation de composants modulaires est essentielle pour une meilleure compréhension, une évaluation de la confiance et l'évolution potentielle de l'architecture. Bien sûr, le SRA idéal est encore loin d'être atteint. Plusieurs perspectives et dont celle de l'intégration des incertitudes sont abordée dans le manuscrit. En particulier cette dernière est primordiale vis a vis de nos spécifications et fait l'objet d'une description détaillée dans le chapitre 5.

B.5.1 Obtenir un niveau d'incertitude

Jusqu'à présent, nous avons décrit les capacités d'apprentissage des compétences en termes de composantes symboliques et connexionnistes. Nous avons montré qu'un manque de connaissances symboliques sur une tâche, en termes d'informations procédurales ou perceptives, conduit à un *échec* et déclenche un événement d'apprentissage interactif. Cependant, cela n'est pas suffisant car les modules perceptifs, tels que les modules basés sur des réseaux neuronaux profonds, peuvent être non fiables face à de situations nouvelles. Par conséquent, de mauvaises prédictions peuvent conduire à des décisions potentiellement dommageables. L'agent SRA doit connaître le niveau de certitude ou d'incertitude relatifs aux processus de perception et de raisonnement. Il s'agit d'un indicateur clé pour doter l'agent SRA d'une meilleure compréhension de ce qu'il sait, de ce qu'il ne sait pas et de ce dont il n'est pas certain pour prendre de meilleures décisions.

Par exemple, dans notre module d'apprentissage d'affordance de préhension, nous avons réduit les risques de saisie dans une zone interdite en apprenant une zone neutre de séparation. Cependant des échecs restent possibles lorsque l'agent n'a pas vu suffisamment de démonstrations pour certains objets. Ainsi, l'agent a besoin d'un moyen d'estimer si la zone de préhension autorisée prédite est pertinente ou non. Il doit avoir du recul sur ses prédictions.

Il existe une littérature abondante sur la taxonomie de l'incertitude (voir [18] pour une étude complète). Brièvement, en général l'incertitude τ d'un modèle d'apprentissage automatique peut être décomposée à haut niveau en deux types d'incertitude, l'incertitude *aléatoire* (τ_a) liée à la nature physique du phénomène (inhérent au phénomène, quelque soit le nombre de données) et l'incertitude *épistémique* (τ_e) liée au manque d'expérience (et donc de données) de l'agent.

Comme dans la plupart des problèmes d'apprentissage automatique, nous pouvons distinguer les problèmes de classification et les problèmes de régression.

Pour la classification, un modèle doit fournir une étiquette avec son degré de confiance. Par exemple, demander à un réseau de neurones profond de prédire un résultat avec 100% de confiance pour "un pile ou face" n'a aucun sens. Nous aimerions que le réseau soit incertain quant à sa prédiction et, dans l'idéal, qu'il produise une distribution des résultats possibles ($1/2 \rightarrow$ pile, $1/2 \rightarrow$ face). Le réseau doit également avoir un comportement similaire avec des données éloignées de celles sur lesquelles il a été entraîné ou "à la frontière" entre des classes. D'une manière générale, pour un problème de classification, l'incertitude devrait produire une distribution de prédiction sur les classes. De cette façon, la confiance dans une classe prédite, ainsi que les confusions particulières avec d'autres classes, peuvent être mises en évidence. Même si les sorties softmax d'un réseau de classification ressemblent à une telle distribution, elles sont connues pour être sujettes à une mauvaise calibration et à une confiance excessive [19]. Par conséquent, on ne peut pas s'y fier en tant que mesure de confiance et d'incertitude.

La représentation de l'incertitude liée aux prédictions d'un réseau de neurones est un sujet ouvert

en apprentissage profond. Dans la littérature, plusieurs approches sont étudiées pour apprendre de tels modèles afin de dériver une métrique d'incertitude fiable. On peut citer quelques exemples et illustrées dans la section 5.2 du manuscrit :

- les techniques d'apprentissage basées sur la méthode bayésienne
- les méthodes ensemblistes
- les modèles tenant compte d'une notion de distance dans l'espace des prédictions
- les mesures externes

Prenons l'exemple des méthodes ensemblistes qui permettent de générer et d'agréger un N-échantillon de prédictions. Pour un problème de classification, la classe retenue est celle la plus prédite, et la mesure d'incertitude correspond à la proportion des prédictions dans l'ensemble des autres classes. Pour un problème de régression la valeur moyenne et l'écart-type des prédictions fournissent un intervalle de confiance autour de la valeur moyenne prédite.

Une bonne estimation et taxonomie de l'incertitude aidera à quantifier la confiance que l'on peut avoir dans les prédictions de l'agent SRA et pour faire face aux potentiels biais. Du point de vue de l'opérateur, cela pourrait améliorer les garanties de sécurité et d'acceptabilité qui sont des exigences fortes pour l'industrie. Du côté de l'agent SRA, c'est un moyen de questionner et de raisonner sur son propre comportement.

B.5.2 Incertitude dans la prise de décision

Une fois que l'on dispose d'une mesure d'incertitude bien calibrée [19] (lorsque la précision prédictive est proche de la confiance en les prédictions), il est possible d'utiliser cette mesure d'incertitude dans la prise de décision.

En suivant [20], nous pouvons adapter la matrice de contingence classique [21] comme indiqué dans le tableau B.5. A partir de cette matrice, nous pouvons dériver des mesures de performance, ainsi qu'un processus d'apprentissage actif pour notre SRA.

		Résultats de prédictions	
		Mauvaise	Bonne
Incertitude	Elevée (Question)	Vrai Positif (VP)	Faux Positif (FP) (fausse alarme)
	Faible (Action)	Faux Négatifs (FN) (non détection)	Vrai Négatifs (VN)

τ_{thresh}

Table B.5: Tableau de contingence avec mesure d'incertitude τ . Au-delà d'un seuil d'incertitude τ_{thresh} , une alarme est déclenchée, par exemple dans un cadre d'apprentissage actif.

Étant donné un seuil d'incertitude τ_{thresh} , les prédictions peuvent être classées en deux catégories : confiantes (incertitude "faible") et incertaines (incertitude "élevée"), pour lesquelles la décision respective est de déclencher ("Positive") ou non ("Négative") une alarme. La pertinence du déclenchement ou non d'une alarme est désignée par "Vrai". Étant donné une prédiction, "Vrai" correspond à la diagonale principale de la matrice de contingence, c'est à dire d'avoir déclenché (resp ne pas déclenché) une alarme si la prédiction était effectivement mauvaise (resp bonne). La proportion de "Vrai" est ainsi une mesure d'expertise de l'agent à déclencher ou non une alarme à bon escient.

Cette matrice de contingence peut être considérée comme une classification binaire permettant d'évaluer les potentielles mauvaises prédictions, par exemple concernant les prédictions d'un réseau de neurones. Une bonne métrique d'incertitude devrait maximiser les proportions dans la diagonale principale (VP et VN), et éradiquer les non détections (FN) correspondant aux mauvaises prédictions qui sont considérées comme juste et certaines, tout en limitant le nombre de fausses alarmes (FP). En effet, les non détections (FN) peuvent entraîner des problèmes de sécurité dans les applications industrielles de robotique collaborative, et les fausses alarmes (FP) représentent un coût supplémentaire dans un contexte industriel. Une alarme déclenche un échec du système qui doit être résolue par des interactions humaines (démonstrations et énoncés) dans le processus d'apprentissage interactif. Par conséquent, nous voulons limiter le nombre d'interactions inutiles (FP).

La table de contingence peut être exploitée pour fixer le seuil τ_{thresh} de la quantité de vrais positifs acceptables par rapport aux faux positifs, représenté dans le tableau B.5. Au-dessus de ce seuil, l'agent est suffisamment confiant et décide d'agir. En dessous de ce seuil, il doit décider de demander. Ceci est illustré dans la figure B.18. Étant donné un histogramme de prédictions, on représente en vert l'histogramme des bonnes prédictions et en rouge l'histogramme des mauvaises prédictions pour une incertitude τ donnée. Divers outils statistiques (tels que les courbes ROC) et indicateurs clés de performance (KPI) dérivée de la table de contingence et reposant sur le calcul de plusieurs ratios peuvent être utilisés pour affiner l'analyse et pour la prise de décision (voir section 5.3.2 du manuscrit).

Dans un contexte industriel, il nous faut éradiquer les non détections (FN) tout en limitant les fausses alarmes (FP). Cela motive la mise en place d'un apprentissage actif où τ_{thresh} évoluera (augmentera) dans le temps. Un bon modèle de prédiction associé à un modèle d'incertitude bien calibré et un apprentissage actif devrait permettre de séparer clairement les deux distributions, c'est à dire de rendre expert notre agent (matrice de contingence diagonale).

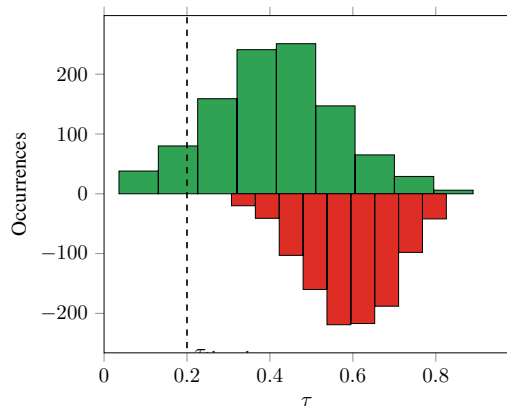


Figure B.18: Illustration de l'utilisation de τ_{thresh} pour limiter les mauvaises actions

B.5.3 Apprentissage actif

Maintenant que nous avons une meilleure idée de ce qu'est l'incertitude et de la façon dont elle peut être estimée, nous pouvons nous concentrer sur la façon de l'intégrer dans une application robotique réelle. La notion d'incertitude a été utilisée dans plusieurs travaux en robotique à des fins d'apprentissage. Il n'existe pas de définition claire, mais ces méthodes exploitent une certaine notion d'inconnu pour susciter des comportements spécifiques d'exploration. Le domaine de l'apprentissage actif se concentre sur les stratégies qui peuvent réduire l'inconnu et améliorer le modèle sous-jacent. Nous pouvons donc exploiter les notions d'incertitude dans le contexte de l'apprentissage actif et de l'intégrer dans l'architecture de l'agent SRA. Une méthodologie générale est illustrée dans la figure B.19 et peut être résumée à haut niveau par la procédure suivante :

- Un réseau de neurones est entraîné sur un ensemble de données avec une fonction de perte L et avec une notion d'incertitude, idéalement calibrée.
- Ensuite, la prédiction et l'incertitude du modèle sont exploitées au cours du processus d'apprentissage interactif de l'agent SRA.
- Une fois la prédiction effectuée, l'incertitude est propagée au niveau supérieur afin que l'agent décide si il doit agir ou demander en fonction du paramètre d'incertitude τ par rapport au seuil $\tau_{thresh}(t)$ qui évolue au fil de l'apprentissage actif. Plusieurs cas sont possibles et sont précisés dans la section 5.4.2 du manuscrit.

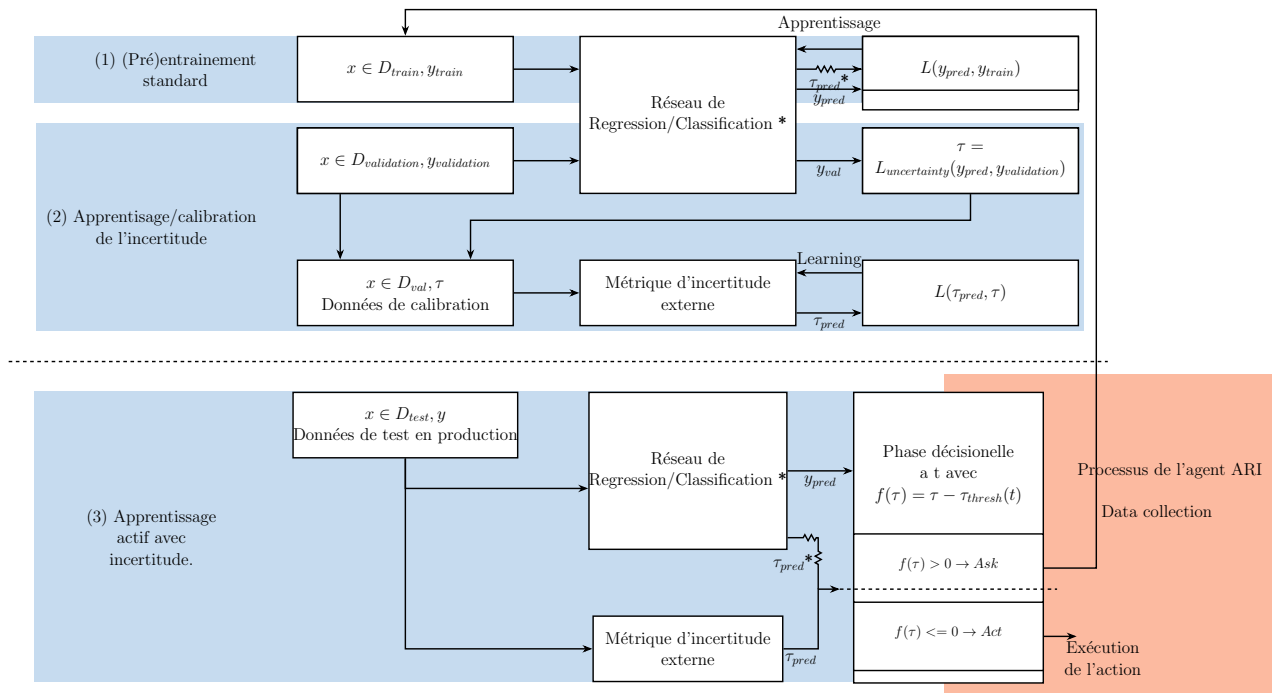


Figure B.19: Processus général pour l'apprentissage actif

B.5. VERS PLUS DE FLEXIBILITÉS

Afin d'intégrer les travaux sus-mentionnés sur l'architecture, au niveau de la prise de décision, nous devons étendre le modèle de comportement de l'architecture avec des capacités de traitement et de propagation de l'incertitude. Cette dernière peut être effectuée par des arbres de comportements spécifiques (BBT) [22]. Dans ce cadre, l'agent peut raisonner sur un état de croyance dans sa mémoire de travail et propager l'incertitude à travers les noeuds de l'arbre. L'agent ne raisonne plus dès lors sur des valeurs uniques mais sur des distributions d'états et d'actions.

Nous pouvons ensuite intégrer le processus d'apprentissage actif dans ce cadre par exemple dans l'exemple de préhension (voir la figure B.20). Étant donné plusieurs conditions et modules perceptifs intégrant la notion d'incertitude, l'agent SRA sélectionne les prédictions en fonction du seuil $\tau_{thresh}(t)$. Les prédictions et les incertitudes sont propagées vers les noeuds supérieurs, une impasse se produit et peut être résolue en demandant des démonstrations dans un cadre d'apprentissage actif, si aucun état candidat s pour la préhension n'est valide.

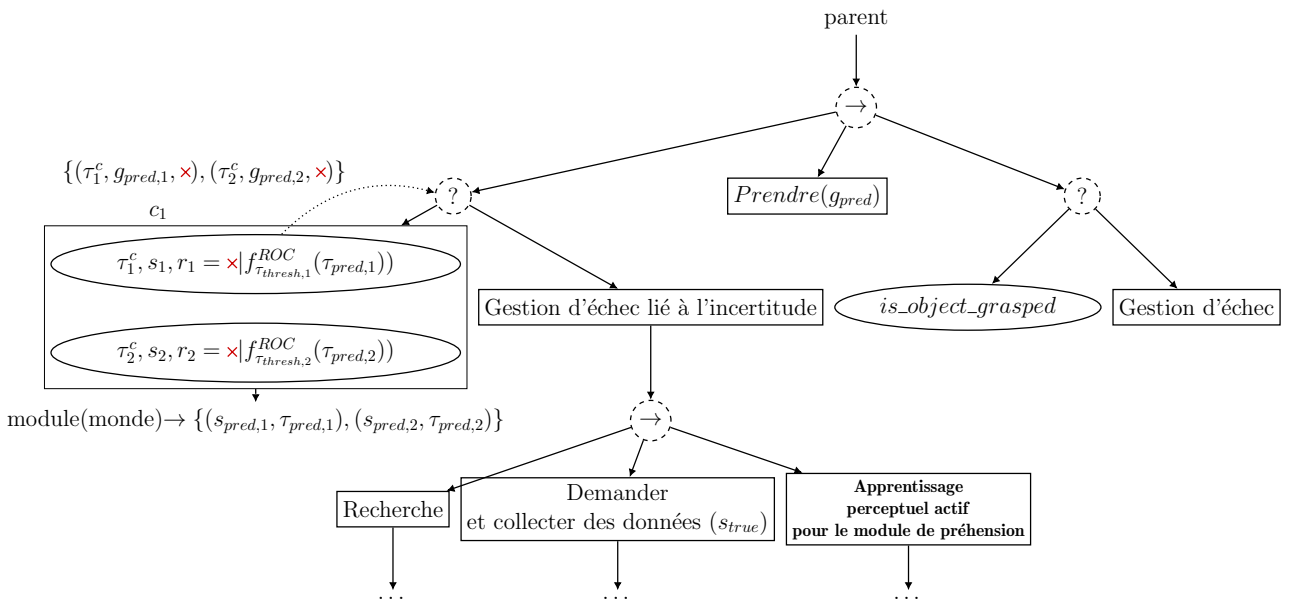
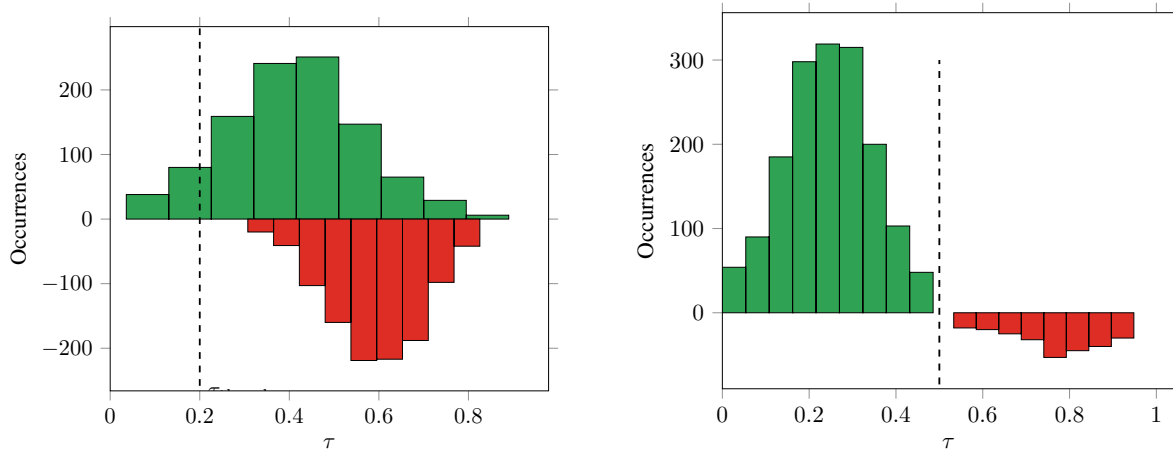


Figure B.20: L'agent SRA peut exploiter la structure spécifique des BBTs pour gérer l'incertitude fournie par les modules de niveau inférieur

En reconsidérant l'histogramme de prédiction (Figure B.21a), idéalement, après quelques itérations, la distribution des bonnes et mauvaises prédictions devrait être bien séparée (B.21b). Le SRA devient expert lorsque les demandes deviennent rares et que les actions sont bonnes. Les demandes ne devraient concerner que certains cas exceptionnels particulièrement éloignés des données d'apprentissage. Cela pourrait être réalisé en faisant varier le seuil d'incertitude au fur et à mesure que l'agent gagne en expertise.



(a) L'histogramme des prédictions n'est pas bien séparable au début de l'apprentissage.

(b) Histogramme idéal de prédictions. Étant donnée le seuil d'incertitude, une action certaine est toujours bonne, tandis qu'une non-action pour cause d'incertitude est toujours justifiée car elle aurait conduit à une erreur. Les non-détections sont éliminées.

Figure B.21: Illustration de l'utilisation de τ_{thresh} pour éviter les mauvaises actions, compte tenu de l'incertitude τ .

B.5.4 Conclusion

Dans ce chapitre, nous avons défini comment l'incertitude peut être vue comme une composition de différents types d'incertitudes. Son intégration est déterminante, si l'on veut qu'un agent SRA soit capable de raisonner et de faire face à plusieurs types de biais. De plus, nous avons montré que l'exploitation de l'incertitude peut être au cœur d'un processus d'apprentissage actif au sein de l'architecture. En effet, en appliquant un seuil à la quantité d'incertitude acceptable, nous pouvons dériver une notion de curiosité ou de motivation qui peut être exploitée par l'agent pendant l'interaction en décidant de demander plutôt que d'agir. Ainsi, un agent SRA peut avoir beaucoup plus de recul sur ses capacités de prédiction, ce qui conduit à des comportements plus sûrs et à la possibilité de demander de l'aide dans des scénarios plus riches ou éloignés de ceux sur lesquels il a appris.

D'autres perspectives de plus long terme sont également abordées dans le manuscrit (voir section 7.3) et questionnent certaines limites et potentiels d'amélioration de l'architecture en termes :

- de raisonnement sur une taxonomie plus fine d'incertitudes intégrant une distinction entre les différents types d'incertitude
- de fusion de données multimodales qui devrait permettre à l'agent de raisonner sur les incertitudes en fonctions de la nature des modalités de perception
- d'amélioration de la généralisation des préférences dans le cadre multi-utilisateur où des préférences similaires peuvent se retrouver chez plusieurs individus.

- d'amélioration des modèles de comportement en termes d'apprentissage et de représentations, par exemple en exploitant des modèles causaux.

B.6 Conclusion générale

Nous avons présenté un prototype d'architecture de base axé sur l'intégration de plusieurs fonctions relatifs à la planification, à l'action et à l'apprentissage incrémentale de compétences dans un cadre d'apprentissage interactif. L'état actuel de développement de l'architecture a pu valider dans un scénario simple la plupart de nos spécifications. Des validations expérimentales ont été faites avec robot collaboratif industriel pour l'enseignement de tâches liées à la préhension, avec des préférences pour la manipulation. L'agent SRA est capable d'apprendre **en ligne** des représentations de haut niveau et de bas niveau de la tâche, pendant **une interaction d'initiative mixte**. L'utilisation d'**arbres de comportements** comme modèle permet de construire des représentations **modulaires** et explicites de la tâche qui aident à **expliquer** et **interpréter** les comportements du robot. Le processus décisionnel exploite des modules perceptifs basés sur des réseaux de neurones pour apprendre des caractéristiques perceptives complexes telles que l'affordance de préhension. **L'apprentissage est rapide** grâce à l'exploitation de réseaux pré-entraînés, à l'apprentissage par transfert vers des tâches spécifiques et aux techniques d'augmentation avec des sous-réseaux spécialisés. De plus, comme les réseaux neuronaux les plus grands sont difficilement interprétables, l'utilisation de sous-réseaux spécialisés est susceptible d'aider à **interpréter** et **corriger** les défaillances du système. Grâce à la modularité, si la prédiction d'un sous-réseau échoue, il peut être possible de corriger uniquement ce sous-réseau sans affecter les autres modules. Nous avons montré qu'il est possible, à travers un scénario d'initiative mixte, d'enseigner une tâche avec des variations. L'adaptation aux **préférences humaines**, validée ici sur les préférences d'affordance de préhension et l'adaptation à la main dominante, est une exigence importante de notre architecture. En effet, elle offre une interaction personnalisée qui est susceptible d'aider à l'**acceptabilité** des systèmes robotiques par les opérateurs.

Notre travail est un premier pas vers un assistant robotique industriel collaboratif intelligent. L'architecture est ouverte et extensible à plusieurs améliorations. Enfin, au fur et à mesure que l'architecture mûrira, il deviendra nécessaire de la tester dans des contextes plus complexes et avec de vrais non-experts.

B.6. CONCLUSION GÉNÉRALE

References

Contents

Chapitre 1: Introduction	193
Chapitre 2: State of the art on cognitive systems	195
Chapitre 3: Design of a cognitive architecture for Industry 4.0	202
Chapitre 4: Complementary ML approaches for IRL on planar grasping use cases	204
Chapitre 5: Learning under uncertainty	208
Chapitre 6 : Implementation and validation on a planar pick and place learning task	212
Chapitre 7: Conclusion and perspectives	215
Résumé étendu en français	217

Chapter 1: Introduction

- [1] A. Gasparetto and L. Scalerà. “A Brief History of Industrial Robotics in the 20th Century”. In: *Advances in Historical Studies* 08.01 (2019), pp. 24–35. ISSN: 2327-0438, 2327-0446. DOI: [10.4236/ahs.2019.81002](https://doi.org/10.4236/ahs.2019.81002).
- [2] Mohammad Safeea, Pedro Neto, and Richard Béarée. “Precise Hand-Guiding of Redundant Manipulators with Null Space Control for in-Contact Obstacle Navigation”. In: *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*. Vol. 1. Oct. 2019, pp. 693–698. DOI: [10/gm4hgs](https://doi.org/10/gm4hgs).
- [3] Iina Aaltonen and Timo Salmi. “Experiences and Expectations of Collaborative Robots in Industry and Academia: Barriers and Development Needs”. In: *Procedia Manufacturing*. 29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2019), June 24-28, 2019, Limerick, Ireland, Beyond Industry 4.0: Industrial Advances, Engineering Education and Intelligent Manufacturing 38 (Jan. 2019), pp. 1151–1158. ISSN: 2351-9789. DOI: [10.1016/j.promfg.2020.01.204](https://doi.org/10.1016/j.promfg.2020.01.204).
- [4] Pooja Makula et al. “Multimodal Smart Robotic Assistant”. In: *Proceedings of 2015 International Conference on Signal Processing, Computing and Control, ISPCC 2015*. Institute of Electrical and Electronics Engineers Inc., Jan. 2016, pp. 18–23. ISBN: 978-1-4799-8436-7. DOI: [10.1109/ISPCC.2015.7374991](https://doi.org/10.1109/ISPCC.2015.7374991).
- [5] Ronan Alexandre Riochet et al. “IntPhys: A Benchmark for Visual Intuitive Physics Reasoning”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021),

- pp. 1–1. ISSN: 1939-3539. DOI: [10 . 1109 / TPAMI . 2021 . 3083839](https://doi.org/10.1109/TPAMI.2021.3083839).
- [6] Allen Newell. “Précis of Unified Theories of Cognition”. In: *Behavioral and Brain Sciences* 15.3 (Sept. 1992), pp. 425–437. ISSN: 1469-1825, 0140-525X. DOI: [10/dpgd25](https://doi.org/10/dpgd25).
- [7] Christian Lebiere. “Act-R”. In: *Proceedings of the Twenty-Fourth Annual Conference of the Cognitive Science Society*. 2019. DOI: [10 . 4324/9781315782379-4](https://doi.org/10.4324/9781315782379-4).
- [8] Chris Eliasmith. *How to Build a Brain: A Neural Architecture for Biological Cognition*. Oxford University Press, June 2013. ISBN: 978-0-19-979454-6. DOI: [10 . 1093 / acprof : oso / 9780199794546 . 001 . 0001](https://doi.org/10.1093/acprof:oso/9780199794546.001.0001).
- [9] Feng-Xuan Choo. “Spaun 2.0: Extending the World’s Largest Functional Brain Model”. In: *undefined* (2018).
- [10] John E. Laird. *The Soar Cognitive Architecture*. 2018. DOI: [10 . 7551 / mitpress / 7688 . 001 . 0001](https://doi.org/10.7551/mitpress/7688.001.0001).
- [11] Paul Rosenbloom, Abram Demski, and Volkan Ustun. “The Sigma Cognitive Architecture and System: Towards Functionally Elegant Grand Unification”. In: *Journal of Artificial General Intelligence* 7 (Jan. 2016). DOI: [10/gd3zrf](https://doi.org/10/gd3zrf).
- [12] Matthias Scheutz et al. “An Overview of the Distributed Integrated Cognition Affect and Reflection DIARC Architecture”. In: *Cognitive Architectures*. Ed. by Maria Isabel Aldinhas Ferreira, João Silva Sequeira, and Rodrigo Ventura. Intelligent Systems, Control and Automation: Science and Engineering. Cham: Springer International Publishing, 2019, pp. 165–193. ISBN: 978-3-319-97550-4. DOI: [10 . 1007 / 978 - 3 - 319 - 97550 - 4 _ 11](https://doi.org/10.1007/978-3-319-97550-4_11).
- [13] Matej Hoffmann and Rolf Pfeifer. “The Implications of Embodiment for Behavior and Cognition: Animal and Robotic Case Studies”. In: *CoRR* abs/1202.0440 (Feb. 2012). DOI: [10/gm4hj4](https://doi.org/10/gm4hj4).
- [14] Sonia Chernova and Andrea Thomaz. “Robot Learning from Human Teachers”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8 (Apr. 2014), pp. 1–121. DOI: [10/gf7bxx](https://doi.org/10/gf7bxx).
- [15] John E. Laird et al. “Interactive Task Learning”. In: *IEEE Intelligent Systems* (2017). ISSN: 15411672. DOI: [10 . 1109 / MIS . 2017 . 3121552](https://doi.org/10.1109/MIS.2017.3121552).
- [16] Joost Broekens and Mohamed Chetouani. “Towards Transparent Robot Learning Through TDRL-Based Emotional Expressions”. In: *IEEE Transactions on Affective Computing* 12.2 (Apr. 2021), pp. 352–362. ISSN: 1949-3045. DOI: [10 . 1109 / TAFFC . 2019 . 2893348](https://doi.org/10.1109/TAFFC.2019.2893348).
- [17] François Héli on et al. “Learning Prohibited and Authorised Grasping Locations from a Few Demonstrations”. In: *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. Aug. 2020, pp. 1094–1100. DOI: [10/gk8djs](https://doi.org/10/gk8djs).
- [18] François Héli on et al. “Cognitive Architecture for Intuitive and Interactive Task Learning in Industrial Collaborative Robotics”. In: *The 5th International Conference on Robotics, Control and Automation, ICRC A 2021*. 2021. ISBN: 978-1-4503-8748-4.

Chapter 2: State of the art on cognitive systems

- [1] Iuliia Kotseruba and John K. Tsotsos. “40 Years of Cognitive Architectures: Core Cognitive Abilities and Practical Applications”. In: *Artificial Intelligence Review* 53.1 (July 2020), pp. 17–94. ISSN: 15737462. DOI: [10.1007/s10462-018-9646-y](https://doi.org/10.1007/s10462-018-9646-y).
- [2] Alejandro Barredo Arrieta et al. “Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI”. In: *Information Fusion* 58 (June 2020), pp. 82–115. ISSN: 1566-2535. DOI: [10.1016/j.inffus.2019.12.012](https://doi.org/10.1016/j.inffus.2019.12.012).
- [3] Guglielmo Papagni and Sabine Koeszegi. “Understandable and Trustworthy Explainable Robots: A Sensemaking Perspective”. In: *Paladyn, Journal of Behavioral Robotics* 12.1 (Jan. 2021), pp. 13–30. ISSN: 2081-4836. DOI: [10/gmsxft](https://doi.org/10/gmsxft).
- [4] Yoshua Bengio, Yann Lecun, and Geoffrey Hinton. “Deep Learning for AI”. In: *Communications of the ACM* 64.7 (June 2021), pp. 58–65. ISSN: 0001-0782. DOI: [10/gkx7tb](https://doi.org/10/gkx7tb).
- [5] Emily M. Bender et al. “On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜” in: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. FAccT 21. New York, NY, USA: Association for Computing Machinery, Mar. 2021, pp. 610–623. ISBN: 978-1-4503-8309-7. DOI: [10.1145/3442188.3445922](https://doi.org/10.1145/3442188.3445922).
- [6] Tom Brown et al. “Language Models Are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1877–1901.
- [7] Eoin M. Kenny et al. “Explaining Black-Box Classifiers Using Post-Hoc Explanations-by-Example: The Effect of Explanations and Error-Rates in XAI User Studies”. In: *Artificial Intelligence* 294 (May 2021), p. 103459. ISSN: 0004-3702. DOI: [10.1016/j.artint.2021.103459](https://doi.org/10.1016/j.artint.2021.103459).
- [8] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ““Why Should I Trust You?”: Explaining the Predictions of Any Classifier”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 1135–1144. ISBN: 978-1-4503-4232-2. DOI: [10/gfgrbd](https://doi.org/10/gfgrbd).
- [9] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. “Feature Visualization”. In: *Distill* 2.11 (Nov. 2017), e7. ISSN: 2476-0757. DOI: [10/gf2pnb](https://doi.org/10/gf2pnb).
- [10] Chaofan Chen et al. “This Looks Like That: Deep Learning for Interpretable Image Recognition”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.
- [11] Karol Hausman et al. “Learning an Embedding Space for Transferable Robot Skills”. In: *International Conference on Learning Representations*. Feb. 2018.
- [12] Chelsea Finn, Kelvin Xu, and Sergey Levine. “Probabilistic Model-Agnostic Meta-Learning”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS’18. Red Hook, NY, USA: Curran Associates Inc., Dec. 2018, pp. 9537–9548.
- [13] Jean Kaddour, Steindor Saemundsson, and Marc Deisenroth (he/him). “Probabilistic Active Meta-Learning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 20813–20822.
- [14] DeepMind ELLIS UCL CSML Seminar Series. *Marta Garnelo - Meta-Learning and Neural Processes*. Jan. 2021.

- [15] Bernhard Schölkopf et al. “Toward Causal Representation Learning”. In: *Proceedings of the IEEE* 109.5 (May 2021), pp. 612–634. ISSN: 1558-2256. DOI: [10/gjhqrh](https://doi.org/10/gjhqrh).
- [16] Brenden M. Lake et al. “Building Machines That Learn and Think like People”. In: *The Behavioral and Brain Sciences* 40 (Jan. 2017), e253. ISSN: 1469-1825. DOI: [10.1017/S0140525X16001837](https://doi.org/10.1017/S0140525X16001837).
- [17] Stevan Harnad. “The Symbol Grounding Problem”. In: *Physica D: Nonlinear Phenomena* 42.1 (June 1990), pp. 335–346. ISSN: 0167-2789. DOI: [10/c7bhzt](https://doi.org/10/c7bhzt).
- [18] John E. Laird. *The Soar Cognitive Architecture*. 2018. DOI: [10.7551/mitpress/7688.001.0001](https://doi.org/10.7551/mitpress/7688.001.0001).
- [19] Christian Lebiere. “Act-R”. In: *Proceedings of the Twenty-Fourth Annual Conference of the Cognitive Science Society*. 2019. DOI: [10.4324/9781315782379-4](https://doi.org/10.4324/9781315782379-4).
- [20] Paul Rosenbloom, Abram Demski, and Volkan Ustun. “The Sigma Cognitive Architecture and System: Towards Functionally Elegant Grand Unification”. In: *Journal of Artificial General Intelligence* 7 (Jan. 2016). DOI: [10/gd3zrf](https://doi.org/10/gd3zrf).
- [21] Frank van Harmelen and Annette ten Teije. “A Boxology of Design Patterns for Hybrid Learning and Reasoning Systems”. In: *Journal of Web Engineering* 18.1 (Jan. 2019), pp. 97–124. ISSN: 1540-9589, 1544-5976. DOI: [10.13052/jwe1540-9589.18133](https://doi.org/10.13052/jwe1540-9589.18133).
- [22] Tom Gruber. “Ontology”. In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 1963–1965. ISBN: 978-0-387-39940-9. DOI: [10.1007/978-0-387-39940-9_1318](https://doi.org/10.1007/978-0-387-39940-9_1318).
- [23] Markus Waibel et al. “RoboEarth”. In: *IEEE Robotics Automation Magazine* 18.2 (June 2011), pp. 69–82. ISSN: 1558-223X. DOI: [10.1109/MRA.2011.941632](https://doi.org/10.1109/MRA.2011.941632).
- [24] Guillaume Sarthou, Aurélie Clodic, and Rachid Alami. “Ontologenius : A Long-Term Semantic Memory for Robotic Agents”. In: (Oct. 2019).
- [25] R. Alami et al. “An Architecture for Autonomy”. In: *The International Journal of Robotics Research* 17.4 (Apr. 1998), pp. 315–337. ISSN: 0278-3649. DOI: [10/b2ts55](https://doi.org/10/b2ts55).
- [26] Mehdi Khamassi et al. “Integration of Action, Joint Action and Learning in Robot Cognitive Architectures”. In: *Intellectica - La revue de l'Association pour la Recherche sur les sciences de la Cognition (ARCo)* 2016/1.65 (June 2016), pp. 169–203. DOI: [10/gnhc4q](https://doi.org/10/gnhc4q).
- [27] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning and Acting*. Cambridge: Cambridge University Press, 2016. ISBN: 978-1-107-03727-4. DOI: [10.1017/CB09781139583923](https://doi.org/10.1017/CB09781139583923).
- [28] Sebastian Schneider and Franz Kummert. “Comparing Robot and Human Guided Personalization: Adaptive Exercise Robots Are Perceived as More Competent and Trustworthy”. In: *International Journal of Social Robotics* 13.2 (Apr. 2021), pp. 169–185. ISSN: 1875-4805. DOI: [10.1007/s12369-020-00629-w](https://doi.org/10.1007/s12369-020-00629-w).
- [29] Thibaut Munzer, Marc Toussaint, and Manuel Lopes. “Preference Learning on the Execution of Collaborative Human-Robot Tasks”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. May 2017, pp. 879–885. DOI: [10.1109/ICRA.2017.7989108](https://doi.org/10.1109/ICRA.2017.7989108).
- [30] Nils Wilde, Dana Kulić, and Stephen L. Smith. “Learning User Preferences in Robot Motion Planning Through Interaction”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. May 2018, pp. 619–626. DOI: [10.1109/ICRA.2018.8460586](https://doi.org/10.1109/ICRA.2018.8460586).

- [31] Giampiero Salvi et al. “Language Bootstrapping: Learning Word Meanings From Perception–Action Association”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42.3 (June 2012), pp. 660–671. ISSN: 1941-0492. DOI: [10.1109/TSMCB.2011.2172420](https://doi.org/10.1109/TSMCB.2011.2172420).
- [32] Victor Paleologue. *Teaching Robots Behaviors Using Spoken Language in Rich and Open Scenarios*. Tech. rep. Sorbonne Université, Dec. 2019.
- [33] Maxwell Forbes et al. “Robot Programming by Demonstration with Situated Spatial Language Understanding”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. May 2015, pp. 2014–2020. DOI: [10.1109/ICRA.2015.7139462](https://doi.org/10.1109/ICRA.2015.7139462).
- [34] Thibaut Munzer, Marc Toussaint, and Manuel Lopes. “Efficient Behavior Learning in Human–Robot Collaboration”. In: *Autonomous Robots* 42.5 (June 2018), pp. 1103–1115. ISSN: 1573-7527. DOI: [10.1007/s10514-017-9674-5](https://doi.org/10.1007/s10514-017-9674-5).
- [35] Marc Toussaint et al. “Relational Activity Processes for Modeling Concurrent Cooperation”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. May 2016, pp. 5505–5511. DOI: [10.1109/ICRA.2016.7487765](https://doi.org/10.1109/ICRA.2016.7487765).
- [36] S. Lauria et al. “Training Personal Robots Using Natural Language Instruction”. In: *IEEE Intelligent Systems* 16.5 (Sept. 2001), pp. 38–45. ISSN: 1941-1294. DOI: [10.1109/MIS.2001.956080](https://doi.org/10.1109/MIS.2001.956080).
- [37] Monica N. Nicolescu and Maja J. Mataric. “Natural Methods for Robot Task Learning: Instructive Demonstrations, Generalization and Practice”. In: *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*. AAMAS '03. New York, NY, USA: Association for Computing Machinery, July 2003, pp. 241–248. ISBN: 978-1-58113-683-8. DOI: [10.1145/860575.860614](https://doi.org/10.1145/860575.860614).
- [38] Paul E. Rybski et al. “Interactive Robot Task Training through Dialog and Demonstration”. In: *HRI 2007 - Proceedings of the 2007 ACM/IEEE Conference on Human-Robot Interaction - Robot as Team Member*. 2007. ISBN: 1-59593-617-3. DOI: [10.1145/1228716.1228724](https://doi.org/10.1145/1228716.1228724).
- [39] Paul E. Rybski et al. “Using Dialog and Human Observations to Dictate Tasks to a Learning Robot Assistant”. In: *Intelligent Service Robotics* 1.2 (Apr. 2008), pp. 159–167. ISSN: 1861-2784. DOI: [10.1007/s11370-008-0016-5](https://doi.org/10.1007/s11370-008-0016-5).
- [40] Alfredo Weitzenfeld, Carlos Ramos, and Peter Dominey. “Coaching Robots to Play Soccer via Spoken-Language”. In: vol. 5399. July 2008, pp. 379–390. DOI: [10.1007/978-3-642-02921-9_33](https://doi.org/10.1007/978-3-642-02921-9_33).
- [41] S. Lallee et al. “Human-Robot Cooperation Based on Interaction Learning”. In: *From Motor Learning to Interaction Learning in Robots*. Ed. by Olivier Sigaud and Jan Peters. Studies in Computational Intelligence. Berlin, Heidelberg: Springer, 2010, pp. 491–536. ISBN: 978-3-642-05181-4. DOI: [10.1007/978-3-642-05181-4_21](https://doi.org/10.1007/978-3-642-05181-4_21).
- [42] Stephane Lallee et al. “Towards a Platform-Independent Cooperative Human Robot Interaction System: III An Architecture for Learning and Executing Actions and Shared Plans”. In: *IEEE Transactions on Autonomous Mental Development* 4.3 (Sept. 2012), pp. 239–253. ISSN: 1943-0612. DOI: [10.1109/TAMD.2012.2199754](https://doi.org/10.1109/TAMD.2012.2199754).
- [43] Jonathan Grizou, Manuel Lopes, and Pierre-Yves Oudeyer. “Robot Learning from Unlabelled Teaching Signals”. In: *HRI 2014 Pioneers Workshop*. Mar. 2014.
- [44] Lanbo She et al. “Teaching Robots New Actions through Natural Language Instructions”. In: *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*. Aug. 2014, pp. 868–873. DOI: [10.1109/ROMAN.2014.6926362](https://doi.org/10.1109/ROMAN.2014.6926362).

- [45] Marwin Sorce et al. “Proof of Concept for a User-Centered System for Sharing Cooperative Plan Knowledge over Extended Periods and Crew Changes in Space-Flight Operations”. In: *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. Aug. 2015, pp. 776–783. DOI: [10.1109/ROMAN.2015.7333565](https://doi.org/10.1109/ROMAN.2015.7333565).
- [46] Shiwali Mohan. “From Verbs to Tasks: An Integrated Account of Learning Tasks from Situated Interactive Instruction”. PhD thesis. 2015. ISBN: 9781321727371.
- [47] Anahita Mohseni-Kabir et al. “Interactive Hierarchical Task Learning from a Single Demonstration”. In: *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*. HRI '15. New York, NY, USA: Association for Computing Machinery, Mar. 2015, pp. 205–212. ISBN: 978-1-4503-2883-8. DOI: [10/ghbrhj](https://doi.org/10/ghbrhj).
- [48] Maxime Petit and Yiannis Demiris. “Hierarchical Action Learning by Instruction through Interactive Grounding of Body Parts and Proto-Actions”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. May 2016, pp. 3375–3382. DOI: [10.1109/ICRA.2016.7487513](https://doi.org/10.1109/ICRA.2016.7487513).
- [49] Gavin Suddrey et al. “Teaching Robots Generalizable Hierarchical Tasks Through Natural Language Instruction”. In: *IEEE Robotics and Automation Letters* 2.1 (Jan. 2017), pp. 201–208. ISSN: 2377-3766. DOI: [10.1109/LRA.2016.2588584](https://doi.org/10.1109/LRA.2016.2588584).
- [50] Chris Paxton et al. “CoSTAR: Instructing Collaborative Robots with Behavior Trees and Vision”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. 2017. ISBN: 978-1-5090-4633-1. DOI: [10.1109/ICRA.2017.7989070](https://doi.org/10.1109/ICRA.2017.7989070). arXiv: [1611.06145](https://arxiv.org/abs/1611.06145).
- [51] Moritz Tenorth and Michael Beetz. “KnowRob: A Knowledge Processing Infrastructure for Cognition-Enabled Robots”. In: *International Journal of Robotics Research* (2013). ISSN: 02783649. DOI: [10.1177/0278364913481635](https://doi.org/10.1177/0278364913481635).
- [52] Michael Beetz et al. “Know Rob 2.0 - A 2nd Generation Knowledge Processing Framework for Cognition-Enabled Robotic Agents”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. 2018. ISBN: 978-1-5386-3081-5. DOI: [10.1109/ICRA.2018.8460964](https://doi.org/10.1109/ICRA.2018.8460964).
- [53] Clement Moulin-Frier et al. “DAC-h3: A Proactive Robot Cognitive Architecture to Acquire and Express Knowledge about the World and the Self”. In: *IEEE Transactions on Cognitive and Developmental Systems* (2018). ISSN: 23798939. DOI: [10.1109/TCDS.2017.2754143](https://doi.org/10.1109/TCDS.2017.2754143). arXiv: [1706.03661](https://arxiv.org/abs/1706.03661).
- [54] Anahita Mohseni-Kabir et al. “Simultaneous Learning of Hierarchy and Primitives for Complex Robot Tasks”. In: *Autonomous Robots* 43.4 (Apr. 2019), pp. 859–874. ISSN: 0929-5593. DOI: [10.1007/s10514-018-9749-y](https://doi.org/10.1007/s10514-018-9749-y).
- [55] Shiwali Mohan et al. “Building Jarvis - A Learner-Aware Conversational Trainer”. In: *CEUR Workshop Proceedings*. Vol. 2327. CEUR-WS, 2019.
- [56] Alexandre Angleraud, Quentin Houbre, and Roel Pieters. “Teaching Semantics and Skills for Human-Robot Collaboration”. In: *Paladyn, Journal of Behavioral Robotics* 10.1 (Jan. 2019), pp. 318–329. ISSN: 2081-4836. DOI: [10.1515/pjbr-2019-0025](https://doi.org/10.1515/pjbr-2019-0025).
- [57] Aaron Mininger and John Laird. “Interactively Learning a Blend of Goal-Based and Procedural Tasks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 2018). ISSN: 2374-3468.

- [58] Matthias Scheutz et al. “Spoken Instruction-Based One-Shot Object and Action Learning in a Cognitive Robotic Architecture”. In: *AAMAS*. 2017.
- [59] Matthias Scheutz et al. “Recursive Spoken Instruction-Based One-Shot Object and Action Learning”. In: *IJCAI*. 2018. DOI: [10.24963/ijcai.2018/752](https://doi.org/10.24963/ijcai.2018/752).
- [60] Francisco Martín et al. “Evolution of a Cognitive Architecture for Social Robots: Integrating Behaviors and Symbolic Knowledge”. In: *Applied Sciences* 10.17 (Jan. 2020), p. 6067. DOI: [10.3390/app10176067](https://doi.org/10.3390/app10176067).
- [61] François Hélon et al. “Cognitive Architecture for Intuitive and Interactive Task Learning in Industrial Collaborative Robotics”. In: *The 5th International Conference on Robotics, Control and Automation, ICRCA 2021*. 2021. ISBN: 978-1-4503-8748-4.
- [62] Mihalis Yannakakis. “Hierarchical State Machines”. In: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2000. ISBN: 3-540-67823-9. DOI: [10.1007/3-540-44929-9_24](https://doi.org/10.1007/3-540-44929-9_24).
- [63] Tyler Frasca et al. “Enabling Fast Instruction-Based Modification of Learned Robot Skills”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.7 (May 2021), pp. 6075–6083. ISSN: 2374-3468.
- [64] Matthias Scheutz et al. “An Overview of the Distributed Integrated Cognition Affect and Reflection DIARC Architecture”. In: *Cognitive Architectures*. Ed. by Maria Isabel Aldinhas Ferreira, João Silva Sequeira, and Rodrigo Ventura. Intelligent Systems, Control and Automation: Science and Engineering. Cham: Springer International Publishing, 2019, pp. 165–193. ISBN: 978-3-319-97550-4. DOI: [10.1007/978-3-319-97550-4_11](https://doi.org/10.1007/978-3-319-97550-4_11).
- [65] Shiwali Mohan et al. “Acquiring Grounded Representation of Words with Situated Interactive Instruction”. In: *Advances in Cognitive Systems 2* (2012), pp. 113–130.
- [66] Guglielmo Gemignani, E. Bastianelli, and D. Nardi. “Teaching Robots Parametrized Executable Plans Through Spoken Interaction”. In: *AAMAS*. 2015.
- [67] V. A. Ziparo et al. “Petri Net Plans: A Formal Model for Representation and Execution of Multi-Robot Plans”. In: *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*. AAMAS ’08. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 2008, pp. 79–86. ISBN: 978-0-9817381-0-9.
- [68] Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. “CRAM - A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments”. In: *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*. 2010. ISBN: 978-1-4244-6675-7. DOI: [10.1109/IROS.2010.5650146](https://doi.org/10.1109/IROS.2010.5650146).
- [69] Michael a Covington, Donald Nute, and André Vellino. “Prolog Programming in Depth”. In: *Knowledge Creation Diffusion Utilization* (1995).
- [70] Francois Felix Ingrand et al. “PRS: A High Level Supervision and Control Language for Autonomous Mobile Robots”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. 1996. DOI: [10.1109/robot.1996.503571](https://doi.org/10.1109/robot.1996.503571).
- [71] R. Alami et al. “An Architecture for Autonomy”. In: *The International Journal of Robotics Research* 17.4 (Apr. 1998), pp. 315–337. ISSN: 0278-3649. DOI: [10/b2ts55](https://doi.org/10/b2ts55).
- [72] Sunandita Patra et al. “Integrating Acting, Planning, and Learning in Hierarchical Operational Models”. In: *International Confer-*

- ence on Automated Planning and Scheduling (ICAPS). Oct. 2020.
- [73] Kutluhan Erol, James Hendler, and Dana Nau. “HTN Planning: Complexity and Expressivity”. In: *Proceedings of the National Conference on Artificial Intelligence 2* (May 1994).
- [74] Pascal Bercher, Ron Alford, and Daniel Höller. “A Survey on Hierarchical Planning – One Abstract Idea, Many Concrete Realizations”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. Macao, China: International Joint Conferences on Artificial Intelligence Organization, Aug. 2019, pp. 6267–6275. ISBN: 978-0-9992411-4-1. DOI: [10/gnhf2k](https://doi.org/10/gnhf2k).
- [75] Anahita Mohseni-Kabir et al. “SLHAP: Simultaneous Learning of Hierarchy and Primitives”. In: *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. HRI ’17. New York, NY, USA: Association for Computing Machinery, Mar. 2017, p. 412. ISBN: 978-1-4503-4885-0. DOI: [10.1145/3029798.3036641](https://doi.org/10.1145/3029798.3036641).
- [76] Monica N. Nicolescu and Maja J. Mataric. “A Hierarchical Architecture for Behavior-Based Robots”. In: *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1*. AAMAS ’02. New York, NY, USA: Association for Computing Machinery, July 2002, pp. 227–233. ISBN: 978-1-58113-480-3. DOI: [10/dp3zww](https://doi.org/10/dp3zww).
- [77] Michele Colledanchise and Petter Ögren. “Behavior Trees in Robotics and AI: An Introduction”. In: *arXiv* (Aug. 2017). DOI: [10.1201/9780429489105](https://doi.org/10.1201/9780429489105). arXiv: [1709.00084](https://arxiv.org/abs/1709.00084).
- [78] Francesco Rovida, Bjarne Grossmann, and Volker Krüger. “Extended Behavior Trees for Quick Definition of Flexible Robotic Tasks”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2017, pp. 6793–6800. DOI: [10.1109/IROS.2017.8206598](https://doi.org/10.1109/IROS.2017.8206598).
- [79] Xenija Neufeld, Sanaz Mostaghim, and Sandy Brand. “A Hybrid Approach to Planning and Execution in Dynamic Environments Through Hierarchical Task Networks and Behavior Trees”. In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment 14.1* (Sept. 2018), pp. 201–207. ISSN: 2334-0924.
- [80] Raja Chatila et al. “Toward Self-Aware Robots”. In: *Frontiers in Robotics and AI 5* (Aug. 2018). ISSN: 2296-9144. DOI: [10.3389/frobt.2018.00088](https://doi.org/10.3389/frobt.2018.00088).
- [81] J.E. Allen, C.I. Guinn, and E. Horvitz. “Mixed-Initiative Interaction”. In: *IEEE Intelligent Systems and their Applications 14.5* (Sept. 1999), pp. 14–23. ISSN: 2374-9423. DOI: [10/ch7j9k](https://doi.org/10/ch7j9k).
- [82] Charles Rich, Candace L. Sidner, and Neal Lesh. “COLLAGEN: Applying Collaborative Discourse Theory to Human-Computer Interaction”. In: *AI Magazine* (2001). ISSN: 07384602.
- [83] Sonia Chernova and Andrea Thomaz. “Robot Learning from Human Teachers”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning 8* (Apr. 2014), pp. 1–121. DOI: [10/gf7bxx](https://doi.org/10/gf7bxx).
- [84] C. Crangle and P. Suppes. “Language and Learning for Robots”. In: 1994. DOI: [10.2307/416124](https://doi.org/10.2307/416124).
- [85] Scott B. Huffman and John E. Laird. “Flexibly Instructable Agents”. In: *Journal of Artificial Intelligence Research 3.1* (Nov. 1995), pp. 271–324. ISSN: 1076-9757.
- [86] Jonathan Grizou, Manuel Lopes, and Pierre-Yves Oudeyer. “Robot Learning Simultaneously a Task and How to Interpret Human Instructions”. In: *2013 IEEE Third Joint International Conference on Development and Learning and Epigenetic Robotics (ICDL)*.

- Aug. 2013, pp. 1–8. DOI: [10.1109/DevLrn.2013.6652523](https://doi.org/10.1109/DevLrn.2013.6652523).
- [87] Aaron Mininger. “Expanding Task Diversity in Explanation-Based Interactive Task Learning”. In: (2021). DOI: [10.7302/1453](https://doi.org/10.7302/1453).

Chapter 3: Design of a cognitive architecture for Industry 4.0

- [1] Aldo Gangemi et al. “Sweetening Ontologies with DOLCE”. In: *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*. Ed. by G. Goos et al. Vol. 2473. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 166–181. ISBN: 978-3-540-44268-4 978-3-540-45810-4. DOI: [10.1007/3-540-45810-7_18](https://doi.org/10.1007/3-540-45810-7_18).
- [2] Victor Paleologue. *Teaching Robots Behaviors Using Spoken Language in Rich and Open Scenarios*. Tech. rep. Sorbonne Université, Dec. 2019.
- [3] Moritz Tenorth and Michael Beetz. “KnowRob: A Knowledge Processing Infrastructure for Cognition-Enabled Robots”. In: *International Journal of Robotics Research* (2013). ISSN: 02783649. DOI: [10.1177/0278364913481635](https://doi.org/10.1177/0278364913481635).
- [4] Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. “CRAM - A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments”. In: *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*. 2010. ISBN: 978-1-4244-6675-7. DOI: [10.1109/IROS.2010.5650146](https://doi.org/10.1109/IROS.2010.5650146).
- [5] Mattias Appelgren and Alex Lascarides. “Interactive Task Learning via Embodied Corrective Feedback”. In: *Autonomous Agents and Multi-Agent Systems* 34.2 (Sept. 2020), p. 54. ISSN: 1573-7454. DOI: [10/gmwdkm](https://doi.org/10/gmwdkm).
- [6] Ronald A. Rensink. “Seeing, Sensing, and Scrutinizing”. In: *Vision Research*. 2000. DOI: [10.1016/S0042-6989\(00\)00003-1](https://doi.org/10.1016/S0042-6989(00)00003-1).
- [7] Francisco Cruz et al. “Training Agents With Interactive Reinforcement Learning and Contextual Affordances”. In: *IEEE Transactions on Cognitive and Developmental Systems* 8.4 (Dec. 2016), pp. 271–284. ISSN: 2379-8920, 2379-8939. DOI: [10/f9hnd6](https://doi.org/10/f9hnd6).
- [8] Mihai Andries et al. “Affordance Equivalences in Robotics: A Formalism”. In: *Frontiers in Neurorobotics* 12.JUN (June 2018). ISSN: 16625218. DOI: [10.3389/fnbot.2018.00026](https://doi.org/10.3389/fnbot.2018.00026).
- [9] Michele Colledanchise and Petter Ögren. “Behavior Trees in Robotics and AI: An Introduction”. In: *arXiv* (Aug. 2017). DOI: [10.1201/9780429489105](https://doi.org/10.1201/9780429489105). arXiv: [1709.00084](https://arxiv.org/abs/1709.00084).
- [10] Damian Isla. “Gdc 2005 Proceeding: Handling Complexity in the Halo 2 Ai”. In: *Retrieved October 21 (2005)*, p. 2009.
- [11] Vincent Berenz and Stefan Schaal. “The Playful Software Platform: Reactive Programming for Orchestrating Robotic Behavior”. In: *IEEE Robotics Automation Magazine* 25.3 (Sept. 2018), pp. 49–60. ISSN: 1558-223X. DOI: [10/ggzjwf](https://doi.org/10/ggzjwf).
- [12] Vincent Berenz et al. “Learning Sensory-Motor Associations from Demonstration”. In: *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. Aug. 2020, pp. 1081–1087. DOI: [10/gnfpmr](https://doi.org/10/gnfpmr).
- [13] Francesco Roviida, Bjarne Grossmann, and Volker Krüger. “Extended Behavior Trees for Quick Definition of Flexible Robotic Tasks”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2017, pp. 6793–6800. DOI: [10.1109/IROS.2017.8206598](https://doi.org/10.1109/IROS.2017.8206598).
- [14] Evgenii Safronov et al. “Asynchronous Behavior Trees with Memory Aimed at Aerial Vehicles with Redundancy in Flight Controller”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nov. 2019, pp. 3113–3118. DOI: [10/ggv9zp](https://doi.org/10/ggv9zp).

- [15] Chris Paxton et al. “CoSTAR: Instructing Collaborative Robots with Behavior Trees and Vision”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. 2017. ISBN: 978-1-5090-4633-1. DOI: [10 . 1109 / ICRA . 2017 . 7989070](https://doi.org/10.1109/ICRA.2017.7989070). arXiv: [1611.06145](https://arxiv.org/abs/1611.06145).
- [16] Francisco Martín et al. “Evolution of a Cognitive Architecture for Social Robots: Integrating Behaviors and Symbolic Knowledge”. In: *Applied Sciences* 10.17 (Jan. 2020), p. 6067. DOI: [10.3390/app10176067](https://doi.org/10.3390/app10176067).
- [17] Razan Ghzouli et al. “Behavior Trees in Action: A Study of Robotics Applications”. In: *Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering*. SLE 2020. New York, NY, USA: Association for Computing Machinery, Nov. 2020, pp. 196–209. ISBN: 978-1-4503-8176-5. DOI: [10/gh2qd3](https://doi.org/10/gh2qd3).
- [18] Matteo Iovino et al. “A Survey of Behavior Trees in Robotics and AI”. In: (May 2020). arXiv: [2005.05842](https://arxiv.org/abs/2005.05842).
- [19] Michele Colledanchise and Lorenzo Natale. “On the Implementation of Behavior Trees in Robotics”. In: *IEEE Robotics and Automation Letters* 6.3 (July 2021), pp. 5929–5936. ISSN: 2377-3766. DOI: [10.1109/LRA.2021.3087442](https://doi.org/10.1109/LRA.2021.3087442).
- [20] Zhongxuan Cai et al. “BT Expansion: A Sound and Complete Algorithm for Behavior Planning of Intelligent Robots with Behavior Trees”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.7 (May 2021), pp. 6058–6065. ISSN: 2374-3468.
- [21] Sinno Jialin Pan and Qiang Yang. “A Survey on Transfer Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (Oct. 2010), pp. 1345–1359. ISSN: 1041-4347. DOI: [10/bc4vws](https://doi.org/10/bc4vws).
- [22] Fuzhen Zhuang et al. “A Comprehensive Survey on Transfer Learning”. In: *Proceedings of the IEEE* 109.1 (Jan. 2021), pp. 43–76. ISSN: 0018-9219, 1558-2256. DOI: [10 / ghcsqr](https://doi.org/10/ghcsqr).
- [23] Gao Huang et al. “Densely Connected Convolutional Networks”. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Vol. 2017-Janua. Aug. 2017, pp. 2261–2269. ISBN: 978-1-5386-0457-1. DOI: [10 . 1109 / CVPR . 2017 . 243](https://doi.org/10.1109/CVPR.2017.243). arXiv: [1608.06993](https://arxiv.org/abs/1608.06993).
- [24] Mark Sandler et al. “MobileNetV2: Inverted Residuals and Linear Bottlenecks”. In: (Jan. 2018). arXiv: [1801.04381](https://arxiv.org/abs/1801.04381).

Chapter 4: Complementary ML approaches for IRL on planar grasping use cases

- [1] Andy Zeng et al. *Learning Synergies between Pushing and Grasping with Self-supervised Deep Reinforcement Learning*. Tech. rep. 2018, p. 7. URL: <http://vpg.cs.princeton.edu>.
- [2] Sonia Chernova and Andrea Thomaz. “Robot Learning from Human Teachers”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8 (Apr. 2014), pp. 1–121. DOI: [10/gf7bxx](https://doi.org/10/gf7bxx).
- [3] Fionn Murtagh and Pedro Contreras. “Algorithms for Hierarchical Clustering: An Overview, II”. In: *WIREs Data Mining and Knowledge Discovery* 7.6 (2017), e1219. ISSN: 1942-4795. DOI: [10/gctpcdc](https://doi.org/10/gctpcdc).
- [4] Pierre Baldi. “Autoencoders, Unsupervised Learning, and Deep Architectures”. In: (), p. 14.
- [5] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: [10/ggbw6](https://doi.org/10/ggbw6).
- [6] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction, Second Edition*. 2018. ISBN: 0-262-19398-1. DOI: [10.1016/S0140-6736\(51\)92942-X](https://doi.org/10.1016/S0140-6736(51)92942-X). arXiv: [1603.02199](https://arxiv.org/abs/1603.02199).
- [7] Sinno Jialin Pan and Qiang Yang. “A Survey on Transfer Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (Oct. 2010), pp. 1345–1359. ISSN: 1041-4347. DOI: [10/bc4vws](https://doi.org/10/bc4vws).
- [8] Fuzhen Zhuang et al. “A Comprehensive Survey on Transfer Learning”. In: *Proceedings of the IEEE* 109.1 (Jan. 2021), pp. 43–76. ISSN: 0018-9219, 1558-2256. DOI: [10/ghcsqr](https://doi.org/10/ghcsqr).
- [9] Berk Calli, Martijn Wisse, and Pieter Jonker. “Grasping of unknown objects via curvature maximization using active vision”. In: *IEEE International Conference on Intelligent Robots and Systems* (2011), pp. 995–1001. DOI: [10.1109/IRoS.2011.6048739](https://doi.org/10.1109/IRoS.2011.6048739).
- [10] Yun Jiang, Stephen Moseson, and Ashutosh Saxena. “Efficient grasping from RGBD images: Learning using a new rectangle representation”. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2011), pp. 3304–3311. ISSN: 10504729. DOI: [10.1109/ICRA.2011.5980145](https://doi.org/10.1109/ICRA.2011.5980145).
- [11] Jeffrey Mahler et al. “Dex-Net 2.0: Deep learning to plan Robust grasps with synthetic point clouds and analytic grasp metrics”. In: *Robotics: Science and Systems* 13 (2017). ISSN: 2330765X. DOI: [10.15607/rss.2017.xiii.058](https://doi.org/10.15607/rss.2017.xiii.058).
- [12] Shehan Caldera, Alexander Rassau, and Douglas Chai. “Review of Deep Learning Methods in Robotic Grasp Detection”. In: *Multimodal Technologies and Interaction* 2.3 (2018), p. 57. DOI: [10.3390/mti2030057](https://doi.org/10.3390/mti2030057).
- [13] Joseph Redmon and Anelia Angelova. “Real-time grasp detection using convolutional neural networks”. In: *Proceedings - IEEE International Conference on Robotics and Automation* 2015-June.June (2015), pp. 1316–1322. ISSN: 10504729. DOI: [10.1109/ICRA.2015.7139361](https://doi.org/10.1109/ICRA.2015.7139361).

- [14] Sulabh Kumra and Christopher Kanan. “Robotic grasp detection using deep convolutional neural networks”. In: *IEEE International Conference on Intelligent Robots and Systems 2017-Septe* (2017), pp. 769–776. ISSN: 21530866. DOI: [10.1109/IRoS.2017.8202237](https://doi.org/10.1109/IRoS.2017.8202237).
- [15] Fu Jen Chu, Ruinian Xu, and Patricio A. Vela. “Real-world multiobject, multigrasp detection”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3355–3362. ISSN: 23773766. DOI: [10.1109/LRA.2018.2852777](https://doi.org/10.1109/LRA.2018.2852777).
- [16] Amaury Depierre, Emmanuel Dellandrea, and Liming Chen. “Jacquard: A Large Scale Dataset for Robotic Grasp Detection”. In: *IEEE International Conference on Intelligent Robots and Systems* (2018), pp. 3511–3516. ISSN: 21530866. DOI: [10.1109/IRoS.2018.8593950](https://doi.org/10.1109/IRoS.2018.8593950).
- [17] Jeffrey Mahler et al. “Dex-Net 2.0: Deep learning to plan Robust grasps with synthetic point clouds and analytic grasp metrics”. In: *Robotics: Science and Systems* 13 (2017). ISSN: 2330765X.
- [18] Ulrich Viereck et al. “Learning a visuomotor controller for real world robotic grasping using simulated depth images”. In: *CoRL* (2017), pp. 1–10. URL: <http://arxiv.org/abs/1706.04652>.
- [19] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *International Conference on Learning Representations (ICLR)* (2015), pp. 1–14.
- [20] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: (2015). ISSN: 1664-1078. DOI: [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90). URL: <http://arxiv.org/abs/1512.03385>.
- [21] Gao Huang et al. “Densely Connected Convolutional Networks”. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Vol. 2017-Janua. Aug. 2017, pp. 2261–2269. ISBN: 978-1-5386-0457-1. DOI: [10.1109/CVPR.2017.243](https://doi.org/10.1109/CVPR.2017.243). arXiv: [1608.06993](https://arxiv.org/abs/1608.06993).
- [22] Russakovsky Olga et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *IJCV* (2015).
- [23] Tsung Yi Lin et al. “Microsoft COCO: Common objects in context”. In: *Lecture Notes in Computer Science (including sub-series Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8693 LNCS.PART 5 (2014), pp. 740–755. ISSN: 16113349. DOI: [10.1007/978-3-319-10602-1_48](https://doi.org/10.1007/978-3-319-10602-1_48).
- [24] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: (2015), pp. 1–14.
- [25] Marwan Qaid Mohammed, Kwek Lee Chung, and Chua Shing Chyi. “Review of Deep Reinforcement Learning-Based Object Grasping: Techniques, Open Challenges, and Recommendations”. In: *IEEE Access* 8 (2020), pp. 178450–178481. ISSN: 2169-3536. DOI: [10/gj5vd5](https://doi.org/10/gj5vd5).
- [26] Jeffrey Mahler and Ken Goldberg. “Learning Deep Policies for Robot Bin Picking by Simulating Robust Grasping Sequences”. In: *Proceedings of the 1st Annual Conference on Robot Learning*. PMLR, Oct. 2017, pp. 515–524.
- [27] Volodymyr Mnih et al. “Human-Level Control through Deep Reinforcement Learning”. In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. ISSN: 1476-4687. DOI: [10/gc3h75](https://doi.org/10/gc3h75).
- [28] François Hélienon et al. “Learning Prohibited and Authorised Grasping Locations from a Few Demonstrations”. In: *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. Aug. 2020, pp. 1094–1100. DOI: [10/gk8djs](https://doi.org/10/gk8djs).

- [29] Ian Lenz, Honglak Lee, and Ashutosh Saxena. “Deep learning for detecting robotic grasps”. In: *International Journal of Robotics Research* 34.4-5 (2015), pp. 705–724. ISSN: 17413176. DOI: [10.1177/0278364914549607](https://doi.org/10.1177/0278364914549607).
- [30] Edward Johns, Stefan Leutenegger, and Andrew J. Davison. “Deep learning a grasp function for grasping under gripper pose uncertainty”. In: *IEEE International Conference on Intelligent Robots and Systems* 2016-Novem (2016), pp. 4461–4468. ISSN: 21530866. DOI: [10.1109/IRoS.2016.7759657](https://doi.org/10.1109/IRoS.2016.7759657).
- [31] Sulabh Kumra and Christopher Kanan. “Robotic grasp detection using deep convolutional neural networks”. In: *IEEE International Conference on Intelligent Robots and Systems* 2017-Septe (2017), pp. 769–776. ISSN: 21530866. DOI: [10.1109/IRoS.2017.8202237](https://doi.org/10.1109/IRoS.2017.8202237).
- [32] Lerrel Pinto and Abhinav Gupta. “Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours”. In: *Proceedings - IEEE International Conference on Robotics and Automation* 2016-June (2016), pp. 3406–3413. ISSN: 10504729. DOI: [10.1109/ICRA.2016.7487517](https://doi.org/10.1109/ICRA.2016.7487517).
- [33] Renaud Detry, Jeremie Papon, and Larry Matthies. “Task-oriented grasping with semantic and geometric scene understanding”. In: *IEEE International Conference on Intelligent Robots and Systems*. Vol. 2017-Septe. IEEE, Sept. 2017, pp. 3266–3273. ISBN: 9781538626825. DOI: [10.1109/IRoS.2017.8206162](https://doi.org/10.1109/IRoS.2017.8206162). URL: <http://ieeexplore.ieee.org/document/8206162/>.
- [34] Kuan Fang et al. “Learning Task-Oriented Grasping for Tool Manipulation from Simulated Self-Supervision”. In: (June 2018). URL: <http://arxiv.org/abs/1806.09266>.
- [35] Rika Antonova et al. “Global Search with Bernoulli Alternation Kernel for Task-oriented Grasping Informed by Simulation”. In: CoRL (2018). URL: <http://arxiv.org/abs/1810.04438>.
- [36] Austin Myers et al. “Affordance detection of tool parts from geometric features”. In: *Proceedings - IEEE International Conference on Robotics and Automation* 2015-June.June (2015), pp. 1374–1381. ISSN: 10504729. DOI: [10.1109/ICRA.2015.7139369](https://doi.org/10.1109/ICRA.2015.7139369).
- [37] Peter R. Florence, Lucas Manuelli, and Russ Tedrake. “Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation”. In: (June 2018). URL: <http://arxiv.org/abs/1806.08756>.
- [38] James Gibson. *The theory of affordances*. 1979, Chapter 8.
- [39] Raghvendra Jain and Tetsunari Inamura. “Learning of Tool Affordances for autonomous tool manipulation”. In: *2011 IEEE/SICE International Symposium on System Integration, SII 2011* December (2011), pp. 814–819. DOI: [10.1109/SII.2011.6147553](https://doi.org/10.1109/SII.2011.6147553).
- [40] Thanh Toan Do, Anh Nguyen, and Ian Reid. “AffordanceNet: An End-to-End Deep Learning Approach for Object Affordance Detection”. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2018), pp. 5882–5889. ISSN: 10504729. DOI: [10.1109/ICRA.2018.8460902](https://doi.org/10.1109/ICRA.2018.8460902).
- [41] Anh Nguyen et al. “Detecting object affordances with convolutional neural networks”. In: *IEEE International Conference on Intelligent Robots and Systems* 2016-Novem.October (2016), pp. 2765–2770. ISSN: 21530866. DOI: [10.1109/IRoS.2016.7759429](https://doi.org/10.1109/IRoS.2016.7759429).
- [42] Anh Nguyen et al. “Object-based affordances detection with Convolutional Neural Networks and dense Conditional Random Fields”. In: *IEEE International Conference on Intelligent Robots and Systems* 2017-Septe.September (2017), pp. 5908–

5915. ISSN: 21530866. DOI: [10.1109/IRoS.2017.8206484](https://doi.org/10.1109/IRoS.2017.8206484).
- [43] Pieter Van Molle et al. “Learning to Grasp from a Single Demonstration”. In: (2018). URL: <http://arxiv.org/abs/1806.03486>.
- [44] Gao Huang et al. “Densely connected convolutional networks”. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-Janua (2017), pp. 2261–2269. ISSN: 0022-4790. DOI: [10.1109/CVPR.2017.243](https://doi.org/10.1109/CVPR.2017.243).
- [45] Hang Zhao et al. “Loss Functions for Neural Networks for Image Processing”. In: (2015), pp. 1–11. URL: <http://arxiv.org/abs/1511.08861>.
- [46] Mohammad Safeea and Pedro Neto. “KUKA Sunrise Toolbox: Interfacing Collaborative Robots With MATLAB”. In: *IEEE Robotics & Automation Magazine* 26.1 (Mar. 2019), pp. 91–96. ISSN: 1070-9932. DOI: [10.1109/MRA.2018.2877776](https://doi.org/10.1109/MRA.2018.2877776). URL: <https://ieeexplore.ieee.org/document/8542757/>.
- [47] Peter R. Florence, Lucas Manuelli, and Russ Tedrake. “Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation”. In: (June 2018). URL: <http://arxiv.org/abs/1806.08756>.

Chapter 5: Learning under uncertainty

- [1] Chuan Guo et al. “On Calibration of Modern Neural Networks”. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML’17. Sydney, NSW, Australia: JMLR.org, Aug. 2017, pp. 1321–1330.
- [2] Saurabh Garg et al. “A Unified View of Label Shift Estimation”. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 3290–3300.
- [3] Zachary C. Lipton, Yu-Xiang Wang, and Alex Smola. “Detecting and Correcting for Label Shift with Black Box Predictors”. In: *arXiv:1802.03916 [cs, stat]* (July 2018). arXiv: [1802.03916 \[cs, stat\]](https://arxiv.org/abs/1802.03916).
- [4] Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. “Recent Advances in Open Set Recognition: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–1. ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: [10 / ghhqwk](https://doi.org/10.1109/tpami.2020.2988888). arXiv: [1811.08581](https://arxiv.org/abs/1811.08581).
- [5] Shibani Santurkar, Dimitris Tsipras, and Aleksander Madry. “BREEDS: Benchmarks for Subpopulation Shift”. In: *arXiv:2008.04859 [cs, stat]* (Aug. 2020). arXiv: [2008.04859 \[cs, stat\]](https://arxiv.org/abs/2008.04859).
- [6] Alexander Meinke and Matthias Hein. “Towards Neural Networks That Provably Know When They Don’t Know”. In: *International Conference on Learning Representations*. Sept. 2019.
- [7] Eyke Hüllermeier and Willem Waegeman. “Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods”. In: *Machine Learning* 110.3 (Mar. 2021), pp. 457–506. ISSN: 1573-0565. DOI: [10/gjpb3](https://doi.org/10.1007/s10994-021-0565-0).
- [8] S. Geman, E. Bienenstock, and R. Doursat. “Neural Networks and the Bias/Variance Dilemma”. In: *Neural Computation* (1992). DOI: [10/dz9dd3](https://doi.org/10.1162/neco.1992.4.2.265).
- [9] Pedro Domingos. “A Unified Bias-Variance Decomposition”. In: (), p. 22.
- [10] Dustin Tran et al. “Bayesian Layers: A Module for Neural Network Uncertainty”. In: *NeurIPS*. 2019.
- [11] Joshua V. Dillon et al. “TensorFlow Distributions”. In: *arXiv:1711.10604 [cs, stat]* (Nov. 2017). arXiv: [1711.10604 \[cs, stat\]](https://arxiv.org/abs/1711.10604).
- [12] Martín Abadi et al. *TensorFlow: Large-scale Machine Learning on Heterogeneous Systems*. 2015.
- [13] Fabian Pedregosa et al. “Scikit-Learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830. ISSN: 1533-7928.
- [14] Jeremiah Zhe Liu et al. “Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness”. In: *NeurIPS* (2020).
- [15] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: (2015). ISSN: 1664-1078. DOI: [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90). URL: <http://arxiv.org/abs/1512.03385>.
- [16] Alex Kendall and Yarin Gal. “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?” In: (), p. 11.
- [17] Alex Graves. “Practical Variational Inference for Neural Networks”. In: *Advances in Neural Information Processing Systems*. Vol. 24. Curran Associates, Inc., 2011.
- [18] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles”. In: *Proceedings of the 31st International Conference on Neural*

- Information Processing Systems*. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 6405–6416. ISBN: 978-1-5108-6096-4.
- [19] Yarin Gal and Zoubin Ghahramani. “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning Yarin”. In: 3982 (2016). ISSN: 0277786X.
- [20] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. DOI: [10.1109/ICAEES.2016.7888100](https://doi.org/10.1109/ICAEES.2016.7888100).
- [21] Alex Kendall and Yarin Gal. “What uncertainties do we need in Bayesian deep learning for computer vision?” In: *Advances in Neural Information Processing Systems 2017-Decem.Nips* (2017), pp. 5575–5585. ISSN: 10495258.
- [22] Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. “Variational Adversarial Active Learning”. In: (2019). URL: <http://arxiv.org/abs/1904.00370>.
- [23] Charles Corbière et al. “Addressing Failure Prediction by Learning Model Confidence”. In: *NeurIPS* (2019), pp. 1–11. URL: <http://arxiv.org/abs/1910.04851>.
- [24] Yarin Gal, Jiri Hron, and Alex Kendall. “Concrete dropout”. In: *Advances in Neural Information Processing Systems 2017-Decem* (2017), pp. 3582–3591. ISSN: 10495258.
- [25] Wesley J Maddox et al. “A Simple Baseline for Bayesian Uncertainty in Deep Learning”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.
- [26] Pavel Izmailov et al. “Averaging Weights Leads to Wider Optima and Better Generalization: 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018”. In: *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*. 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018 (2018). Ed. by Ricardo Silva, Amir Globerson, and Amir Globerson, pp. 876–885.
- [27] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles”. In: *nips*. 2017. DOI: [10.1007/BF00378152](https://doi.org/10.1007/BF00378152).
- [28] Charles Richter and N. Roy. “Safe Visual Navigation via Deep Learning and Novelty Detection”. In: *Robotics: Science and Systems*. 2017. DOI: [10/gfx5pb](https://doi.org/10/gfx5pb).
- [29] Lukas Brunke et al. “Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning”. In: *arXiv:2108.06266 [cs, eess]* (Aug. 2021). arXiv: [2108.06266 \[cs, eess\]](https://arxiv.org/abs/2108.06266).
- [30] Jeremiah Zhe Liu et al. “Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness”. In: (), p. 15.
- [31] Moksh Jain et al. “DEUP: Direct Epistemic Uncertainty Prediction”. In: *ArXiv* (2021).
- [32] Heinrich Jiang et al. “To trust or not to trust a classifier”. In: *Advances in Neural Information Processing Systems 2018-Decem.Nips* (2018), pp. 5541–5552. ISSN: 10495258.
- [33] Théo Lacombe et al. “Topological Uncertainty: Monitoring Trained Neural Networks through Persistence of Activation Graphs”. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2021*, p. 13.
- [34] Yaniv Ovadia et al. “Can You Trust Your Model’s Uncertainty? Evaluating Predictive Uncertainty under Dataset Shift”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.

- [35] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. “Obtaining well calibrated probabilities using Bayesian Binning”. In: *Proceedings of the National Conference on Artificial Intelligence 4* (2015), pp. 2901–2907. ISSN: 2159-5399.
- [36] Mingxing Tan and Quoc V. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 6105–6114.
- [37] Tilmann Gneiting and Adrian E. Raftery. “Strictly proper scoring rules, prediction, and estimation”. In: *Journal of the American Statistical Association* 102.477 (2007), pp. 359–378. ISSN: 01621459. DOI: [10.1198/016214506000001437](https://doi.org/10.1198/016214506000001437).
- [38] C. Chow. “An Optimum Character Recognition System Using Decision Functions”. In: *IRE Trans. Electron. Comput.* (1957). DOI: [10/b4ntr4](https://doi.org/10/b4ntr4).
- [39] Yonatan Geifman and Ran El-Yaniv. “Selective Classification for Deep Neural Networks”. In: (), p. 10.
- [40] Wenming Jiang, Ying Zhao, and Zehan Wang. “Risk-Controlled Selective Prediction for Regression Deep Neural Network Models”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. Glasgow, United Kingdom: IEEE, July 2020, pp. 1–8. ISBN: 978-1-72816-926-2. DOI: [10/gm5dgc](https://doi.org/10/gm5dgc).
- [41] Yukun Ding et al. “Revisiting the Evaluation of Uncertainty Estimation and Its Application to Explore Model Complexity-Uncertainty Trade-Off”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Seattle, WA, USA: IEEE, June 2020, pp. 22–31. ISBN: 978-1-72819-360-1. DOI: [10/gm47n4](https://doi.org/10/gm47n4).
- [42] Felix Salfner, Maren Lenk, and Mirosław Malek. “A survey of online failure prediction methods”. In: *ACM Computing Surveys* 42.3 (2010), pp. 1–68. ISSN: 03600300. DOI: [10.1145/1670679.1670680](https://doi.org/10.1145/1670679.1670680).
- [43] Pierre-Yves Oudeyer. “Autonomous Development and Learning in Artificial Intelligence and Robotics: Scaling up Deep Learning to Human-like Learning”. In: *The Behavioral and brain sciences* 40 (Dec. 2017), e275. DOI: [10.1017/S0140525X17000243](https://doi.org/10.1017/S0140525X17000243). arXiv: [1712.01626](https://arxiv.org/abs/1712.01626).
- [44] Oliver Groth et al. “Is Curiosity All You Need? On the Utility of Emergent Behaviours from Curious Exploration”. In: *arXiv:2109.08603 [cs]* (Sept. 2021). arXiv: [2109.08603 \[cs\]](https://arxiv.org/abs/2109.08603).
- [45] Cédric Colas et al. “Intrinsically Motivated Goal-Conditioned Reinforcement Learning: A Short Survey”. In: (Dec. 2020). arXiv: [2012.09830](https://arxiv.org/abs/2012.09830).
- [46] Corey Lynch et al. “Learning Latent Plans from Play”. In: *arXiv:1903.01973 [cs]* (Dec. 2019). arXiv: [1903.01973 \[cs\]](https://arxiv.org/abs/1903.01973).
- [47] Corey Lynch and Pierre Sermanet. “Language Conditioned Imitation Learning over Unstructured Data”. In: *arXiv:2005.07648 [cs]* (July 2021). arXiv: [2005.07648 \[cs\]](https://arxiv.org/abs/2005.07648).
- [48] Charu C Aggarwal et al. “Active Learning: A Survey”. In: *Algorithms and Applications* (), p. 35.
- [49] Sonia Chernova and Andrea Thomaz. “Robot Learning from Human Teachers”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8 (Apr. 2014), pp. 1–121. DOI: [10/gf7bxx](https://doi.org/10/gf7bxx).
- [50] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. “Self-Supervised Exploration via Disagreement”. In: *International Conference on Machine Learning*. PMLR, May 2019, pp. 5062–5071.

- [51] H. S. Seung, M. Opper, and H. Sompolinsky. “Query by Committee”. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. New York, NY, USA: Association for Computing Machinery, July 1992, pp. 287–294. ISBN: 978-0-89791-497-0. DOI: [10/d7nmqs](https://doi.org/10/d7nmqs).
- [52] William R. Clements et al. “Estimating Risk and Uncertainty in Deep Reinforcement Learning”. In: *arXiv:1905.09638 [cs, stat]* (Sept. 2020). arXiv: [1905.09638 \[cs, stat\]](https://arxiv.org/abs/1905.09638).
- [53] Evgenii Safronov, Michele Colledanchise, and Lorenzo Natale. “Task Planning with Belief Behavior Trees”. In: *arXiv:2008.09393 [cs]* (Aug. 2020). arXiv: [2008.09393 \[cs\]](https://arxiv.org/abs/2008.09393).

Chapter 6 : Implementation and validation on a planar pick and place learning task

- [1] Shiwali Mohan. “From Verbs to Tasks: An Integrated Account of Learning Tasks from Situated Interactive Instruction”. PhD thesis. 2015. ISBN: 9781321727371.
- [2] Rikke Gade and Thomas B. Moeslund. “Thermal Cameras and Applications: A Survey”. In: *Machine Vision and Applications* 25.1 (Jan. 2014), pp. 245–262. ISSN: 0932-8092. DOI: [10.1007/s00138-013-0570-5](https://doi.org/10.1007/s00138-013-0570-5).
- [3] Miguel Simão, Pedro Neto, and Olivier Gibaru. “EMG-based online classification of gestures with recurrent neural networks”. In: *Pattern Recognition Letters* 128 (2019), pp. 45–51. ISSN: 0167-8655. DOI: <https://doi.org/10.1016/j.patrec.2019.07.021>. URL: <http://www.sciencedirect.com/science/article/pii/S0167865519302089>.
- [4] Yang Zhang and Chris Harrison. “Tomo: Wearable, Low-Cost Electrical Impedance Tomography for Hand Gesture Recognition”. In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. UIST ’15. Charlotte, NC, USA: ACM, 2015, pp. 167–173. ISBN: 978-1-4503-3779-3. DOI: [10.1145/2807442.2807480](https://doi.org/10.1145/2807442.2807480). URL: <http://doi.acm.org/10.1145/2807442.2807480>.
- [5] B. J. Edelman et al. “Noninvasive neuroimaging enhances continuous neural tracking for robotic device control”. In: *Science Robotics* 4.31 (2019). DOI: [10.1126/scirobotics.aaw6844](https://doi.org/10.1126/scirobotics.aaw6844). eprint: <https://robotics.sciencemag.org/content/4/31/eaaw6844.full.pdf>. URL: <https://robotics.sciencemag.org/content/4/31/eaaw6844>.
- [6] Iretiayo Akinola et al. “Task level hierarchical system for BCI-enabled shared autonomy”. In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)* (2017), pp. 219–225.
- [7] Hakim Si-Mohammed et al. “Towards BCI-based Interfaces for Augmented Reality: Feasibility, Design and Evaluation”. In: *IEEE Transactions on Visualization and Computer Graphics* PP (Oct. 2018), pp. 1–1. DOI: [10.1109/TVCG.2018.2873737](https://doi.org/10.1109/TVCG.2018.2873737).
- [8] Keyur Patel et al. “Facial Sentiment Analysis Using AI Techniques: State-of-the-Art, Taxonomies, and Challenges”. In: *IEEE Access* 8 (2020), pp. 90495–90519. ISSN: 2169-3536. DOI: [10/gmkz59](https://doi.org/10/gmkz59).
- [9] Wansoo Kim et al. “Anticipatory Robot Assistance for the Prevention of Human Static Joint Overloading in Human-Robot Collaboration”. In: *IEEE Robotics and Automation Letters* 3.1 (2018), pp. 68–75. ISSN: 23773766. DOI: [10.1109/LRA.2017.2729666](https://doi.org/10.1109/LRA.2017.2729666).
- [10] Martin Schepers, Matteo Giuberti, and G. Bellusci. *Xsens MVN: Consistent Tracking of Human Motion Using Inertial Sensing*. Mar. 2018. DOI: [10.13140/RG.2.2.22099.07205](https://doi.org/10.13140/RG.2.2.22099.07205).
- [11] Mohammad Safeea, Pedro Neto, and Richard Bearee. “On-Line Collision Avoidance for Collaborative Robot Manipulators by Adjusting off-Line Generated Paths: An Industrial Use Case”. In: *Robotics and Autonomous Systems* 119 (Sept. 2019), pp. 278–288. ISSN: 0921-8890. DOI: [10/ggs7wb](https://doi.org/10/ggs7wb).
- [12] Zhe Cao et al. “Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields”. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Vol. 2017-Janua. Dec. 2017, pp. 1302–1310. ISBN: 978-1-5386-0457-

1. DOI: [10.1109/CVPR.2017.143](https://doi.org/10.1109/CVPR.2017.143). arXiv: [1812.08008](https://arxiv.org/abs/1812.08008).
- [13] Christian Zimmermann et al. “3D Human Pose Estimation in RGBD Images for Robotic Task Learning”. In: *CoRR* abs/1803.02622 (2018). arXiv: [1803.02622](https://arxiv.org/abs/1803.02622). URL: <http://arxiv.org/abs/1803.02622>.
- [14] Arnon Amir et al. “A Low Power, Fully Event-Based Gesture Recognition System”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, July 2017, pp. 7388–7397. ISBN: 978-1-5386-0457-1. DOI: [10/gncfgm](https://doi.org/10/gncfgm).
- [15] Guillermo Gallego et al. “Event-Based Vision: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–1. ISSN: 1939-3539. DOI: [10/gghfk7c](https://doi.org/10/gghfk7c).
- [16] Ali Shafti, Pavel Orlov, and A. Aldo Faisal. “Gaze-Based, Context-aware Robotic System for Assisted Reaching and Grasping”. In: (Sept. 2018). arXiv: [1809.08095](https://arxiv.org/abs/1809.08095).
- [17] Kuniyuki Takahashi and Jethro Tan. “Deep Visuo-Tactile Learning: Estimation of Tactile Properties from Images”. In: (2018). arXiv: [1803.03435](https://arxiv.org/abs/1803.03435). URL: <http://arxiv.org/abs/1803.03435>.
- [18] Ganna Pugach et al. “Touch-Based Admittance Control of a Robotic Arm Using Neural Learning of an Artificial Skin”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2016, pp. 3374–3380. ISBN: 978-1-5090-3762-9. DOI: [10.1109/IROS.2016.7759519](https://doi.org/10.1109/IROS.2016.7759519).
- [19] Zhe Su et al. “Force Estimation and Slip Detection/Classification for Grip Control Using a Biomimetic Tactile Sensor”. In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, Nov. 2015, pp. 297–303. ISBN: 978-1-4799-6885-5. DOI: [10.1109/HUMANOIDS.2015.7363558](https://doi.org/10.1109/HUMANOIDS.2015.7363558).
- [20] Mohammad Safeea, Pedro Neto, and Richard Béarée. “Precise Hand-Guiding of Redundant Manipulators with Null Space Control for in-Contact Obstacle Navigation”. In: *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*. Vol. 1. Oct. 2019, pp. 693–698. DOI: [10/gm4hgs](https://doi.org/10/gm4hgs).
- [21] Matej Vecerík et al. “Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards”. In: *CoRR* abs/1707.08817 (2017). arXiv: [1707.08817](https://arxiv.org/abs/1707.08817). URL: <http://arxiv.org/abs/1707.08817>.
- [22] Takahito Suzuki et al. “Knowledge Distillation for Throat Microphone Speech Recognition”. In: Sept. 2019, pp. 461–465. DOI: [10.21437/Interspeech.2019-1597](https://doi.org/10.21437/Interspeech.2019-1597).
- [23] Li Wan et al. “Generalized End-to-End Loss for Speaker Verification”. In: *arXiv e-prints*, arXiv:1710.10467 (2017), arXiv:1710.10467. arXiv: [1710.10467](https://arxiv.org/abs/1710.10467) [[eess.AS](https://arxiv.org/abs/1710.10467)].
- [24] Alan Akbik et al. “FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP”. In: *Proceedings of the 2019 Conference of the North*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2019, pp. 54–59. DOI: [10.18653/v1/N19-4010](https://doi.org/10.18653/v1/N19-4010).
- [25] Alan Akbik, Duncan Blythe, and Roland Vollgraf. “Contextual String Embeddings for Sequence Labeling”. In: Aug. 2018.
- [26] Zhe Cao et al. “OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields”. In: (Dec. 2018). arXiv: [1812.08008](https://arxiv.org/abs/1812.08008).
- [27] Gines Hidalgo Martinez et al. “Single-Network Whole-Body Pose Estimation”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019. ISBN: 978-1-72814-803-8. DOI: [10.1109/ICCV.2019.00708](https://doi.org/10.1109/ICCV.2019.00708). arXiv: [1909.13423](https://arxiv.org/abs/1909.13423).

- [28] Morgan Quigley et al. “ROS: An Open-Source Robot Operating System”. In: *ICRA Workshop on Open Source Software*. 2009.
- [29] Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. “YARP: Yet Another Robot Platform”. In: *International Journal of Advanced Robotic Systems* 3.1 (Mar. 2006), p. 8. ISSN: 1729-8814. DOI: [10.5772/5761](https://doi.org/10.5772/5761).
- [30] Albert S. Huang, Edwin Olson, and David C. Moore. “LCM: Lightweight Communications and Marshalling”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2010, pp. 4057–4062. DOI: [10/cgvtdt](https://doi.org/10/cgvtdt).
- [31] Ayssam Elkady and Tarek Sobh. “Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography”. In: *Journal of Robotics* 2012 (May 2012), e959013. ISSN: 1687-9600. DOI: [10/gb79nj](https://doi.org/10/gb79nj).
- [32] S. Garrido-Jurado et al. “Generation of Fiducial Marker Dictionaries Using Mixed Integer Linear Programming”. In: *Pattern Recognition* 51 (Mar. 2016), pp. 481–491. ISSN: 0031-3203. DOI: [10.1016/j.patcog.2015.09.023](https://doi.org/10.1016/j.patcog.2015.09.023).
- [33] Francisco J. Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. “Speeded up Detection of Squared Fiducial Markers”. In: *Image and Vision Computing* 76 (Aug. 2018), pp. 38–47. ISSN: 0262-8856. DOI: [10/gd5gng](https://doi.org/10/gd5gng).
- [34] François Hélénon et al. “Learning Prohibited and Authorised Grasping Locations from a Few Demonstrations”. In: *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. Aug. 2020, pp. 1094–1100. DOI: [10/gk8djs](https://doi.org/10/gk8djs).

Chapter 7: Conclusion and perspectives

- [1] Carlos Rubert et al. “Grasp success prediction with quality metrics”. In: (2018), pp. 1–5. URL: <http://arxiv.org/abs/1809.03276>.
- [2] Robert Krug et al. “Analytic grasp success prediction with tactile feedback”. In: *Proceedings - IEEE International Conference on Robotics and Automation 2016-June* (2016), pp. 165–171. ISSN: 10504729. DOI: [10.1109/ICRA.2016.7487130](https://doi.org/10.1109/ICRA.2016.7487130).
- [3] Máximo A. Roa and Raúl Suárez. “Grasp quality measures: review and performance”. In: *Autonomous Robots* 38.1 (2014), pp. 65–88. ISSN: 09295593. DOI: [10.1007/s10514-014-9402-3](https://doi.org/10.1007/s10514-014-9402-3).
- [4] Ghazal Ghazaei et al. “Dealing with Ambiguity in Robotic Grasping via Multiple Predictions”. In: *Computer Vision – ACCV 2018*. Springer International Publishing, May 2019, pp. 38–55. DOI: [10/gmshbk](https://doi.org/10/gmshbk).
- [5] Charles Corbière et al. “Addressing Failure Prediction by Learning Model Confidence”. In: *NeurIPS* (2019), pp. 1–11. URL: <http://arxiv.org/abs/1910.04851>.
- [6] Tom Schaul et al. “Prioritized Experience Replay”. In: (2015), pp. 1–21. URL: <http://arxiv.org/abs/1511.05952>.
- [7] Marcin Andrychowicz et al. “Hindsight Experience Replay”. In: *arXiv:1707.01495 [cs]* (Feb. 2018). arXiv: [1707.01495 \[cs\]](https://arxiv.org/abs/1707.01495).
- [8] David Isele and Akansel Cosgun. “Selective Experience Replay for Lifelong Learning”. In: (), p. 8.
- [9] M. Turk. “Multimodal Interaction: A Review”. In: *Pattern Recognit. Lett.* (2014). DOI: [10/f5mh52](https://doi.org/10/f5mh52).
- [10] Khaled Bayouhdh et al. “A Survey on Deep Multimodal Learning for Computer Vision: Advances, Trends, Applications, and Datasets”. In: *The Visual Computer* (June 2021). ISSN: 1432-2315. DOI: [10/gkjdrj](https://doi.org/10/gkjdrj).
- [11] Kuniyuki Takahashi and Jethro Tan. “Deep Visuo-Tactile Learning: Estimation of Tactile Properties from Images”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2019, pp. 8951–8957. ISBN: 978-1-5386-6027-0. DOI: [10.1109/ICRA.2019.8794285](https://doi.org/10.1109/ICRA.2019.8794285).
- [12] Wenzhong Guo, Jianwen Wang, and Shiping Wang. “Deep Multimodal Representation Learning: A Survey”. In: *IEEE Access* 7 (2019), pp. 63373–63394. ISSN: 21693536. DOI: [10.1109/ACCESS.2019.2916887](https://doi.org/10.1109/ACCESS.2019.2916887).
- [13] Humam Alwassel et al. “Self-Supervised Learning by Cross-Modal Audio-Video Clustering”. In: *arXiv:1911.12667 [cs]* (Oct. 2020). arXiv: [1911.12667 \[cs\]](https://arxiv.org/abs/1911.12667).
- [14] George Papandreou et al. “Adaptive Multimodal Fusion by Uncertainty Compensation With Application to Audiovisual Speech Recognition”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 17.3 (Mar. 2009), pp. 423–435. ISSN: 1558-7924. DOI: [10/cswfz7](https://doi.org/10/cswfz7).
- [15] Susanne Trick et al. “Multimodal Uncertainty Reduction for Intention Recognition in Human-Robot Interaction”. In: *arXiv:1907.02426 [cs, stat]* (July 2019). arXiv: [1907.02426 \[cs, stat\]](https://arxiv.org/abs/1907.02426).
- [16] Fionn Murtagh and Pedro Contreras. “Algorithms for Hierarchical Clustering: An Overview, II”. In: *WIREs Data Mining and Knowledge Discovery* 7.6 (2017), e1219. ISSN: 1942-4795. DOI: [10/gctpdc](https://doi.org/10/gctpdc).

- [17] Michele Colledanchise and Lorenzo Natale. “Improving the Parallel Execution of Behavior Trees”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2018, pp. 7103–7110. DOI: [10/gmx33x](https://doi.org/10/gmx33x).
- [18] Michele Colledanchise and Lorenzo Natale. “Analysis and Exploitation of Synchronized Parallel Executions in Behavior Trees”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nov. 2019, pp. 6399–6406. DOI: [10/gmx33z](https://doi.org/10/gmx33z).
- [19] Shelley Nason and John E. Laird. “Soar-RL: Integrating Reinforcement Learning with Soar”. In: *Cognitive Systems Research. Special Issue of Cognitive Systems Research - The Best Papers from ICCM2004 6.1 (Mar. 2005)*, pp. 51–59. ISSN: 1389-0417. DOI: [10/cxhg7](https://doi.org/10/cxhg7).
- [20] Raja Chatila et al. “Toward Self-Aware Robots”. In: *Frontiers in Robotics and AI* 5 (Aug. 2018). ISSN: 2296-9144. DOI: [10.3389/frobt.2018.00088](https://doi.org/10.3389/frobt.2018.00088).
- [21] Rahul Dey and Chris Child. “QL-BT: Enhancing Behaviour Tree Design and Implementation with Q-learning”. In: *2013 IEEE Conference on Computational Intelligence in Games (CIG)*. Aug. 2013, pp. 1–8. DOI: [10/gmmtvs](https://doi.org/10/gmmtvs).
- [22] Dianmu Zhang. “Building Modular, Human-Interpretable AI Systems with Behavior Trees”. In: (), p. 87.
- [23] Mart Kartasev. *Integrating Reinforcement Learning into Behavior Trees by Hierarchical Composition*. 2019.
- [24] Joris Guérin et al. “Learning Local Trajectories for High Precision Robotic Tasks: Application to KUKA LBR Iiwa Cartesian Positioning”. In: *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*. Oct. 2016, pp. 5316–5321. DOI: [10/gmx4b5](https://doi.org/10/gmx4b5).
- [25] Andrea L. Thomaz et al. “Interaction for Task Instruction and Learning”. In: *Interactive Task Learning: Humans, Robots, and Agents Acquiring New Tasks through Natural Interactions*. MIT Press, Sept. 2019, pp. 91–110. ISBN: 978-0-262-03882-9.
- [26] Bernhard Schölkopf et al. “Towards Causal Representation Learning”. In: *arXiv:2102.11107 [cs]* (Feb. 2021). arXiv: [2102.11107 \[cs\]](https://arxiv.org/abs/2102.11107).
- [27] Thomas Hellström. “The Relevance of Causation in Robotics: A Review, Categorization, and Analysis”. In: *Paladyn, Journal of Behavioral Robotics* 12.1 (Jan. 2021), pp. 238–255. ISSN: 2081-4836. DOI: [10/gjzng5](https://doi.org/10/gjzng5).
- [28] Judea Pearl. “3. The Foundations of Causal Inference”. In: *Sociological Methodology* 40.1 (Aug. 2010), pp. 75–149. ISSN: 0081-1750, 1467-9531. DOI: [10/cws6cv](https://doi.org/10/cws6cv).
- [29] E. Bareinboim et al. *1 On Pearl ’ s Hierarchy and the Foundations of Causal Inference*. 2021.
- [30] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. 1st. USA: Basic Books, Inc., 2018. ISBN: 978-0-465-09760-9.
- [31] Amandine Mayima, Aurélie Clodic, and Rachid Alami. “Towards Robots Able to Measure in Real-time the Quality of Interaction in HRI Contexts”. In: *International Journal of Social Robotics* (2021). DOI: [10/gnjdpv](https://doi.org/10/gnjdpv).
- [32] Sonia Chernova and Andrea Thomaz. “Robot Learning from Human Teachers”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8 (Apr. 2014), pp. 1–121. DOI: [10/gf7bxx](https://doi.org/10/gf7bxx).

Résumé étendu en français

- [1] Pooja Makula et al. “Multimodal Smart Robotic Assistant”. In: *Proceedings of 2015 International Conference on Signal Processing, Computing and Control, ISPCC 2015*. Institute of Electrical and Electronics Engineers Inc., Jan. 2016, pp. 18–23. ISBN: 978-1-4799-8436-7. DOI: [10.1109/ISPCC.2015.7374991](https://doi.org/10.1109/ISPCC.2015.7374991).
- [2] François Hélénon et al. “Learning Prohibited and Authorised Grasping Locations from a Few Demonstrations”. In: *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. Aug. 2020, pp. 1094–1100. DOI: [10/gk8djs](https://doi.org/10/gk8djs).
- [3] François Hélénon et al. “Cognitive Architecture for Intuitive and Interactive Task Learning in Industrial Collaborative Robotics”. In: *The 5th International Conference on Robotics, Control and Automation, ICRCA 2021*. 2021. ISBN: 978-1-4503-8748-4.
- [4] Iuliia Kotseruba and John K. Tsotsos. “40 Years of Cognitive Architectures: Core Cognitive Abilities and Practical Applications”. In: *Artificial Intelligence Review* 53.1 (July 2020), pp. 17–94. ISSN: 15737462. DOI: [10.1007/s10462-018-9646-y](https://doi.org/10.1007/s10462-018-9646-y).
- [5] Tom Gruber. “Ontology”. In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 1963–1965. ISBN: 978-0-387-39940-9. DOI: [10.1007/978-0-387-39940-9_1318](https://doi.org/10.1007/978-0-387-39940-9_1318).
- [6] Markus Waibel et al. “RoboEarth”. In: *IEEE Robotics Automation Magazine* 18.2 (June 2011), pp. 69–82. ISSN: 1558-223X. DOI: [10.1109/MRA.2011.941632](https://doi.org/10.1109/MRA.2011.941632).
- [7] Moritz Tenorth and Michael Beetz. “KnowRob: A Knowledge Processing Infrastructure for Cognition-Enabled Robots”. In: *International Journal of Robotics Research* (2013). ISSN: 02783649. DOI: [10.1177/0278364913481635](https://doi.org/10.1177/0278364913481635).
- [8] Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. “CRAM - A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments”. In: *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*. 2010. ISBN: 978-1-4244-6675-7. DOI: [10.1109/IROS.2010.5650146](https://doi.org/10.1109/IROS.2010.5650146).
- [9] R. Alami et al. “An Architecture for Autonomy”. In: *The International Journal of Robotics Research* 17.4 (Apr. 1998), pp. 315–337. ISSN: 0278-3649. DOI: [10/b2ts55](https://doi.org/10/b2ts55).
- [10] Mehdi Khamassi et al. “Integration of Action, Joint Action and Learning in Robot Cognitive Architectures”. In: *Intellectica - La revue de l'Association pour la Recherche sur les sciences de la Cognition (ARCo)* 2016/1.65 (June 2016), pp. 169–203. DOI: [10/gnhc4q](https://doi.org/10/gnhc4q).
- [11] Victor Paleologue. *Teaching Robots Behaviors Using Spoken Language in Rich and Open Scenarios*. Tech. rep. Sorbonne Université, Dec. 2019.
- [12] Francesco Roviida, Bjarne Grossmann, and Volker Krüger. “Extended Behavior Trees for Quick Definition of Flexible Robotic Tasks”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2017, pp. 6793–6800. DOI: [10.1109/IROS.2017.8206598](https://doi.org/10.1109/IROS.2017.8206598).
- [13] Michele Colledanchise and Petter Ögren. “Behavior Trees in Robotics and AI: An Introduction”. In: *arXiv* (Aug. 2017). DOI: [10.1201/9780429489105](https://doi.org/10.1201/9780429489105). arXiv: [1709.00084](https://arxiv.org/abs/1709.00084).

- [14] Zhongxuan Cai et al. “BT Expansion: A Sound and Complete Algorithm for Behavior Planning of Intelligent Robots with Behavior Trees”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.7 (May 2021), pp. 6058–6065. ISSN: 2374-3468.
- [15] Andy Zeng et al. *Learning Synergies between Pushing and Grasping with Self-supervised Deep Reinforcement Learning*. Tech. rep. 2018, p. 7. URL: <http://vpg.cs.princeton.edu><http://vpg.cs.princeton.edu>.
- [16] Gao Huang et al. “Densely connected convolutional networks”. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-Janua (2017), pp. 2261–2269. ISSN: 0022-4790. DOI: [10.1109/CVPR.2017.243](https://doi.org/10.1109/CVPR.2017.243).
- [17] Shiwali Mohan. “From Verbs to Tasks: An Integrated Account of Learning Tasks from Situated Interactive Instruction”. PhD thesis. 2015. ISBN: 9781321727371.
- [18] Eyke Hüllermeier and Willem Waegeman. “Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods”. In: *Machine Learning* 110.3 (Mar. 2021), pp. 457–506. ISSN: 1573-0565. DOI: [10/gjp8b3](https://doi.org/10/gjp8b3).
- [19] Chuan Guo et al. “On Calibration of Modern Neural Networks”. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70. ICML’17*. Sydney, NSW, Australia: JMLR.org, Aug. 2017, pp. 1321–1330.
- [20] Yukun Ding et al. “Revisiting the Evaluation of Uncertainty Estimation and Its Application to Explore Model Complexity-Uncertainty Trade-Off”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Seattle, WA, USA: IEEE, June 2020, pp. 22–31. ISBN: 978-1-72819-360-1. DOI: [10/gm47n4](https://doi.org/10/gm47n4).
- [21] Felix Salfner, Maren Lenk, and Mirosław Malek. “A survey of online failure prediction methods”. In: *ACM Computing Surveys* 42.3 (2010), pp. 1–68. ISSN: 03600300. DOI: [10.1145/1670679.1670680](https://doi.org/10.1145/1670679.1670680).
- [22] Evgenii Safronov, Michele Colledanchise, and Lorenzo Natale. “Task Planning with Belief Behavior Trees”. In: *arXiv:2008.09393 [cs]* (Aug. 2020). arXiv: [2008.09393 \[cs\]](https://arxiv.org/abs/2008.09393).



François Hélénon
**Architecture robotique et cognitive pour
l'apprentissage de tâches en interaction avec
l'humain. Une application pour la
collaboration homme/robot dans l'Industrie
4.0.**

**HESAM
UNIVERSITÉ**

Résumé : Dans l'Industrie 4.0/5.0, les robots collaboratifs peuvent désormais assister dans de nombreuses tâches, contribuant ainsi à réduire les risques de troubles musculo-squelettiques pour les travailleurs humains. Cependant, la reconfiguration des robots collaboratifs manque encore de flexibilité et est souvent hors de portée du travailleur du quotidien, qui n'est ni un programmeur ni un expert en robotique. De telles exigences conduisent à un changement de paradigme dans la façon dont les robots collaboratifs doivent être programmés. Un robot collaboratif idéal devrait devenir un Assistant Robotique Intelligent (SRA), centré sur l'humain, capable d'adapter dynamiquement son comportement à la diversité de chaque situation, y compris les tâches, les changements d'environnement, les caractéristiques des travailleurs et leurs préférences. Durant cette thèse, nous avons choisi de développer un type d'architecture cognitive pour la robotique collaborative dans un contexte industriel, prenant en compte différentes spécifications pour qu'un humain puisse enseigner des tâches à un robot de manière incrémentale et avec des modalités d'interaction naturelles.

Mots clés : Apprentissage robotique en interaction, Apprentissage de tâches en interaction, Architecture cognitive hybride, Apprendre par démonstrations humaines, Robotique Collaborative

Abstract : In Industry 4.0/5.0, collaborative robots can now assist in many tasks, helping to reduce the risk of musculoskeletal disorders for human workers. However, the reconfiguration of collaborative robots still lacks flexibility and is often beyond the reach of the everyday worker, who is neither a programmer nor a robotics expert. Such requirements lead to a paradigm shift in the way collaborative robots should be programmed. An ideal collaborative robot should become a human-centered Smart Robotic Assistant (SRA), capable of dynamically adapting its behavior to the diversity of each situation, including tasks, environment changes, workers' characteristics and preferences. During this thesis, we chose to develop a type of cognitive architecture for collaborative robotics in an industrial context, taking into account different specifications so that a human can teach tasks to a robot in an incremental way and with natural interaction modalities.

Keywords : Interactive Robot Learning, Interactive Task Learning, Hybrid cognitive architecture, Learning from human demonstrations, Collaborative robotics