Modèle profond pour le contrôle vocal adaptatif d'un habitat intelligent

Alexis Brenon

▶ **To cite this version:**

Alexis Brenon. Modèle profond pour le contrôle vocal adaptatif d'un habitat intelligent. Intelligence artificielle [cs.AI]. Université Grenoble Alpes, 2017. Français. NNT : 2017GREAM057 . tel-01818123

HAL Id: tel-01818123
https://theses.hal.science/tel-01818123

Submitted on 18 Jun 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de
DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES
Spécialité : **Informatique**
Arrêté ministériel : 25 mai 2016

Présentée par
Alexis Brenon

Thèse dirigée par **Michel Vacher, Ingénieur de Recherche CNRS HDR, LIG**, et
co-encadrée par **François Portet, Maître de Conférences, Grenoble INP, LIG**

préparée au sein du **Laboratoire d'Informatique de Grenoble**
dans **l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

Modèle profond pour le contrôle vocal adaptatif d'un habitat intelligent

Thèse soutenue publiquement le **14 décembre 2017**,
devant le jury composé de :
Mme. Christine Verdier
Professeur, Université Grenoble Alpes, présidente
M. François Charpillet
Directeur de Recherche INRIA, LORIA, Nancy, rapporteur
M. Norbert Noury
Professeur des Universités, Université de Lyon, rapporteur
M. Anthony Fleury
Enseignant-chercheur, Institut Mines-Télécom Lille Douai, examinateur
M. Michel Vacher
Ingénieur de Recherche CNRS HDR, LIG, directeur de thèse
M. François Portet
Maître de Conférences, Grenoble INP, LIG, co-encadrant de thèse

À mon fils :
Sur les chemins de la découverte,
la curiosité sera toujours une boussole.

À mon ego du passé :
Parce que changer le monde
se fait pas après pas.

Résumé

Les habitats intelligents, résultants de la convergence de la domotique, de l'informatique ubiquitaire et de l'intelligence artificielle, assistent leurs habitants dans les situations du quotidien pour améliorer leur qualité de vie. En permettant aux personnes dépendantes et âgées de rester à domicile plus longtemps, ces habitats permettent de fournir une première réponse à des problèmes de société comme la dépendance due au vieillissement de la population.

En nous plaçant dans un habitat contrôlé par la voix, l'habitat doit répondre aux requêtes d'un utilisateur concernant un ensemble d'actions pouvant être automatisées (contrôle des lumières, des volets, des dispositifs multimédia, etc.). Pour atteindre cet objectif, le système de contrôle de l'habitat a besoin de prendre en compte le contexte dans lequel un ordre est donné mais également de connaître les habitudes et préférences de l'utilisateur. Pour cela, le système doit pouvoir agréger les informations issues du réseau de capteurs domotiques hétérogènes et prendre en compte le comportement (variable) de l'utilisateur.

La mise au point de systèmes de contrôle intelligent d'un habitat est particulièrement ardue du fait de la grande variabilité concernant aussi bien la topologie des habitats que les habitudes des utilisateurs. Par ailleurs, l'ensemble des informations contextuelles doivent être représentées dans un référentiel commun dans un objectif de raisonnement et de prise de décision. Pour répondre à ces problématiques, nous proposons de développer un système qui d'une part modifie continuellement son modèle de manière à s'adapter à l'utilisateur, et qui d'autre part utilise directement les données issues des capteurs à travers une représentation graphique. L'intérêt et l'originalité de cette méthode sont de ne pas nécessiter d'inférence pour déterminer le contexte. Notre système repose ainsi sur une méthode d'apprentissage par renforcement profond qui couple un réseau de neurones profond du type convolutif permettant l'extraction de données contextuelles, avec un mécanisme d'apprentissage par renforcement pour la prise de décision.

Ce mémoire présente alors deux systèmes, un premier reposant uniquement sur l'apprentissage par renforcement et montrant les limites de cette approche sur des environnements réels pouvant comporter plusieurs milliers d'états possibles. L'introduction de l'apprentissage profond a permis la mise au point du second système, ARCADES, dont les bonnes performances montrent la pertinence d'une telle approche, tout en ouvrant de nombreuses voies d'améliorations.

Mots-clés

Habitat intelligent – Prise de décision en contexte – Apprentissage par renforcement – Apprentissage profond

Keywords

Smart-home – Decision system – Context-aware – Reinforcement learning – Deep learning

Abstract

Smart-homes, resulting of the merger of home-automation, ubiquitous computing and artificial intelligence, support inhabitants in their activity of daily living to improve their quality of life. Allowing dependent and aged people to live at home longer, these homes provide a first answer to society problems as the dependency tied to the aging population.

In voice controlled home, the home has to answer to user's requests covering a range of automated actions (lights, blinds, multimedia control, etc.). To achieve this, the control system of the home need to be aware of the context in which a request has been done, but also to know user habits and preferences. Thus, the system must be able to aggregate information from a heterogeneous home-automation sensors network and take the (variable) user behavior into account.

The development of smart home control systems is hard due to the huge variability regarding the home topology and the user habits. Furthermore, the whole set of contextual information need to be represented in a common space in order to be able to reason about them and make decisions. To address these problems, we propose to develop a system which updates continuously its model to adapt itself to the user and which uses raw data from the sensors through a graphical representation. This new method is particularly interesting because it does not require any prior inference step to extract the context. Thus, our system uses deep reinforcement learning; a convolutional neural network allowing to extract contextual information and reinforcement learning used for decision-making.

Then, this memoir presents two systems, a first one only based on reinforcement learning showing limits of this approach against real environment with thousands of possible states. Introduction of deep learning allowed to develop the second one, Arcades, which gives good performances proving that this approach is relevant and opening many ways to improve it.

Remerciements

Bien que ce mémoire soit signé de mon seul nom, il n'en est pas moins le résultat d'un travail qui n'aurait pu être réalisé seul. Au cours de ces trois années passées sur ce projet, j'ai eu l'occasion de côtoyer différentes personnes qui d'une façon ou d'une autre ont conduit à cet aboutissement. Pour leur concours dans cette réalisation, je souhaiterais ici les remercier.

Mes premiers remerciements vont à Michel Vacher et François Portet, mes directeurs de thèse. Avant de commencer cette thèse, j'avais été prévenu : « Choisir son sujet c'est bien, choisir ses directeurs c'est primordial ». Je ne peux que confirmer ce conseil, tant l'implication des directeurs dans le travail de leur doctorant est important pour un bon déroulement. Merci donc à Michel pour sa présence quasi-quotidienne, son écoute et son expérience de la rédaction sans quoi la qualité de mes écrits serait certainement moindre. Merci à François pour son ouverture d'esprit, sa rigueur scientifique et la qualité de ses corrections malgré des efforts à faire concernant la lisibilité. Au-delà de leurs qualités personnelles, leur complémentarité fût un réel atout tout au long de la thèse. Je souhaite également remercier Laurent Besacier qui, même s'il n'est pas directement lié à la thèse, a pris en charge la plupart des interactions avec les membres du projet CASSIE et défini les lignes directrices à suivre à ce sujet.

Mes remerciements vont ensuite à l'ensemble du jury. D'abord, François Charpillet et Norbert Noury, qui ont accepté de rapporter mes travaux. Ils ont ainsi permis, non seulement d'améliorer ce mémoire, mais également de préparer au mieux la soutenance. Ensuite, Christine Verdier et Anthony Fleury, qui par leurs nombreuses questions ont fait émerger de nouvelles pistes de recherches toujours plus intéressantes.

Enfin, je remercie les membres de l'équipe GETALP, et plus généralement du LIG, qui ont su créer un cadre de travail serein et agréable pour le bon déroulement de ce projet. Une mention spéciale aux (anciens) (post-)doctorants avec qui une réelle complicité a pu

naître au cours de nos pérégrinations dans les couloirs du laboratoire : Élodie, Jérémy, David, Belen et Christophe.

Bien sûr je remercie ma famille, mes parents et mes frères qui assurément ont joué un rôle dans la personne que je suis devenue. Alison et Éliam sans qui je ne serais que le tiers de ce que je suis aujourd'hui.

Mes derniers remerciements vont à des personnes qui ont pris part de manière plus indirecte à cette réalisation. Laurence Pierre et ma mère sans qui je n'aurais probablement jamais eu le courage de me lancer dans l'aventure scientifique d'une thèse. Tim Berners-Lee, Sergey Brin et Larry Page pour leurs contributions à l'Internet, un outil qui façonne assurément notre façon de faire de la recherche aujourd'hui.

Enfin, puisque la totalité vaut plus que la somme des parties, j'ai bon espoir que ce mémoire soit bénéfique pour chacune des personnes citée jusqu'ici.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Liste des acronymes

A

ADL : *Activities of Daily Living* (Activité de la vie quotidienne)

ARCADES : *Adaptive Reinforced Context-Aware Deep Decision System* (Système adaptatif de prise de décision en contexte par apprentissage profond par renforcement)

C

CNN : *Convolutional Neural Network* (Réseau de neurones convolutifs)

D

DL : *Description Logic* (Logique de description)

DQN : *Deep Q-Network*

G

GPU : *Graphics Processing Unit* (Processeur/Carte graphique)

H

HIS : Habitat Intelligent pour la Santé

L

LOSOCV : *Leave One Subject Out Cross Validation* (Validation croisée par sujets)

M

MDP : *Markov Decision Process* (Processus de décision Markovien)

MLN : *Markov Logic Network* (Réseau logique de Markov)

MLP : *Multi-Layer Perceptron* (Perceptron multi-couches)

MOMDP : *Mixed Observability Markov Decision Process* (Processus de décision Markovien à observabilité mixte)

N

NEC : *Neural Episodic Control*

NN : *Neural Network* (Réseau de neurones)

O

OWL : *Ontology Web Language* (Langage d'ontologie pour le Web)

P

POMDP : *Partially Observable Markov Decision Process* (Processus de décision Markovien partiellement observable)

R

RL : *Reinforcement Learning* (Apprentissage par renforcement)

RMSProp : *Running Mean Square Propagation*

RNN : *Recurent Neural Network* (Réseau de neurones récurrents)

S

SGD : *Stochastic Gradient Descent* (Descente stochastique du gradient)

SVG : *Scalable Vector Graphic*

T

t-SNE : *t-distributed Stochastic Neighbor Embedding*

V

VUI : *Voice-User Interface* (Interface vocale)

Table des notations

Les lettres majuscules sont utilisées pour représenter des ensembles ou des fonctions. Les lettres minuscules sont utilisées pour représenter un élément d'un ensemble ou l'image d'une valeur par une fonction.

$\mathbb{E}X$	Espérance mathématique de la variable aléatoire X
$U : \mathcal{S} \to \mathbb{R}$	Fonction d'utilité des états
$\text{argmax}_a f(a)$	Valeur de a pour laquelle $f(a)$ est maximale
$\mathcal{U}$	Fonction de sélection aléatoire suivant une distribution de probabilité uniforme

Dans un processus de décision Markovien :

$\mathcal{S}$	Ensemble des états possibles
$\mathcal{A}$	Ensemble des actions possibles
P	Modèle de transition de l'environnement
$R : \mathcal{S} \to \mathbb{R}$	Fonction de récompense
γ	Coefficient d'atténuation de la récompense
s, s'	États
a, a'	Actions
r	Récompense
(s, a)	Q-état
t	Pas de temps
s_t, a_t, r_t	État, action, récompense à l'instant t
τ	Longueur d'un épisode
$P(s'\|s, a)$	Probabilité de transiter vers l'état s' depuis l'état s en exécutant l'action a
$V : \mathcal{S} \to \mathbb{R}$	Fonction de valeur

$Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$	Fonction de Q-valeur
$v \approx V(s)$	Valeur approchée d'un état
$q \approx Q(s, a)$	Valeur approchée d'un Q-état (Q-valeur)
π	Stratégie
$\pi(a\|s)$	Probabilité de choisir l'action a depuis l'état s en suivant la stratégie π

Dans l'apprentissage par renforcement :

k	Itération d'apprentissage
α	Coefficient d'apprentissage
ϵ	Probabilité de sélection aléatoire pour la stratégie ϵ-*greedy*
$\mathcal{I}$	Ensemble des interactions réalisées
Q-table $= (q_{s,a})_{s \in \mathcal{S}, a \in \mathcal{A}}$	Matrice des Q-valeurs permettant d'approximer la fonction de Q-valeur

Dans les réseaux de neurones :

$\mathbf{x}$	Vecteur d'entrée
$\mathbf{w}$	Vecteur de poids appris
b	Biais
f	Fonction d'activation, de non linéarité
n	Nombre de neurones
FC_n	Couche de neurones de taille n totalement connectée à la précédente
n	Nombre de filtres
f	Taille des filtres en pixels
s	Pas de convolution (*stride*) en pixels
p	Bourrage (*padding*) de l'image d'entrée
$C_n[f + p/s]$	Couche de neurones convolutive de n filtres de taille f, utilisant un pas de s et un bourrage de p

Dans l'apprentissage profond par renforcement :

Q-Network	Réseau de neurones approximant la fonction de Q-valeur
w	Paramètres du réseau de neurones
$\mathcal{B}$	Lot d'apprentissage (tiré aléatoirement parmi $\mathcal{I}$)

η	Taux d'apprentissage du réseau de neurones
∇	Gradient du réseau de neurones (calculé à partir de la différence entre la valeur de sortie et la valeur attendue)
`target_q`	Période de mise à jour du réseau target-Q
`minibatch_size`	Taille du lot d'apprentissage $\mathcal{B}$

Chapitre I

Introduction

Je n'ai pas peur des ordinateurs.
J'ai peur qu'ils viennent à nous manquer.

Isaac Asimov

I.1 Motivation

De nombreux termes sont utilisés pour désigner notre habitat, la maison, le domicile, le foyer, etc. et tous tirent leurs racines étymologiques du lieu central dans lequel on demeure, on séjourne. Du fait du temps qu'on y passe, on souhaite que ce lieu soit sécurisant et confortable et son aménagement joue sans nul doute un rôle prépondérant dans la qualité de vie de ses habitants. Les différentes révolutions technologiques de l'habitat n'ont fait qu'améliorer cette qualité de vie en proposant de nouveaux services. L'électricité et l'électronique ont ainsi permis l'émergence de la domotique, permettant de contrôler son habitat à travers des systèmes tels que la télésurveillance ou la régulation du chauffage.

Les travaux présentés dans ce mémoire se situent dans le domaine de **l'informatique ubiquitaire** (Weiser 1991) caractérisé par l'utilisation de l'informatique dans l'ensemble des objets qui nous entourent. L'application de méthodes d'intelligence artificielle aux données pouvant alors être récoltées a permis l'émergence de **l'intelligence ambiante** (Ducatel et coll. 2001) dont l'objectif est d'améliorer la qualité de vie des utilisateurs d'un environnement donné. L'intelligence ambiante peut être utilisée pour différentes applications. Avec un large ensemble de dispositifs visant à assister le conducteur, l'automobile est un exemple concret d'utilisation, atteignant son paroxysme avec les voitures autonomes.

Si la voiture est un espace restreint dédié à une seule activité, il n'en va pas de même pour l'habitat qui représente un espace bien plus complexe de par sa configuration et la

variété des activités qui s'y tiennent. La domotique est aujourd'hui limitée à la définition de quelques règles de fonctionnement, une manipulation manuelle et focalisée sur les problèmes d'interopérabilité. La principale motivation de nos travaux est alors de répondre au besoin de fournir une intelligence aux systèmes domotiques, une capacité d'adaptation à différents niveaux (à la personne, à l'habitat, à la situation) leur permettant d'adapter leur fonctionnement pour toujours améliorer la qualité de vie des habitants d'un tel **habitat intelligent**.

I.2 Domaine de recherche

Ce travail de thèse, réalisé entre 2014 et 2017, est financé par le projet d'investissement d'avenir CASSIE, en partenariat avec un ensemble d'industriels. L'objectif premier du projet était le développement de services incluant des applications mobiles au profit des usagers d'habitats domotisés.

Au sein de l'équipe GETALP (Groupe d'Étude en Traduction Automatique et Traitement Automatisé des Langues et de la Parole), ces travaux font suite à différents projets réalisés depuis plusieurs années. Les projets Reside-HIS, DesdHIS2 et Sweethome visaient chacun la réalisation d'un système de contrôle d'un habitat intelligent, se focalisant sur différents aspects, comme la détermination de situations de détresse, la reconnaissance d'appels de détresse ou le contrôle vocal d'un système domotique. Tous ces projets reposaient alors sur l'utilisation d'un habitat largement instrumenté et comprenant notamment des microphones dans chacune des pièces afin de récolter des informations ou pour le contrôle de l'habitat.

Les interfaces vocales (*Voice-User Interface* – VUI) intégrées aux environnements domestiques ont récemment soulevé un nouvel intérêt, comme le montre le nombre de projets d'habitats incluant un système de reconnaissance de la parole (Charalampos et Maglogiannis 2008 ; Popescu et coll. 2008 ; Badii et Boudy 2009 ; Hamill et coll. 2009 ; Filho et Moir 2010 ; Lecouteux, Vacher et Portet 2011 ; Gemmeke et coll. 2013 ; Christensen et coll. 2013 ; Cristoforetti et coll. 2014). Couplé à un habitat équipé de capteurs, tels que des détecteurs de présence, des systèmes multimédias, etc., ce type d'interface est particulièrement adapté aux personnes en perte d'autonomie (Portet et coll. 2013 ; Vacher, Caffiau et coll. 2015).

Héritage de ces projets, nos travaux prennent donc place dans un habitat domotisé commandé par la voix. Cet habitat a la capacité de fournir l'état de ses capteurs à un contrôleur mais également de modifier l'état de ses actionneurs. Toutefois, ces caractéristiques ne suffisent pas pour le qualifier d'habitat intelligent. En effet, l'intelligence du système réside dans sa capacité à utiliser ces données, issues de l'environnement, pour modifier sa

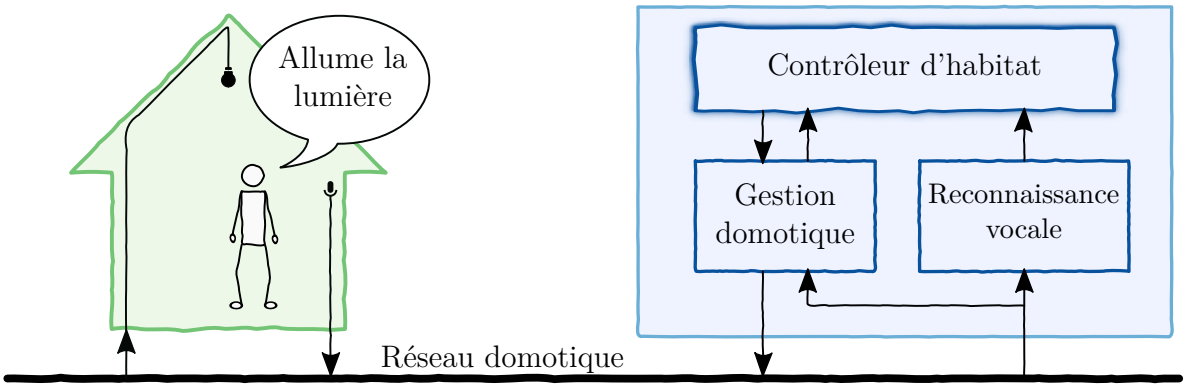


FIG. I.1 : Une architecture typique d'un habitat intelligent commandé par la voix. Exemple d'un utilisateur demandant à allumer la lumière.

façon d'agir sur le monde qui l'entoure. Dans un système contrôlé par la parole, l'habitat doit être *réactif* aux ordres vocaux pour réaliser l'action la plus judicieuse étant donné le contexte. Ce comportement peut être illustré par le scénario suivant :

Scénario 1 *L'habitant se lève au milieu de la nuit et prononce l'ordre « Allume la lumière ». Cette commande nécessite de connaitre certaines informations contextuelles (comme la localisation de l'utilisateur et son activité) pour déterminer quelle lampe allumer et avec quelle intensité. Ici, le système décide d'allumer la lampe de chevet à une intensité modérée car une luminosité plus importante (plafonnier, intensité maximale) pourrait créer un désagrément pour l'utilisateur.*

Les interactions entre l'utilisateur, son habitat et le système de contrôle sont représentées Figure I.1. Le réseau domotique, composé des différents capteurs (interrupteurs, détecteurs de mouvement, microphones, etc.) et actionneurs (lampes, volets, systèmes multimédias, etc.), permet alors au contrôleur d'interagir avec l'utilisateur et son environnement. De plus, l'ensemble de microphones placés dans chacune des pièces permet de commander le système depuis n'importe où et ce, sans nécessiter le port d'un quelconque objet.

I.3 Définition des problèmes

Le manque d'évolutivité des systèmes domotiques actuels est un frein à leur adoption. En effet, les cas dans lesquels une configuration pré-définie devient inadaptée sont nombreux et l'on peut en identifier quelques-uns :

- l'utilisateur change ses habitudes ou préférences ;
- un capteur tombe en panne ;
- un capteur est modifié ou supprimé ;

- l'utilisateur déménage.

Or, la complexité des systèmes actuels rend leur configuration par un utilisateur non-initié impossible et nécessite alors l'intervention d'un technicien. Une solution à ce problème est alors de proposer un système capable de s'adapter automatiquement aux préférences et habitudes de l'utilisateur ainsi qu'à l'environnement (topologie, capteurs, etc.) dans lequel il est utilisé. Mais cette solution soulève plusieurs autres problèmes.

La capacité d'adaptation du système est conditionnée par sa capacité à prendre en compte de continuelles informations permettant de définir le comportement adéquat. Nous proposons pour cela l'utilisation de l'apprentissage automatique. L'apprentissage devra pouvoir se faire de manière incrémentielle, permettant de prendre en compte les nouvelles informations qui pourraient arriver à tout moment. Dans le cas d'un apprentissage supervisé, les étiquettes utilisées ne devront pas nécessiter de coût trop élevé pour l'utilisateur.

Selon la solution d'apprentissage retenue, le problème de la représentation des données se posera également. En effet, les habitats domotisés ont la particularité d'agréger des données en provenance de nombreux capteurs hétérogènes. Par exemple, les prises de courant fournissent leur état sous forme binaire (allumée ou éteinte) alors que les stores intérieurs utilisent quatre valeurs distinctes (pour l'ouverture et l'inclinaison des lames). À l'inverse, certains capteurs fournissent une gamme de valeurs presque continue : 100 niveaux d'éclairage pour les lampes, des valeurs allant de 0 à 2^{64} pour la consommation d'eau, etc. Enfin, puisque nous nous plaçons dans un habitat commandé par la voix, le signal audio doit également être pris en compte et la représentation de l'ensemble de ces données de manière unifiée est un problème en soi.

I.4 Objectifs du travail de thèse

L'objectif principal de cette thèse est la mise au point d'un système de prise de décision adaptative pour les habitats intelligents pourvus d'une interface vocale.

La réalisation d'un tel système repose essentiellement sur l'implémentation d'un contrôleur capable d'apprendre à partir d'un ensemble de données afin de généraliser par la suite son comportement à des situations nouvelles. Pour cela, nos travaux doivent aboutir à deux contributions distinctes. D'une part, un système de prise de décision basé sur une méthode d'apprentissage automatique permettant au système d'adapter son modèle. D'autre part, un modèle de représentation des données permettant d'agréger dans un même référentiel des données hétérogènes, et permettant de contextualiser une décision. Ces propositions doivent tenir compte des contraintes que nous nous sommes fixées.

- La prise de décision doit se faire en temps interactif. C'est-à-dire que l'utilisateur ne doit pas ressentir de ralentissement conséquent dans la réalisation de ses tâches lorsqu'il est assisté par notre système.
- Le système doit s'adapter à l'utilisateur et à l'environnement de manière transparente et ce, tout au long de sa vie. Cette spécificité permet de répondre au problème d'évolutivité des systèmes existants.

Afin de valider notre approche, le système sera testé d'une part sur des données artificielles puis sur des données réelles issues d'un corpus précédemment enregistré.

I.5 Plan du manuscrit

Le Chapitre II présentera l'état de l'art lié à l'habitat intelligent ou à la prise de décision. Ce chapitre sera également l'occasion de revenir sur la terminologie utilisée dans l'ensemble de ce mémoire. Après une définition et des exemples d'habitats intelligents, nous aborderons l'extraction et l'utilisation du contexte dans ces habitats. Nous nous concentrerons ensuite sur la prise de décision en général avant d'aborder son application dans les habitats intelligents. Cette dernière section introduira notamment des exemples de systèmes de décision capables de s'adapter en continu grâce à l'utilisation de l'apprentissage par renforcement.

Les détails de fonctionnement de l'apprentissage par renforcement seront alors détaillés en introduction du Chapitre III. Nous présenterons ensuite l'implémentation d'une telle méthode d'apprentissage. La suite de ce chapitre sera consacrée aux détails de l'expérimentation préliminaire permettant de valider notre approche et aux résultats de cette expérimentation.

Au cours du Chapitre IV nous évoquerons notre nouvelle représentation des données. Le système décrit dans le chapitre précédent sera alors modifié, avec notamment un modèle reposant sur un réseau de neurones que nous introduirons dans ce même chapitre. L'implémentation de ce nouveau modèle, détaillée ensuite, donnera lieu à la création du système Arcades, contribution principale de cette thèse.

Le système Arcades sera évalué Chapitre V. Cette évaluation permettra de comparer cette nouvelle approche à celle présentée dans le Chapitre III. Des expérimentations complémentaires permettront par la suite de mieux appréhender le fonctionnement du système Arcades.

Finalement, le Chapitre VI viendra conclure ce mémoire en faisant le bilan des contributions apportées. Ce sera également l'occasion de prendre du recul en pointant les limites de nos approches et en proposant des perspectives d'évolutions.

Chapitre II

État de l'art

Si j'ai vu plus loin que les autres,
c'est que j'étais juché sur les épaules de géants.

Isaac Newton

Le cœur de ce travail de thèse est la réalisation d'un système capable de réagir à un évènement en prenant une *décision*, sur la base du *contexte* courant, dans le cadre d'un *habitat intelligent*. Bien que tous ces mots appartiennent au langage courant, leurs définitions dans un cadre informatique peuvent varier. Au cours de ce chapitre, nous définirons chacun de ces termes en abordant les notions essentielles à l'appréhension du contenu de ce manuscrit. Ces définitions seront également l'occasion de positionner nos travaux dans leur domaine. De la même façon, la présentation de projets abordant les mêmes thématiques nous permettra de comparer nos approches à celles déjà existantes.

II.1 L'habitat intelligent

II.1.1 Définition

Le terme « habitat intelligent » renferme de nombreuses significations selon le domaine dans lequel il est évoqué (Milion et coll. 2005). Dans notre cas, nous le considérerons comme le domaine résultant de la convergence de la *domotique* et de l'*intelligence ambiante*.

Construit à partir du latin *Domus* qui signifie maison, et du suffixe *-tique* associé à la technologie (électrique, électronique, informatique), la domotique est l'utilisation de technologies à l'intérieur et à proximité de la maison. Les applications sont nombreuses et

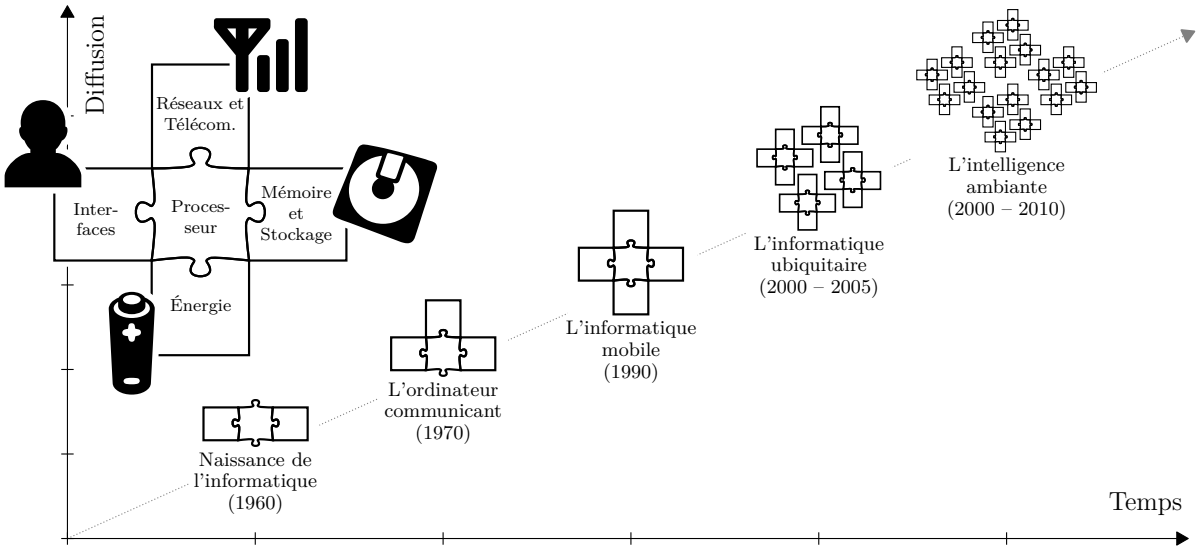


FIG. II.1 : L'évolution de l'informatique selon WALDNER (2007)

variées, de la gestion de l'habitat (éclairages, chauffage, etc.) aux loisirs numériques en passant par la sécurité ou la communication (GALLISSOT 2012). Toutefois, la domotique peine à se répandre. Il s'agit en effet d'une démarche consciente et éclairée ; et les habitants qui déploieront un système domotique dans leur domicile devront prendre en compte les coûts d'installation et les bénéfices qu'ils sont en droit d'attendre. Comme le rappelle GALLISSOT (2012), de nombreuses définitions insistent sur le fait que la domotique est un *art de vivre*, qui s'assimile davantage à une passion qu'à un système grand public. Toutefois, la démocratisation de la domotique peut apporter de nombreux avantages notamment concernant la gestion du handicap et du vieillissement. Ainsi, des projets visent à développer des solutions domotiques pour assurer le maintien à domicile des personnes âgées (PEETOOM et coll. 2014). Afin de faciliter son déploiement et son acceptation, il est nécessaire de masquer le côté avant tout technologique de la domotique, et c'est notamment l'objectif de l'intelligence ambiante.

L'intelligence ambiante, telle que définie par COUTAZ et CROWLEY (2008), est un « milieu ayant la faculté de percevoir, de raisonner, d'agir et d'interagir afin de fournir des services améliorant la qualité de vie des êtres vivants et notamment des personnes. » Cette capacité à percevoir et à agir repose sur l'informatique ubiquitaire introduite par WEISER (1991) comme une nouvelle ère de l'informatique. D'après WALDNER (2007) qui établit une topologie des différentes ères informatiques (Figure II.1), l'informatique ubiquitaire est la quatrième et l'intelligence ambiante la cinquième.

Ainsi, les années 1960 marquent la naissance de l'informatique durant laquelle les ordinateurs n'étaient que des unités simples permettant à leurs utilisateurs de traiter et stocker leurs données. Dix ans plus tard, les unités se sont mises à communiquer. L'ordinateur n'était alors plus une entité isolée, mais le membre d'un réseau de communication, partie

d'un ensemble de machines conscientes de l'existence les unes des autres. Les avancées technologiques liées au stockage de l'énergie ont ensuite permis aux ordinateurs de se libérer de la contrainte d'alimentation fixe, en embarquant leur propre source d'énergie dans des périphériques de plus en plus petits. Les unités mobiles, comme les ordinateurs portables, les PDA ou les téléphones mobiles, ont alors pu émerger durant la troisième ère de l'informatique.

D'après Weiser (1991), la miniaturisation et la multiplication des systèmes informatiques permettraient à terme d'embarquer la technologie dans la plupart des objets qui nous entourent de manière transparente : *"The most profound technologies are those that disappear"*. Or, la tendance actuelle aux objets connectés et l'émergence de l'Internet des Objets montrent bien la concrétisation de cette prédiction qui définit l'ère de l'informatique ubiquitaire. Les ordinateurs sont alors omniprésents et peuvent s'échanger des informations en continu, interagir les uns avec les autres.

Waldner (2007) conclut avec l'ère de l'intelligence ambiante qui ne peut être directement assimilée à l'informatique ubiquitaire. En effet, la définition énoncée précédemment stipule que le milieu doit être capable de raisonner et Ducatel et coll. (2001) avaient déjà insisté sur cette notion en définissant l'intelligence ambiante comme la convergence de l'informatique ubiquitaire et de l'intelligence artificielle. L'intelligence ambiante va alors permettre de réduire les connaissances technologiques nécessaires à la mise en place de systèmes domotiques. En apportant des capacités de raisonnement, elle va rendre les systèmes soit autonomes, soit plus accessibles (au point de rendre les objets naturellement utilisables par l'humain et non plus seulement par l'expert) et permettre ainsi à l'habitat d'être considéré comme un habitat intelligent. Il s'agira alors de déterminer comment les méthodes d'intelligence artificielle et notamment de prise de décision pourront être appliquées à un tel domaine.

II.1.2 Exemples d'habitats intelligents

Bien que le terme « habitat intelligent » soit très lié à l'intelligence ambiante et donc à l'intelligence artificielle et à l'informatique, les premiers habitats intelligents sont bien antérieurs à l'utilisation de ces technologies.

Gallissot (2012) présente notamment deux réalisations historiques, *Le prieuré* de Robert-Houdin et *La maison électrique* de Knap, datant respectivement de 1867 et 1913. La première, construite avant même l'avènement de l'électricité chez les particuliers, permettait à son habitant d'être notifié de certains évènements (comme le dépôt de courrier), rôle endossé aujourd'hui par les capteurs, mais également d'agir sur son environnement (par exemple en ouvrant la porte à distance) comme le font aujourd'hui les actionneurs. Gëorgia Knap, quant à lui, a tiré parti de l'essor de l'électricité dans les

FIG. II.2 : Plan de l'appartement réalisé sur chacun des deux étages de l'habitat *Aware Home* (KIDD et coll. 1999)

foyers aisés pour inventer de nombreux systèmes visant à faciliter la vie des habitants. Ainsi, bon nombre de ses innovations font aujourd'hui partie de notre quotidien comme le chauffe-eau ou le lave-vaisselle.

Il faudra ensuite attendre la fin du XX^{e} siècle pour voir apparaitre de nouveaux projets notoires avec notamment les projets *TRON House* (SAKAMURA 1990) et *Adaptative House* (MOZER 1998). La réticence liée à la maison autonome (et donc possiblement hors de contrôle de son habitant), observée notamment chez les médias, autour du projet de SAKAMURA (1990) sera un réel handicap quant à l'essor de telles installations. Ce qui n'empêchera cependant pas ces premiers travaux de poser les bases des recherches actuelles malgré le faible nombre de publications liées à ces projets. Par la suite, les habitats intelligents expérimentaux, qui servent de support expérimental aux études sur l'intelligence ambiante, vont se multiplier (KUSIAK 2007 ; NORMAN 1998).

Aware Home L'habitat *Aware Home* de l'Institut de Technologie de Géorgie (*Georgia Tech*, KIDD et coll. (1999)) a une vision pluridisciplinaire de l'intelligence ambiante et permet de nombreuses expériences technologiques ou humaines.

Sur une superficie de près de 470 m^2 répartis sur 2 niveaux, deux logements identiques ont été réalisés, chacun occupant un étage de la construction (Figure II.2). Chaque logement se compose d'une cuisine, une salle à manger, un salon, 3 chambres, 2 salles de bain et une buanderie. Un ensemble d'aménagements a également été réalisé pour rendre ces logements accessibles et permettre une installation et une maintenance aisée des différents dispositifs technologiques (gaines techniques, câbles pré-tirés, faux-plafonds).

House_n Le projet pluridisciplinaire *House_n* du Massachusetts Institute of Technology (MIT) (INTILLE 2002) couvre les domaines de l'informatique, de l'architecture, de la

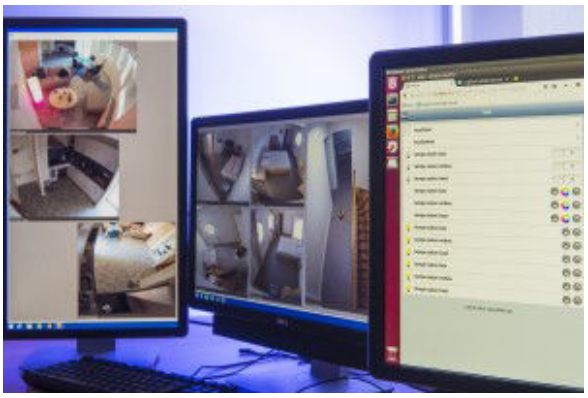
mécanique et de la médecine préventive. À travers le *Place Lab*, une habitation complètement équipée, il permet d'étudier le comportement des personnes et leurs interactions avec les objets et l'habitat. Ce projet a la particularité d'utiliser plusieurs centaines de capteurs, ce qui en fait un des environnements les plus mesurés. Parmi ces capteurs les principaux sont :

- les capteurs d'état sans fil placés au niveau des différentes portes (d'entrées ou de placards), des fenêtres, ainsi que sur les ustensiles de cuisine ;
- les dispositifs radio fréquence de localisation ;
- les microphones ;
- les caméras, y compris les caméras infrarouges ;
- les capteurs médicaux embarqués mesurant le rythme cardiaque, la tension artérielle, la respiration et le taux de glycémie.

The Intelligent Dormitory (iSpace) L'appartement *iSpace* de l'université d'Essex (Holmes, Duman et Pounds-Cornish 2002) a pour objectif d'étudier des méthodes non intrusives pour l'apprentissage des habitudes d'un habitant. Cet appartement type F2 suit l'agencement habituel des chambres étudiantes disponibles dans les résidences de cette université. Il est équipé de capteurs permettant d'enregistrer des données relatives à l'activité de l'occupant, ainsi que d'actionneurs destinés au pilotage du chauffage ou de la lumière.

SOPRANO et SM4ALL Les projets SOPRANO (*Service-Oriented Programmable Smart Environments for Older Europeans* (M. Klein, Schmidt et Lauer 2007)) et SM4ALL (*Smart hoMes for All* (Catarci et coll. 2009)) financés par l'Union Européenne, avaient pour objectif la réalisation de plateformes pour le contrôle d'un habitat intelligent. Bien que ces deux projets ne soient pas directement liés à la réalisation d'un tel habitat, ils démontrent une nouvelle fois la variété des recherches menées dans ce domaine. Ils ont notamment permis de tester différentes techniques liées à l'habitat intelligent, comme un intergiciel (*middleware*) pour l'intégration de capteurs hétérogènes au sein d'un habitat (Wolf, Schmidt et M. Klein 2008), ou des techniques d'interactions novatrices comme l'interaction cerveau-ordinateur (*Brain Computer Interaction* – BCI) permettant le contrôle par analyse d'activité neuronale (Aloise et coll. 2011).

(a) Salon



(b) Régie

FIG. II.3 : L'appartement d'Amiqual4Home peut accueillir un habitant pendant plusieurs semaines et récolter l'ensemble des données générées.

CASAS Le projet CASAS, de l'université de l'état de Washington (COOK et DAS 2007) est certainement l'un des projets les plus actifs. Le site web du projet[1] propose ainsi pas moins de 39 corpus regroupant les données issues de différents habitats intelligents. En mettant au point un kit minimaliste composé d'un ordinateur, 24 capteurs infrarouges, 1 capteur de porte, 2 relais et 2 thermomètres, l'équipe du projet a permis l'installation de nombreux habitats intelligents (COOK, CRANDALL et coll. 2013). De nombreuses études ont été menées dans le cadre de ce projet, notamment sur l'apprentissage des habitudes de l'utilisateur (RASHIDI et coll. 2011), ou la gestion de la consommation énergétique de l'habitat (C. CHEN et COOK 2012).

Amiqual4Home Le projet Amiqual4Home[2] regroupe de nombreux dispositifs dédiés à l'innovation. On y retrouve notamment un habitat intelligent (Figure II.3) équipé de plus de 219 capteurs et actionneurs, permettant de mesurer à la fois les conditions environnementales (température, humidité, etc.) et l'utilisation des objets de l'habitat. L'utilisation de l'intergiciel OpenHAB permet d'assurer l'agrégation des données à des fins de stockage mais également le contrôle à distance de tous les actionneurs de l'habitat. Récemment, un corpus d'activités a été enregistré et rendu publiquement accessible (LAGO et coll. 2017), à l'instar du projet CASAS.

Jarvis Dernièrement le projet Jarvis de ZUCKERBERG (2016) ébauche l'avenir de l'habitat intelligent. Il ne s'agit pas ici d'un projet académique mais de la réalisation personnelle d'un passionné ayant des connaissances technologiques. Ce projet directement installé et testé par l'utilisateur lui-même permet de contrôler l'éclairage, la température, un système

1. http ://casas.wsu.edu/
2. https ://amiqual4home.inria.fr/

multimédia ou des portes automatiques. Son utilisation intensive des dispositifs mobiles (et notamment du smartphone) et du réseau social Facebook® le rend particulièrement pertinent vis-à-vis de l'utilisation actuelle de la technologie.

HIS et Domus Plus proches de Grenoble, l'Habitat Intelligent pour la Santé (HIS) de la faculté de médecine de Grenoble (LeBellego et coll. 2006) et l'appartement Domus (Gallissot et coll. 2013) sur le campus de Grenoble sont des habitats intelligents permettant de réaliser des expérimentations dans des conditions proches du réel (*in vitro*). Ils ont notamment été utilisés pour les projets DESDHIS (Vacher, Portet et coll. 2011) et Sweethome (Vacher, Caffiau et coll. 2015) qui ont montré l'importance des capteurs acoustiques. En effet, dans ces réalisations, l'habitat doit prendre en compte les paroles prononcées par l'utilisateur pour détecter les appels de détresse ou prendre des décisions. L'utilisation de la parole, et en particulier de la parole spontanée, fait apparaitre une problématique de deixis[3], nécessitant la connaissance du contexte dans lequel est prononcé une phrase pour en déterminer toutes les entités.

Or, grâce à l'informatique ubiquitaire, les habitats intelligents sont en mesure d'extraire le contexte d'interaction des informations dont ils disposent. Mais, faut-il encore déterminer de quoi se compose ce contexte.

II.2 Le contexte

II.2.1 Définition

Lors de leurs interactions quotidiennes, les êtres humains sont capables de percevoir, évaluer, combiner et répondre à une multitude d'informations issues de l'ensemble de leurs canaux sensoriels (Vlachostergiou et coll. 2016) de manière à mieux appréhender cette interaction. Il a été montré que pour communiquer, les humains utilisent de nombreuses informations pour déterminer la situation courante (Knudsen et Muzekari 1983 ; Brown, Bovey et X. Chen 1997 ; Kaliouby et coll. 2003). À l'inverse, un humain risque de se trouver dans l'impossibilité d'interpréter correctement une conversation en l'absence de connaissances sur les conditions de son déroulement. Dès lors, il devient évident que les systèmes interagissant avec les humains, ce qui est notamment le cas des habitats intelligents, doivent prendre en compte des informations complémentaires, c'est-à-dire le *contexte* d'interaction.

La notion de *contexte* est relativement floue, chaque domaine ayant sa propre définition. La définition lexicographique (Centre National de Ressources Textuelles et

3. Notion linguistique se référant à une phrase qui ne peut être comprise sans son contexte, par exemples : « Regarde-*le* ! », « Donne-moi *ce* livre. ». en.wikipedia.org/wiki/Deixis

Lexicales (CNRTL) 2016) nous servira de point de départ pour définir ce qu'est le contexte :

> « Ensemble de circonstances liées, situation où un phénomène apparait, un évènement se produit. »

Cette définition est très générique, et ne nous permet que difficilement de nous rendre compte de ce que peut être le contexte dans l'habitat intelligent. Il faudra attendre la publication de Schilit, Adams et Want (1994) afin d'introduire le terme de *contexte* dans l'informatique et d'en poser une première définition :

> *"Where you are, who you are with, and what resources are nearby."*
>
> « Où vous êtes, avec qui vous êtes et quelles ressources sont à proximité. »

Cette définition par l'exemple, qui liste les éléments constitutifs du contexte, est trop restrictive pour Abowd et coll. (1999) qui proposent une nouvelle définition se fondant sur les différentes variantes proposées (Pascoe 1998) depuis les travaux de Schilit, Adams et Want :

> *"Context is any information that can be used to characterize the situation of an entity."*
>
> « Le contexte, c'est toute information qui peut être utile pour caractériser la situation d'une entité. »

Le terme *entité* désignant tout élément (personne, lieu, objet) pertinent dans l'interaction entre l'utilisateur et l'application.

À partir de cette définition, Abowd et coll. (1999) en extrapolent des caractéristiques primaires du contexte qui sont : la localisation, l'identité, l'activité et le temps. Ces caractéristiques permettent ensuite de déduire des caractéristiques secondaires liées à une entité. Plus récemment, Zimmermann, Lorenz et Oppermann (2007) réutiliseront cette définition en y ajoutant une caractéristique appelée *Relation* qui identifie les dépendances entre deux entités.

Toutefois, bien que largement acceptée par la communauté, cette définition reste très vague et d'autant plus difficile à préciser que le contexte utile dépend fortement de l'application. À l'heure actuelle, peu d'instances de contexte sont réutilisées d'une application à une autre et de nombreux problèmes se posent quant à sa normalisation et à sa modélisation de manière concrète et exhaustive.

TAB. II.1 : Taxonomie des contextes et de leurs utilisations selon DEY et coll. (1999).
Types de contexte : **A**ctivité, **I**dentité, **L**ocalisation, **T**emps.
Types d'application : **P**résentation, **E**xécution, **Ét**iquetage.

Système : description	Contexte				App.		
	A	I	L	T	P	E	Ét
Classroom2000 : enregistrement d'un cours			✔	✔			✔
Cyberguide : guide de visite		✔	✔		✔		
Stick-e Documents : guide de visite		✔	✔	✔	✔		✔
Reactive Room : contrôle intelligent de l'audio-visuel	✔	✔	✔			✔	
GUIDE : guide de visite			✔		✔		
Responsive Office : contrôle d'un bureau			✔	✔		✔	
NETMAN : maintenance du réseau			✔		✔		
Augment-able Reality : notes virtuelles			✔		✔		✔
Active Badge : transfert d'appels		✔	✔		✔	✔	

II.2.2 Types et représentation du contexte

Nous venons de le voir, la notion de contexte est très dépendante du domaine et de l'application à laquelle elle se rattache. ABOWD et coll. (1999) ont proposé une taxonomie permettant de classer les applications sensibles au contexte selon la nature du contexte utilisé et l'utilisation qui en est faite (Table II.1). Ainsi, sur les quelque quatorze projets étudiés à l'époque, la majorité reposait principalement sur la localisation de l'utilisateur comme caractéristique du contexte. Ce contexte était ensuite utilisé majoritairement pour adapter la présentation des informations et des services disponibles à l'utilisateur. L'objectif principal était alors de proposer une façon de représenter les connaissances liées au contexte, et de nombreuses représentations ont émergé.

Les approches logiques sont très couramment utilisées pour leur capacité à modéliser les relations entre entités mais également parce qu'elles permettent de réaliser des inférences automatiques. C'est notamment l'approche utilisée par MILEO, MERICO et BISIANI (2011) qui utilisent une méthode de programmation logique, *Answer Set Programming*, pour implémenter un système de prédictions de situations à risques. Une autre méthode, *Event calculus*, a été utilisée pour le développement d'habitats intelligents (L. CHEN et coll. 2008 ; KATZOURIS, ARTIKIS et PALIOURAS 2014) et permet de modéliser des situations en utilisant une logique du premier ordre.

Toutefois, la représentation la plus courante pour les environnements intelligents est aujourd'hui l'ontologie, en particulier celle basée sur la logique de description (*Description Logic* – DL). Le langage d'ontologie pour le Web (*Ontology Web Language* – OWL) est la principale implémentation de la logique de description et a été largement utilisée pour les environnements pervasifs (WOLF, SCHMIDT et M. KLEIN 2008 ; RODRÍGUEZ et coll.

2014b ; H.-C. Liao et Tu 2007). Les ontologies ont également servi à la représentation du contexte pour des applications liées aux périphériques mobiles (Attard et coll. 2013 ; Yilmaz et Erdur 2012). Pour leur capacité à modéliser l'imprécision ainsi qu'à raisonner sur des concepts aussi vagues que « beau temps », les ontologies floues ont également été largement expérimentées (Rodríguez et coll. 2014a ; Helaoui, Riboni et Stuckenschmidt 2013). Dans leurs travaux, Chahuara, Portet et Vacher (2017) utilisent un réseau logique de Markov (*Markov Logic Network* – MLN, Richardson et Domingos 2006) couplé à une ontologie en logique de description pour la reconnaissance de situation et la prise de décision. Ils cherchent ainsi à extraire différentes informations comme la localisation (une pièce parmi 4) et l'activité de l'utilisateur (une activité parmi 7 activités de la vie quotidienne (*Activity of Daily Living* – ADL)) (Chahuara 2013, p.130).

Enfin, d'autres travaux utilisent le contexte sans pour autant le représenter explicitement. C'est notamment le cas du projet *Adaptive Home*, présenté dans la publication de Mozer (1998). La principale application ayant entrainé une publication dans le cadre du projet *Adaptive Home* avait pour but de contrôler l'éclairage de l'habitat pour optimiser la consommation électrique de l'habitant. Le système ACHE, qui contrôlait l'habitat, utilisait pour cela un système de décision qui prenait en compte l'état de l'habitat. Cet état était estimé à partir de plusieurs informations qui semblaient centrales pour caractériser le contexte de l'habitant : l'activité de l'habitant et le niveau de lumière naturelle. Au premier abord, le contexte employé dans cette application utilise donc deux caractéristiques primaires. Toutefois, ne pouvant aisément déterminer l'activité à l'aide des capteurs disponibles, celle-ci est déduite à partir des derniers déplacements de l'utilisateur. En plus d'utiliser des capteurs de présence, l'application utilise un système d'anticipation pour prédire quelle pièce est sur le point d'être occupée en fonction des précédents mouvements de l'utilisateur. Ce système permet de prendre des décisions en avance et donc de supprimer la latence due aux limitations du matériel. Le niveau de luminosité naturelle est une caractéristique secondaire directement liée à la localisation de l'utilisateur. Cette application, comme beaucoup d'autres, se base alors principalement sur la localisation de l'utilisateur pour déterminer le contexte. Bien que cela semble suffisant dans ce cadre, Mozer (2005) précise que ces informations sont insuffisantes pour déterminer complètement l'état de l'environnement.

La grande diversité des approches utilisées montre bien qu'aucune ne fait consensus. En effet, les différents concepts à considérer sont trop disparates d'une application à une autre et ne permettent donc pas d'établir une ontologie universelle. C'est d'ailleurs ce qu'ont conclu Chikhaoui, Benazzouz et Abdulrazak (2009) en tentant de proposer une ontologie universelle pour le contexte : « En dépit des grands efforts fournis pour le développement d'une ontologie pour les environnements intelligents, le développement

d'une ontologie complète reste une tâche difficile ». C'est cette difficulté qui rend nécessaire le développement d'approches alternatives permettant de représenter une large gamme de concepts issus des habitats intelligents et des réseaux de capteurs.

Les contextes ainsi extraits peuvent ensuite être fournis à différentes applications leur permettant d'avoir davantage de connaissances à propos du monde avec lequel elles interagissent. C'est notamment le cas des projets de Chahuara, Portet et Vacher (2017) et Mozer (2005) qui fournissent le contexte à un système de décision, afin que celui-ci adapte sa décision. Dans la prochaine section, nous présenterons les systèmes de décision et comment ceux-ci peuvent prendre en compte ce contexte lors de la décision.

II.3 La prise de décision pour l'habitat intelligent

En comparaison avec les recherches sur le contexte ou la reconnaissance de situations, la prise de décision adaptative en habitat intelligent est un domaine qui compte peu d'acteurs. Toutefois, un regain d'intérêt semble émerger ces dernières années, notamment au travers de la gestion de l'énergie (Li et Jayaweera 2014 ; Berlink et Costa 2015). La *prise de décision* est par définition la sélection à un instant t de *l'action* à réaliser a_t, parmi un ensemble d'actions possibles $\mathcal{A}$, étant donné *l'état* actuel de l'environnement $s_t \in \mathcal{S}$ ($\mathcal{S}$ étant l'ensemble des états possibles de l'environnement). Pour cela, les systèmes de prise de décision utilisent la notion d'*utilité*.

Utilité Mesure quantitative permettant d'estimer la désidérabilité d'un état atteint suite à la réalisation d'une action (Morgenstern et Neumann 1944).

Il convient alors de définir une fonction d'utilité, notée $U : \mathcal{S} \to \mathbb{R}$, qui à chaque état de l'environnement lui associe une valeur d'utilité. L'objectif d'un agent dit *rationnel* est alors de sélectionner l'action a_t qui maximise l'utilité espérée (Russel et Norvig 2009) :

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, \mathbb{E}U(s_{t+1}) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}|s_t, a) U(s_{t+1}) \tag{II.1}$$

Où s_{t+1} est l'état atteint après l'exécution de l'action a_t sachant l'état actuel de l'environnement s_t et $P(s_{t+1}|s_t\,, a)$ est la probabilité d'atteindre l'état s_{t+1} en exécutant l'action a depuis l'état s_t, modélisant donc le comportement de l'environnement. Le contexte joue ici un rôle en nous permettant de déterminer l'état de l'environnement s_t soit directement (le contexte représente totalement l'état actuel), soit de manière indirecte, en extrapolant à l'aide de méthodes d'inférence.

La Table II.2 récapitule les caractéristiques des principaux travaux présentés dans cet état de l'art sur l'utilisation du contexte et la prise de décision. On y voit notamment

Tab. II.2 : Récapitulatif d'études liées à la représentation du contexte ou à la prise de décision

Publication	Type de contexte	Prise de décision	Représentation des données	Interface vocale	Adapt. continue	Décision en ligne	Habitat Intell.	Multiplexe	Utilisation
Mileo et coll. (2011)	Localisation, identité, objets	Non	Answer set programming	✗	✗	✗	✗	–	Données synthétiques
Katzouris et coll. (2014)	Activité	Non	Logique	✗	✗	✗	✗	–	Données synthétiques
Rodríguez et coll. (2014a)	Activité	Non	Ontologie floue	✗	✗	✓	✗	–	Données synthétiques
Helaoui et coll. (2013)	Activité	Non	Ontologie probabiliste	✗	✗	✓	✗	–	Données synthétiques
H.-C. Liao et coll. (2007)	Situations prédéfinies	Niveau de dangerosité	Ontologie	✗	✗	✗	✗	✗	Données synthétiques
Kofler et coll. (2012)	Situations prédéfinies	Raisonnement ontologique	Ontologie	✗	✗	✗	✗	–	Aucune
Gómez-Romero et coll. (2012)	Activité, situations prédéfinies	Reconnaissance de situation	Ontologie	✗	✗	✓	✓	✗	Séquence vidéo d'une personne
Nishiyama et coll. (2011)	Situations prédéfinies	Diagramme d'influence	Modèle bayésien	✗	✗	✗	✗	–	Données synthétiques
Chahuara et coll. (2017)	Localisation, activité, agitation, situations prédéfinies	Diagramme d'influence basé sur MLN	OWL2 et MLN	✓	✗	✓	✓	✓	37 utilisateurs naïfs
Mozer (2005)	Localisation, activité	Apprentissage par renforcement	Ontologie	✗	✓	✓	✓	✓	Usage quotidien
Karami et coll. (2016)	Activité	Apprentissage par renforcement	Ontologie	✗	✓	✗	✓	✓	Données synthétiques
Mnih et coll. (2015)	Aucun	Apprentissage profond par renforcement	Graphique	✗	✓	✓	✗	✗	Simulateur de jeu

les différents modèles utilisés mais également le manque d'expérimentations en conditions réelles. De plus, la majorité de ces travaux se concentre sur un problème particulier des systèmes sensibles au contexte, comme la reconnaissance d'activité, et ne considère alors pas la tâche à un niveau plus abstrait qu'est la prise de décision. Ce tableau fait apparaitre deux catégories d'approches que nous allons détailler : les approches statiques et les approches dynamiques.

II.3.1 La prise de décision statique

Parmi les différentes méthodes de prise de décision, les approches logiques sont très présentes dans la littérature. Moore, Hu et Jackson (2011) ont développé un système utilisant un ensemble de règles floues afin de sélectionner l'action appropriée à partir du contexte. De la même manière, Kofler, Reinisch et Kastner (2012) et Gómez-Romero et coll. (2012) utilisent la logique de description pour définir le comportement d'un système sensible au contexte qui modélise les éléments contextuels au travers d'une ontologie. D'autres travaux appliquent des règles de type Évènement-Condition-Action (ECA) pour développer des systèmes pour les environnements pervasifs (Leong, Ramli et Perumal 2009 ; Yau et Liu 2006). Ces méthodes ont notamment prouvé leur efficacité dans le milieu médical avec le système MYCIN (Shortliffe et Buchanan 1975) d'aide au traitement des infections bactériologiques. Toutefois, l'ensemble de ces systèmes utilise une approche qui s'apparente à un système expert (Bonarini et Maniezzo 1991). Une base de connaissances composée de règles logiques de type `SI… ALORS… SINON…` est rédigée par un expert du domaine concerné et un moteur d'inférence va raisonner en manipulant l'ensemble de ces règles. Ces approches présentent pourtant un inconvénient majeur. Le maintien de la cohérence entre les règles, la maintenance de ces règles, ou les preuves de complétude et de généralisation de ces systèmes rendent leur utilisation compliquée pour des applications pérennes.

Les réseaux bayésiens sont une autre méthode de prise de décision largement utilisée, comme l'a montré Lee et Cho (2012), mais qui ne représentent pas de manière formelle les éléments de la prise de décision. À l'inverse, les diagrammes d'influences (Howard et Matheson 1981) sont une méthode qui étend les réseaux bayésiens en incluant des variables particulières pour modéliser le processus de décision de manière plus appropriée. La Figure II.4 présente un diagramme d'influence utilisé par Chahuara (2013), dans lequel apparait un nœud représentant l'utilité d'un état (représenté par un losange). Dans ces travaux, l'objectif était d'allumer la lumière dans un habitat et le système devait choisir la localisation de la lampe à allumer et son intensité. L'utilité a été définie comme étant fonction du confort de l'utilisateur et de la pertinence du lieu d'éclairage. Cette dépendance est ainsi représentée par les liens entre l'utilité et ces variables aléatoires, elles-mêmes dépendantes d'autres facteurs. En effet, pour déterminer si la localisation de

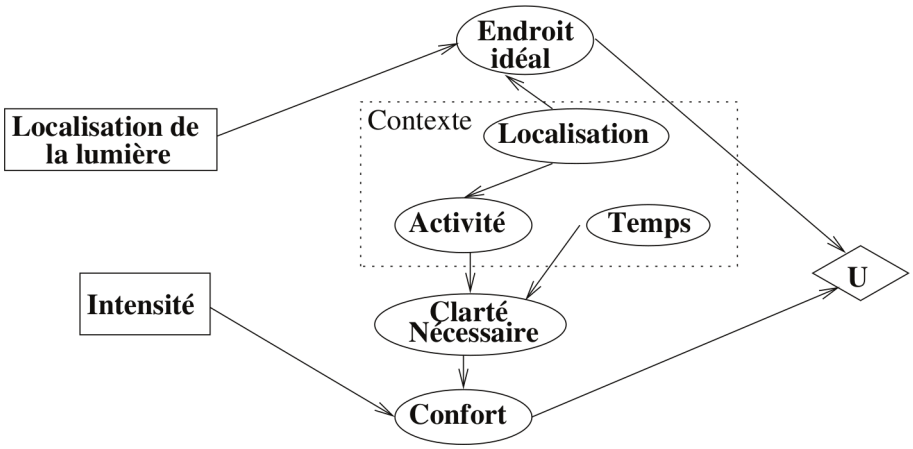


FIG. II.4 : Exemple de diagramme d'influence pour la prise de décision, tiré des travaux de CHAHUARA (2013).

la lampe à allumer est pertinente, il faut connaitre la localisation de l'utilisateur (en supposant que l'on souhaite allumer les lampes proches de l'utilisateur) et la localisation de la lampe. La représentation dans un rectangle de cette dernière indique que sa valeur sera directement dépendante de la décision qui sera prise. Les diagrammes d'influences ont été utilisés pour différents systèmes de prise de décision devant prendre en compte l'incertitude et être capables de mesurer l'utilité des différentes actions (NISHIYAMA, HIBIYA et SAWARAGI 2011 ; DE CAROLIS et COZZOLONGO 2004). Les principaux désavantages d'une telle approche sont sa faible expressivité, limitée à des dépendances conditionnelles entre les variables, ainsi que la représentativité du modèle généré. Alors qu'un ensemble de règles logiques peuvent être comprises et modifiées par un expert, une telle appréhension d'un modèle bayésien semble plus compliquée.

Les réseaux logiques de Markov (*Markov Logic Network* – MLN, RICHARDSON et DOMINGOS 2006) permettent de représenter de manière formelle, au travers de règles logiques, un processus de décision tout en prenant en compte l'incertitude en pondérant ces règles, combinant ainsi les avantages des précédentes méthodes. De plus, les pondérations de ces règles logiques peuvent être apprises de manière automatique à partir de données d'exemple. Cette approche a notamment été utilisée par CHAHUARA, PORTET et VACHER (2017) pour la prise de décision en habitat intelligent. Si cette méthode a prouvé son efficacité sur un corpus de données réelles, elle n'en nécessite pas moins la coopération d'un expert du domaine afin de définir l'ensemble des règles et comporte dans une moindre mesure les mêmes inconvénients que les approches logiques.

Enfin, l'ensemble de ces méthodes reposent sur un modèle déterminé en amont. Le système est ainsi configuré *a priori* pour exécuter une action étant donné une condition, mais ne peut pas adapter son comportement à des évolutions. Les approches dynamiques de prise de décision permettent cependant de répondre à cette limitation.

II.3.2 La prise de décision dynamique

La prise de décision dynamique permet à un système d'adapter continuellement son modèle afin de modifier son comportement si la fonction d'utilité venait à changer. La principale approche utilisée se fonde sur l'utilisation des ***processus de décision Markoviens*** (*Markov Decision Process* – MDP), qui permettent de modéliser un problème de décision séquentiel dans un environnement stochastique Markovien (Russel et Norvig 2009). Cette approche a notamment été utilisée par Mozer (1998) et Karami et coll. (2016) pour développer des systèmes adaptatifs de contrôle d'un habitat intelligent. Non liés à l'habitat intelligent mais utilisant également les MDP, les travaux de Mnih et coll. (2015) sont l'un des derniers grands succès d'une telle approche.

À l'instar des approches présentées plus tôt, les MDP basent également leur décision sur une notion d'utilité. Toutefois, la fonction d'utilité des états ne peut être définie de la même manière que pour un problème non séquentiel. En effet, l'utilité ne se définit plus pour un état seul, mais pour une séquence d'états $\{s_{t+1}, s_{t+2}, s_{t+3}, \ldots, s_{t+n}\}$. Plutôt que de définir une fonction d'utilité pour toutes les séquences possibles, on utilise une fonction de récompense $R : \mathcal{S} \to \mathbb{R}$ qui affecte une récompense à un état. On définit alors l'utilité d'une séquence d'états comme une somme pondérée selon un paramètre $\gamma \in [0\,,1]$ (Équation II.2).

$$\begin{aligned} U\Big(\{s_t, s_{t+1}, \ldots, s_{t+n}\}\Big) &= R(s_t) + \gamma R(s_{t+1}) + \ldots + \gamma^n R(s_{t+n}) \\ &= \sum_{i=0}^{n} \gamma^i R(s_{t+i}) \end{aligned} \tag{II.2}$$

Un MDP est donc un quintuplet $\langle \mathcal{S}\,, \mathcal{A}\,, P\,, R\,, \gamma \rangle$:

- $\mathcal{S}$: l'ensemble des états possibles de l'environnement ;
- $\mathcal{A}$: l'ensemble des actions possibles ;
- P : le modèle de transition de l'environnement, associant la probabilité d'arriver dans un état s_{t+1} après avoir effectué l'action a_t depuis l'état s_t ;
- R : la fonction de récompense, associant à chaque état un nombre réel ;
- γ : le paramètre d'atténuation de la récompense.

La solution d'un MDP est une ***stratégie*** notée $\pi(a_t|s_t)$ qui associe à une paire $\langle s_t\,, a_t \rangle$ la probabilité de choisir l'action a_t depuis l'état s_t. On trouve ici *l'hypothèse de Markov* selon laquelle seul l'état courant s_t est nécessaire au choix de l'action a_t :

$$\pi(a_t|s_t) = \pi(a_t|s_t, s_{t-1}, \ldots, s_0)$$

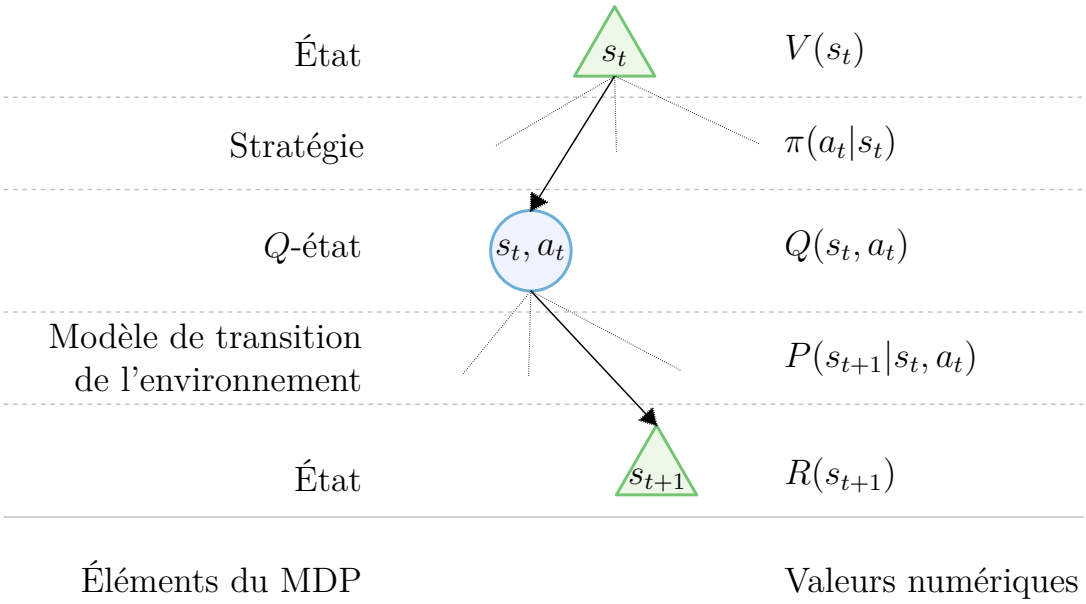


FIG. II.5 : Arbre de recherche d'un MDP pour une décision, faisant apparaitre les fonctions numériques associées aux différents états.

Il est alors possible de définir une ***fonction de valeur*** V dépendante de π qui estime l'utilité non plus d'une séquence, mais d'un seul état. Cette valeur est exprimée comme l'espérance statistique de l'utilité des séquences possibles à partir de l'état en fonction de la stratégie (Équation II.3). La fonction de récompense R représente alors une utilité sur le court terme, tandis que la fonction de valeur V représente une utilité sur le long terme, permettant de planifier et de prendre en compte les futurs possibles.

Une propriété fondamentale de la fonction de valeur est le fait qu'elle satisfasse une relation de récursivité qui permet d'exprimer la valeur d'un état en fonction de la valeur de ses successeurs. Cette relation est appelée *l'équation de Bellman* de la fonction V_π (Équation II.4) :

$$\begin{aligned} V_\pi(s_t) &= \mathbb{E}\left[U\left(\{s_{t+1}, \dots, s_{t_\tau}\}\right)\right] \\ &= \mathbb{E}\left[\sum_{i=1}^{t_\tau - t} \gamma^{i-1} R(s_{t+i})\right] \\ &= \mathbb{E}\left[R(s_{t+1}) + \gamma \sum_{i=1}^{t_\tau - t - 1} \gamma^{i-1} R(s_{t+i+1})\right] \end{aligned} \tag{II.3}$$

$$V_\pi(s_t) = \sum_{a_t}\left[\pi(a_t|s_t) \sum_{s_{t+1}} \left(P(s_{t+1}|s_t, a_t)\Big[R(s_{t+1}) + \gamma V_\pi(s_{t+1})\Big]\right)\right] \tag{II.4}$$

où t_τ (pouvant être infini) est l'instant à partir duquel toutes les décisions précédentes n'auront plus d'effet sur les décisions futures et définit ainsi la longueur d'un *épisode*.

Les décisions d'un MDP peuvent être représentées sous la forme d'un arbre de recherche (Figure II.5). On y voit apparaitre le choix de l'action $a_t \in \mathcal{A}$ déterminée en fonction de s_t

par $\pi(a_t|s_t)$. S'ensuit alors la transition de l'environnement, modélisée par la probabilité de transition $P(s_{t+1}|s_t\,,a_t)$, qui détermine le nouvel état en fonction de l'état précédent et de l'action choisie. Cette figure permet également d'introduire la notion de Q-état $(s_t\,,a_t)$, état intermédiaire représentant une décision possible, entre deux états de l'environnement, et sa valeur associée, la ***Q*-valeur** (Équation II.5). Le triplet $(s_t\,,a_t\,,s_{t+1})$ permet de définir complètement une transition.

$$Q_\pi(s_t,a_t)=\sum_{s_{t+1}}\Big(P(s_{t+1}|s_t,a_t)\Big[R(s_{t+1})+\gamma V_\pi(s_{t+1})\Big]\Big) \tag{II.5}$$

Résolution d'un MDP L'objectif lors de la résolution d'un MDP est de déterminer une stratégie optimale $\pi^\star$ qui maximise l'utilité de la séquence d'états produite lorsqu'on la suit. Les fonctions de valeurs associées à deux stratégies différentes permettent de les ordonner :

$$\pi>\pi' \iff \forall s\in\mathcal{S}, V_\pi(s)>V_{\pi'}(s)$$

On note donc la fonction de valeur optimale $V^\star(s)=\max\limits_\pi V_\pi(s)$.

Cette fonction de valeur respecte l'équation de Bellman présentée plus haut (Équation II.4). En revanche, son optimalité permet de réécrire cette équation sous une nouvelle forme, n'impliquant pas la stratégie (Sutton et Barto 2017) :

$$V^\star(s_t)=\max_a\sum_{s_{t+1}}P(s_{t+1}|s_t,a)\Big[R(s_{t+1})+\gamma V^\star(s_{t+1})\Big] \tag{II.6}$$

$$Q^\star(s_t,a_t)=\sum_{s_{t+1}}P(s_{t+1}|s_t,a_t)\Big[R(s_{t+1})+\gamma V^\star(s_{t+1})\Big] \tag{II.7}$$

Il existe différentes méthodes pour déterminer la stratégie optimale d'un MDP, dont ***l'itération sur la stratégie*** et ***l'itération sur la valeur***, toutes deux fondées sur les principes de la programmation dynamique telle que définie par Bellman (1957).

À partir d'une stratégie arbitraire π_0, **l'itération sur la stratégie** se décompose en deux phases :

- une première au cours de laquelle on évalue la stratégie courante π_k en calculant $V_{\pi_k}(s)$ à l'aide de l'Équation II.4 ;
- une seconde phase durant laquelle la stratégie est modifiée à partir des nouvelles valeurs des états (Équation II.8) pour générer une nouvelle stratégie π_{k+1} :

$$\pi_{k+1}(a_t|s_t)=\begin{cases}\dfrac{1}{|\mathcal{A}^\star|} & \text{si} \quad a_t\in\mathcal{A}^\star=\Big\{\underset{a}{\operatorname{argmax}}\sum\limits_{s_{t+1}}P(s_{t+1}|s_t,a)\big[R(s_{t+1})+\gamma V_{\pi_k}(s_t)\big]\Big\}\\ & \qquad\qquad\quad =\Big\{\underset{a}{\operatorname{argmax}}\,Q_{\pi_k}(s_t,a)\Big\}\\ 0 & \text{sinon}\end{cases} \tag{II.8}$$

où $\mathcal{A}^\star$ est l'ensemble des actions optimales (menant aux états de plus haute valeur ou ayant la meilleure Q-valeur étant donné l'état s_t).

Toutefois, la phase d'évaluation requiert un grand nombre d'itérations avant d'obtenir une évaluation fiable de la stratégie courante (la convergence n'est atteinte qu'aux limites, Sutton et Barto 2017, chap. 4.1) et est exécutée à chaque itération sur la stratégie.

La méthode **d'itération sur la valeur** se contente pour sa part d'une seule itération pour approximer la **fonction de valeur**. Les deux phases peuvent alors être écrites en une seule opération d'affectation relativement simple (Équation II.9).

$$V_{k+1}(s_t) = \max_a \sum_{s_{t+1}} P(s_{t+1}|s_t, a) \left[R(s_{t+1}) + \gamma V_k(s_{t+1})\right] \tag{II.9}$$

Malgré son apparente simplicité, cette méthode est assurée de converger vers la fonction de valeur optimale (Russel et Norvig 2009, sec. 17.2), d'où l'on pourra extraire la stratégie optimale grâce à l'extraction de stratégie (Équation II.10).

$$\pi^\star(a_t|s_t) = \begin{cases} \dfrac{1}{|\mathcal{A}^\star|} & \text{si} \quad a_t \in \mathcal{A}^\star = \left\{ \operatorname*{argmax}_a \sum_{s_{t+1}} P(s_{t+1}|s_t, a)\left[R(s_{t+1}) + \gamma V^\star(s_t)\right] \right\} \\ & \qquad\qquad = \left\{ \operatorname*{argmax}_a Q^\star(s_t, a) \right\} \\ 0 & \text{sinon} \end{cases} \tag{II.10}$$

Les MDP permettent de modéliser simplement le comportement d'un système de décision dans le monde réel. Toutefois, ils font l'hypothèse que l'agent connaisse totalement l'environnement sur lequel il agit. Or, ceci est rarement le cas dans le monde réel et encore moins lorsque l'humain est un élément de cet environnement. Pour prendre en compte cette lacune de connaissance, il est possible d'utiliser les processus de décision Markoviens partiellement observables (*Partially Observable Markov Decision Process* – POMDP, Sondik 1971) ou à observabilité mixte (*Mixed Observability Markov Decision Process* – MOMDP, Araya-López et coll. 2010).

Les POMDP sont une extension des MDP qui ne raisonne plus à partir de l'état actuel du système mais à partir d'un état de croyance, une distribution de probabilité sur l'ensemble des états de l'environnement (Cassandra 2016). Les MOMDP dérivent des POMDP en factorisant d'une part les variables cachées et d'autre part les variables visibles afin de tirer profit de l'efficacité des algorithmes traditionnels permettant la résolution des MDP. Toutefois, la multiplication de paramètres à prendre en compte rend l'utilisation de ces modèles dans les cas pratiques beaucoup plus coûteuse, voire infaisable selon les cas. C'est pourquoi peu d'approches les utilisent actuellement, les solutions proposées préférant généralement modéliser le comportement humain comme du bruit à prendre en

compte dans le modèle de transition d'un MDP classique (Zaidenberg et Reignier 2011).

Les processus de décision séquentiels sont particulièrement adaptés au problème de prise de décision dans un habitat intelligent qui est en perpétuelle évolution lorsque l'habitant l'utilise. Toutefois, ils doivent être totalement définis pour pouvoir être résolus et il s'avère que les fonctions de transition et de récompense d'un environnement réel ne sont pas triviales. Nous verrons en Section III.1.1 qu'il existe tout de même des méthodes permettant de résoudre ce problème.

II.4 Synthèse

Au cours de cet état de l'art nous avons défini les différents termes qui identifient les domaines au sein desquels s'ancre notre travail.

L'habitat intelligent est un habitat dans lequel l'intelligence ambiante, embarquée dans les objets du quotidien, va utiliser le contexte dans lequel évoluent ses habitants pour améliorer leur qualité de vie. Ce *contexte* peut prendre différentes formes et servir plusieurs objectifs mais nous nous cantonnons dans notre cas à la prise de décision. Cette *prise de décision* prend place dans un environnement réel, soumis à différents aléas, dans lequel évoluent des agents dont on ignore les intentions, un environnement incertain.

Toutefois, cette présentation a également soulevé certaines difficultés. En effet, les cas d'usage d'un habitat intelligent sont nombreux et ne peuvent être définis de manière exhaustive en amont de la réalisation d'un système. De plus, de tels systèmes reposent généralement sur la notion de contexte afin de fournir le service adéquat au moment adéquat. Cependant, aucun consensus n'a jusqu'alors permis de trouver une représentation normalisée de ce contexte, rendant son utilisation problématique. Enfin, l'incertitude liée à la réalité de l'interaction (fiabilité des capteurs, interventions humaines, etc.) complique les systèmes de prise de décision. Cette incertitude devant être prise en compte, tout comme doivent l'être les futurs possibles, afin de déterminer l'action la plus judicieuse. Ces constats nous ont poussé à développer lors de ces travaux de thèse un système capable de répondre à ces différentes problématiques.

Les études publiées par Zaidenberg et Reignier (2011) montrent qu'un utilisateur s'attend à ce qu'un système intelligent soit capable d'adapter son comportement au cours du temps, afin de prendre en compte ses préférences. Pour cela, les méthodes d'apprentissage automatique dites incrémentielles permettent à un modèle d'évoluer au cours du temps. L'apprentissage actif (Settles 2012) est l'une de ces méthodes, mais à l'instar des autres méthodes d'apprentissage supervisé, elle requiert un investissement non négligeable pour

la création de corpus finement annotés. Zaidenberg et Reignier (2011) montrent également que si l'utilisateur est prêt à guider le système lors de son apprentissage, ce support ne doit pas être trop intrusif. L'apprentissage par renforcement, déjà expérimenté par Mozer (2005), permet ainsi de tirer parti de l'intelligence de l'utilisateur pour construire un modèle adapté à ses habitudes et préférences, et ceci de la manière la moins intrusive possible. L'utilisation de cette méthode sera l'objet du Chapitre III sur la réalisation d'un système de décision adaptatif.

L'utilisation de l'apprentissage par renforcement ne permet cependant pas de répondre à la question de la représentation des données et du contexte. Les récents travaux de Mnih et coll. (2015) ont ouvert une nouvelle voie en proposant un système fonctionnel couplant à la fois l'apprentissage par renforcement et l'apprentissage profond. Dans le Chapitre IV nous présenterons alors notre solution, reposant sur un modèle d'apprentissage par renforcement profond, permettant une grande liberté quant aux types et à la quantité de données représentables ainsi que de s'abstraire de la représentation du contexte.

Chapitre III

Une première approche pour l'adaptation continue

La connaissance s'acquiert par l'expérience.

Albert Einstein

L'objectif de ce travail de thèse est le développement d'un contrôleur intelligent adaptatif pour l'habitat intelligent. L'utilisation d'un tel système sur le long terme implique que celui-ci soit capable de s'adapter à différents aléas, comme le changement de topologie de l'habitat et de ses capteurs ou le changement d'habitudes et de préférences de l'utilisateur. De telles contraintes rendent difficilement utilisables des systèmes pré-configurés reposants sur des règles logiques ou des ontologies qui sont des solutions trop rigides et non adaptées pour des systèmes ayant une longue durée de vie.

Les évolutions technologiques de ces dernières années permettent aujourd'hui d'équiper les habitats de nombreux capteurs capables de récolter des données à la fois environnementales et sur l'utilisation des différents dispositifs de l'habitat. La quantité de données ainsi recueillies laisse envisager l'utilisation de méthodes statistiques dont l'objectif est de déterminer et d'identifier des régularités dans ces données. Dans le cas de la prise de décision, les approches supervisées, associant une étiquette à chacune des entrées, semblent plus adaptées que les approches non-supervisées qui nécessitent une expertise *a posteriori* pour exploiter les résultats. Toutefois, l'utilisation de méthodes d'apprentissage supervisé pose deux contraintes :

- les étiquettes (les actions dans le cas de la prise de décision) doivent être définies à l'avance ;

- les exemples doivent être étiquetés pour être exploités par le système.

La première contrainte est incompatible avec notre système puisqu'à l'instar des approches logiques, elle réduit la capacité d'adaptation de notre système en restreignant son champ d'application. La seconde est un réel frein à l'adoption d'un système. Elle nécessiterait en effet que l'utilisateur annote tout au long de l'utilisation du système des milliers d'exemples (contexte et action attendue) avant de relancer une (longue) phase d'apprentissage à partir de ces exemples. Comme nous l'avons déjà précisé, Zaidenberg et Reignier (2011) ont montré qu'il était nécessaire d'inclure l'utilisateur dans l'apprentissage du système, mais ceci de la manière la moins intrusive possible.

Les méthodes d'apprentissage semi-supervisé peuvent alors être une alternative. Par exemple, l'apprentissage actif (Settles 2012) utilise les capacités de l'apprentissage non-supervisé pour regrouper les exemples similaires afin de ne nécessiter que d'une seule étiquette pour un groupe. Même si cette méthode permet de réduire le poids de l'annotation manuelle, elle ne résout pas le problème de sélection *a priori* des étiquettes. À l'inverse, les méthodes d'apprentissage par renforcement nécessitent un grand nombre d'exemples, mais annotés seulement avec un indicateur de la qualité de la décision (une récompense). Cette méthode, notamment utilisée par Mozer (2005), peut alors être mise en place dans un habitat intelligent, moyennant un mécanisme de récompense qui ne requiert pas une forte implication de l'utilisateur. Par exemple, dans le projet *Adaptive House*, Mozer (2005) utilise un mécanisme simple : si l'utilisateur intervient sur l'un des choix réalisés par le système, alors un coût (utilisé comme récompense) est appliqué à l'action réalisée par le système. Les expérimentations présentées dans la suite de ce mémoire n'utilisent pas de tels mécanismes car non pertinents dans notre cas, où l'expérimentation n'est pas réalisée de manière interactive. En revanche, la présence de microphones dans l'habitat laisse entrevoir de nombreuses façons de déterminer une récompense, via une interaction vocale, si notre système venait à être utilisé en conditions réelles.

Dans ce chapitre, nous présenterons Section III.1 comment l'apprentissage par renforcement peut être implémenté et utilisé pour la prise de décision dans un habitat intelligent, avant de présenter Section III.2 notre expérimentation et les résultats que nous avons obtenus.

III.1 Apprentissage par renforcement pour l'habitat intelligent

L'*apprentissage par renforcement* (*Reinforcement Learning* – RL) est une technique très connue des psychologues qui l'étudient depuis la fin des années 1930. En 1938, Skinner

publie les résultats de ses recherches sur les conséquences d'une récompense ou d'une punition sur l'évolution du comportement d'un sujet. Cette méthode est par la suite appliquée à l'apprentissage automatique pour « apprendre ce qu'il faut faire en l'absence d'exemple de ce qu'il faut faire » (Russel et Norvig 2009).

D'après Sutton et Barto (2017), il existe trois caractéristiques qui permettent d'identifier un problème particulièrement adapté à l'utilisation de l'apprentissage par renforcement :

1. s'il s'agit d'un problème en boucle fermée dans lequel les actions de l'agent auront une influence sur ses entrées futures ;
2. s'il n'existe pas d'étiquette fournie à l'agent pour lui dire quelle action il aurait dû effectuer, mais seulement un moyen d'évaluer l'erreur ;
3. si l'action choisie n'influence pas seulement la récompense immédiate mais également toutes les récompenses futures.

L'utilisation d'une telle approche dans un habitat intelligent adaptatif semble donc particulièrement adaptée. D'une part, en agissant sur certains dispositifs de l'habitat (comme les lampes ou les volets) le contrôleur intelligent modifie l'état de l'environnement, état qui lui sert de donnée d'entrée lors de sa décision. D'autre part, afin d'éviter un coût d'annotation trop élevé et de conserver notre capacité d'adaptation, nous ne souhaitons utiliser qu'un signal d'évaluation des actions en lieu et place d'annotations plus explicites. Enfin, comme souvent dans les cas réels, une action n'a pas seulement un impact immédiat, mais également des effets sur plus long terme, influençant ainsi les états, les décisions et les récompenses futures.

Sutton et Barto (2017) définissent également deux composantes principales d'un système d'apprentissage par renforcement :

1. l'***agent***, qui est l'élément qui apprend et qui prend les décisions (dans notre cas il s'agit du système de décision) ;
2. l'***environnement***, tout ce qui est extérieur à l'agent et avec lequel celui-ci interagit (ici, l'habitat et son habitant).

Comme le montre la Figure III.1, ces deux éléments interagissent continuellement dans une boucle au cours de laquelle, l'agent choisit une action et l'environnement répond à cette action en présentant une nouvelle situation à l'agent. L'environnement va également

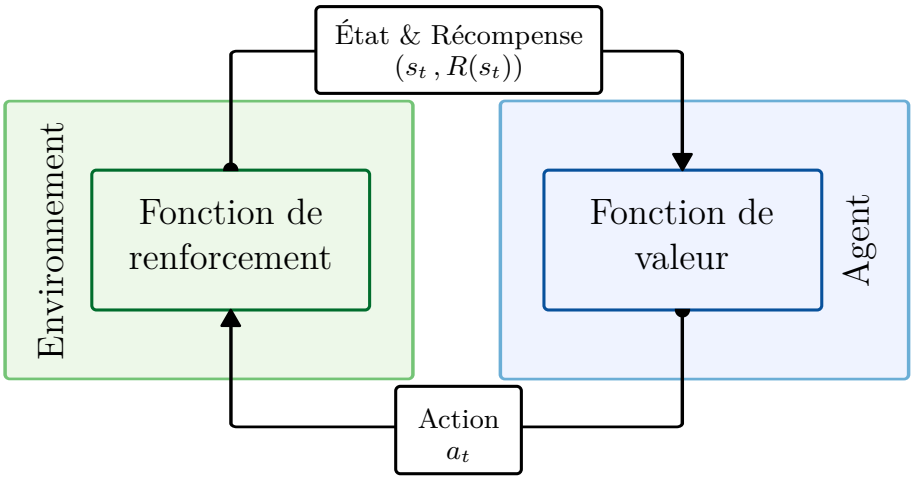


FIG. III.1 : Boucle d'action—réaction pour l'apprentissage par renforcement entre un agent et son environnement.

émettre un signal de récompense, une valeur réelle que l'agent va devoir maximiser au cours du temps.

Un problème d'apprentissage par renforcement est donc directement lié à un **processus de décision Markovien** (MDP – Section II.3.2) puisqu'il peut se définir de manière similaire : un ensemble d'états d'environnement $\mathcal{S}$, un ensemble d'actions $\mathcal{A}$, une distribution de transitions P et une fonction de récompense R et son paramètre d'atténuation γ. Il convient alors de déterminer la stratégie optimale de l'agent.

En revanche, dans le cas d'un problème d'apprentissage par renforcement, le modèle de transition et la fonction de récompense (définissant la fonction de valeur) ne sont pas connus. Il est donc impossible de raisonner *a priori* sur des connaissances que nous n'avons pas. Tout l'enjeu est alors d'obtenir des connaissances permettant de déterminer la meilleure stratégie à partir d'expériences.

III.1.1 Méthodes d'apprentissage

Pour la résolution d'un problème d'apprentissage par renforcement, deux grandes classes de méthodes peuvent être identifiées.

Apprentissage par modélisation Les approches les plus simples, basées sur un modèle (*model-based learning*), vont chercher à déterminer les paramètres du MDP à partir de leurs expériences. En interagissant avec l'environnement, l'agent va construire un ensemble d'interactions $\mathcal{I} = \big\{(s_0\,,a_0\,,s_1)\,,\ldots,(s_{n-1}\,,a_{n-1}\,,s_n)\big\}$. Cet ensemble va permettre d'estimer la distribution sous-jacente au modèle de transition et la fonction de récompense. Une fois le MDP totalement défini, les méthodes **d'itération sur la stratégie**

ou **d'itération sur la valeur**, présentées Section II.3.2, permettront de déterminer la stratégie optimale (D. Klein et Abbeel 2013).

Une telle approche pose cependant un problème. Il est difficile de déterminer à quel moment le MDP est totalement défini, et donc à partir de quel moment il est possible d'exploiter ses connaissances pour en déterminer la stratégie. De plus, tout au long de la première phase, la *phase d'exploration* (décrite par l'Équation II.4), aucune connaissance pouvant être extraite de l'expérience n'est utilisée. L'agent agit de manière erratique jusqu'à la *phase d'exploitation* des connaissances (Équation II.8).

Apprentissage sans modèle À l'inverse des méthodes basées sur un modèle, les méthodes sans modèle (*model-free learning*) vont chercher à déterminer directement la fonction de valeur d'un MDP sans chercher à approximer le modèle de transition et la fonction de récompense. Ces méthodes peuvent être divisées en deux classes différentes, les méthodes passives (contraintes par la stratégie) et les méthodes actives (sans stratégie).

Les méthodes passives utilisent une stratégie fournie initialement. La plus simple, *l'évaluation directe*, va agir en suivant la stratégie et calculer les valeurs des états à la fin des épisodes ($t = t_\tau$) en faisant la moyenne des sommes pondérées des récompenses à chaque fois que cet état a été rencontré (c'est-à-dire la moyenne des différentes utilités calculées pour chaque état). Cette méthode très simpliste ne tient pas compte des transitions d'un état à l'autre et ne mutualise pas l'information entre des états successifs. De fait, elle est relativement longue à converger.

L'évaluation de stratégie (Équation II.4) permet de prendre en compte les liens entre les états. Ainsi, ***l'évaluation de stratégie échantillonnée*** utilise une affectation similaire pour déterminer une fonction de valeur à partir des échantillons rencontrés. Pour toute interaction (s, a, s'), il est possible de calculer une valeur approchée v de la fonction de valeur qui permettra d'améliorer l'estimation actuelle de la valeur d'un état :

$$\left.\begin{aligned} V_k(s) \approx v_1 &= R(s'_1) + \gamma V_k(s'_1), \\ V_k(s) \approx v_2 &= R(s'_2) + \gamma V_k(s'_2), \\ &\vdots \\ V_k(s) \approx v_n &= R(s'_n) + \gamma V_k(s'_n) \end{aligned}\right\} \quad V_{k+1}(s) \leftarrow \frac{1}{n}\sum_i v_i \tag{III.1}$$

Cette approche est intéressante mais irréalisable dans les cas réels. En effet, il est en pratique impossible d'obtenir différents échantillons à partir d'un même état de manière systématique puisque l'environnement peut évoluer indépendamment de la volonté de l'agent.

Une méthode très largement documentée (Sutton et Barto 2017, chap. 6) est ***l'apprentissage par différence temporelle*** (*Temporal Difference (TD) Learning*). Cette

méthode permet, en utilisant une moyenne courante de modifier la valeur $V(s)$ chaque fois qu'une transition $(s\,,a\,,s')$ est rencontrée. On utilise pour cela un coefficient d'apprentissage α qui détermine quel poids possède la nouvelle expérience par rapport aux connaissances acquises depuis le début de l'interaction entre l'agent et son environnement.

$$v = R(s') + \gamma V_k(s') \quad \text{d'où} \quad \begin{aligned} V_{k+1}(s) &\leftarrow (1-\alpha)V_k(s) + \alpha v \\ &= V_k(s) + \alpha(v - V_k(s)) \end{aligned} \tag{III.2}$$

Cette approche imite les approches fondées sur les équations de Bellman (Équation II.4) et permet d'évaluer une stratégie donnée. Toutefois, il n'est pas possible de déterminer une nouvelle stratégie à partir de la fonction de valeur calculée. En effet, l'Équation II.10 rappelée ci-dessous montre qu'en plus de la fonction de valeur il est nécessaire de connaitre le modèle de transition pour déterminer la stratégie.

$$\pi^\star(a_t|s_t) = \begin{cases} \dfrac{1}{|\mathcal{A}^\star|} & \text{si} \quad a_t \in \mathcal{A}^\star = \left\{ \underset{a}{\operatorname{argmax}} \sum_{s_{t+1}} P(s_{t+1}|s_t,a)\Big[R(s_{t+1}) + \gamma V^\star(s_t)\Big] \right\} \\ & \qquad\quad = \left\{ \underset{a}{\operatorname{argmax}}\, Q^\star(s_t,a) \right\} \\ 0 & \text{sinon} \end{cases}$$

Une solution à ce problème est d'appliquer une approche similaire pour déterminer la fonction de Q-valeur qui nous permettra d'en déduire une nouvelle stratégie.

Le **Q-Learning** est un algorithme popularisé par Watkins (1989) permettant de déterminer la fonction de Q-valeur à partir d'expériences de l'agent. Il s'agit d'une méthode d'apprentissage actif dans laquelle l'agent ne va pas simplement suivre une stratégie qui lui est imposée, mais choisir à chaque état l'action qu'il souhaite expérimenter. De la même façon que l'itération sur la valeur permet d'obtenir une fonction de valeur optimale (Équation II.9), il est possible de déterminer une méthode d'itération sur la Q-valeur pour obtenir la fonction de Q-valeur optimale :

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)\left[R(s') + \gamma \max_{a'} Q_k(s',a')\right] \tag{III.3}$$

Le parallèle entre fonction de valeur et fonction de Q-valeur peut alors s'étendre à l'apprentissage par différence temporelle, que l'on peut appliquer à la fonction de Q-valeur (Équation III.4) :

$$q = R(s') + \gamma max_{a'} Q_k(s',a') \quad \text{d'où} \quad \begin{aligned} Q_{k+1}(s,a) &\leftarrow (1-\alpha)Q_k(s,a) + \alpha q \\ &= Q_k(s,a) + \alpha(q - Q_k(s,a)) \end{aligned} \tag{III.4}$$

Cette méthode est aujourd'hui une des plus utilisées et a démontré sa robustesse. En effet, le *Q-Learning* a la particularité de converger vers la stratégie optimale même si l'agent agit

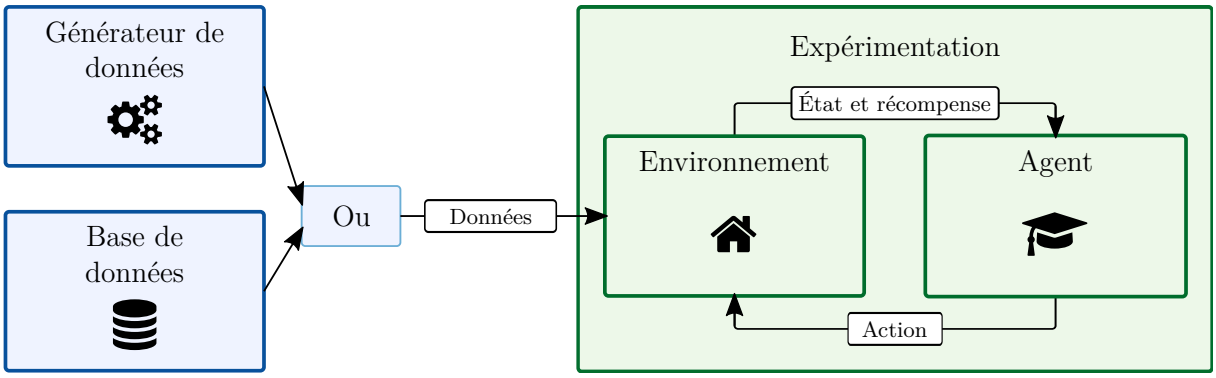


FIG. III.2 : Interactions entre les composants de notre système

de manière sous-optimale dans son exploration des actions. En revanche, en supprimant la contrainte sur le choix de l'action, le temps d'exploration pour obtenir des informations pertinentes rend le temps de convergence plus long que dans le cas des méthodes passives. De plus, cette capacité d'exploration fait apparaitre la problématique de partage entre les phases d'exploration et d'exploitation des connaissances.

Il existe différentes méthodes pour forcer l'agent à explorer son environnement. La plus courante, la méthode ϵ-*greedy*, agit de manière aléatoire (exploration) avec une (faible) probabilité ϵ, et agit suivant une stratégie gloutonne qui sélectionne la meilleure action (exploitation) avec une probabilité $(1-\epsilon)$. Nous ne détaillerons pas ici les autres méthodes d'exploration, avec notamment les fonctions d'exploration, et le lecteur pourra se référer au livre de SUTTON et BARTO (2017) pour de plus amples informations.

III.1.2 Implémentation

La section précédente présentait le cadre théorique de la résolution d'un problème d'apprentissage par renforcement. Ici, nous détaillerons l'implémentation que nous avons faite du *Q-Learning*, avant de présenter notre expérimentation et ses résultats. Cette implémentation repose sur un cadre de travail existant nommé `PyBrain` (SCHAUL et coll. 2010) et qui fournit des mécanismes de base communs à la majorité des problèmes de renforcement. La Figure III.2 présente les différents composants à implémenter. Les éléments en vert (expérimentation, environnement et agent) sont les composants standards que l'on retrouve dans la majorité des implémentations liées à l'apprentissage par renforcement.

L'expérimentation L'expérimentation, parfois appelée *tâche*, est un composant absent de la définition théorique de l'apprentissage par renforcement, revient régulièrement dans les implémentations et permet principalement de gérer les interactions entre l'agent et son environnement. Il est chargé du respect de la logique d'interaction présentée Figure III.1, grâce à l'Algorithme III.1.

Algorithme III.1 Logique d'expérimentation permettant de faire interagir un agent avec son environnement.

Require: $epoch = 0$
  while $epoch < maxEpoch$ **do**
    $step \leftarrow 0$
    repeat
      $step \leftarrow step + 1$
      observation $\leftarrow$ **environment**.getObservableState
      agent.integrateObservations(observation)
      action $\leftarrow$ **agent**.getAction
      environment.performAction(action)
      reward $\leftarrow$ **environment**.getReward
      agent.giveReward(reward)
    until $step > maxSteps$
    agent.learn
    $epoch \leftarrow epoch + 1$
  end while

Le système récupère les données issues de l'environnement et les transmet à l'agent qui les enregistre. Le système peut alors demander à l'agent l'action qu'il souhaite réaliser étant données les nouvelles informations reçues, et exécuter cette action dans l'environnement. Enfin, le système récupère la récompense émise par l'environnement et la transmet à l'agent. Le système réitère alors ces trois étapes, l'agent sauvegardant chacune des interactions qu'il expérimente. Après avoir réalisé un nombre *maxSteps* d'interactions, l'agent peut entrer dans une phase d'apprentissage au cours de laquelle il extrait des connaissances à partir de son expérience. Cette ère (*epoch*) d'apprentissage est réitérée jusqu'à ce qu'une quantité *maxEpoch* d'ères aient été réalisées.

Appliquée à notre exemple présenté en introduction (Scénario 1), l'exécution d'un tel algorithme peut être décrit comme suit :

- le système remonte les informations de l'environnement et les transmet à l'agent : il est 03 :00 du matin, l'utilisateur, qui était en train de dormir dans sa chambre, demande à allumer la lumière.
- l'agent doit choisir la meilleure action à réaliser : il juge que le plus judicieux est d'allumer la lampe de chevet avec une intensité modérée.
- cette action est appliquée et sans retour négatif de la part de l'utilisateur, une récompense positive est émise et transmise à l'agent.
- lors de son apprentissage, l'agent se souviendra de l'état de l'environnement, de l'action qu'il a réalisée et de la récompense positive qu'il a obtenue, et renforcera donc son comportement actuel dans un tel contexte.

L'agent Il est le composant apprenant de notre système et celui qui implémente la méthode d'apprentissage. L'implémentation habituelle repose sur 4 opérations que l'agent doit pouvoir exécuter. Les plus simples consistent à enregistrer l'état actuel de l'environnement ainsi que la récompense obtenue pour une utilisation future. L'opération permettant d'obtenir l'action à réaliser est quant à elle dépendante de la stratégie de l'agent. Dans notre cas, nous utilisons une stratégie de type ϵ-*greedy* déjà évoquée plus tôt et dont l'implémentation est donnée Algorithme III.2, où $\mathcal{U}$ est une fonction de sélection aléatoire suivant une distribution uniforme.

Algorithme III.2 Stratégie de décision de l'agent

Require: $\epsilon \in [0\,;1]$
  $x \sim \mathcal{U}([0\,,1])$
  if $x \leq \epsilon$ **then**
    action $\sim \mathcal{U}(\mathcal{A})$ ▷ Action aléatoire
  else
    Q-values $\leftarrow Q$-table$[state]$
    action $\leftarrow$ argmax(**Q**-values)
  end if
  return action

Cet algorithme fait apparaitre la notion de Q-*values* et de Q-*table*. En effet, la stratégie d'un MDP est dépendante de la fonction de Q-valeur des Q-états de cet MDP. Un agent doit donc connaitre cette fonction de Q-valeur. Cependant, dans un problème d'apprentissage par renforcement, cette fonction n'est pas connue et l'agent doit donc l'approximer. Une méthode commune pour approximer cette fonction est d'utiliser un tableau à double entrées, appelé ici Q-*table*. Ce tableau est indexé d'une part par les états $s \in \mathcal{S}$ de l'environnement et d'autre part par les actions $a \in \mathcal{A}$ disponibles. Ainsi, la valeur $Q\text{-table}_{s,a}$ est égale à la valeur $Q(s\,,a)$.

La dernière opération permet à l'agent d'apprendre de son expérience. Pour cela, il parcourt l'ensemble des interactions $\langle s_t\,,a_t\,,r_t\,,s_{t+1}\rangle \in \mathcal{I}$ qu'il vient d'expérimenter et applique l'Équation III.4 pour mettre à jour les valeurs de la table de Q-valeurs (la Q-*table*) comme le montre l'Algorithme III.3.

Algorithme III.3 Méthode d'apprentissage de l'agent

  for all $\langle s\,,a\,,r\,,s'\rangle \in \mathcal{I}$ **do**
    $q_{old} \leftarrow Q\text{-table}[s][a]$
    $q \leftarrow r + \gamma \max(Q\text{-table}[s'])$
    $q_{new} \leftarrow q_{old} + \alpha(q - q_{old})$
    $Q\text{-table}[s][a] \leftarrow q_{new}$
  end for

L'environnement Ce composant logiciel sert d'interface pour l'environnement réel (l'habitat et son habitant dans notre cas) sur lequel agit l'agent. Il permet d'en masquer la complexité en ne proposant ainsi que trois opérations de base. La première permet de faire remonter l'état (observable) actuel de l'environnement. Le plus souvent, cet état correspond à la valeur actuelle de l'ensemble des capteurs de l'environnement mais peut également être le résultat de traitements sur ces valeurs, comme nous le verrons par la suite. La seconde opération permet d'appliquer une action sur l'environnement, c'est-à-dire d'agir sur les actionneurs afin de modifier leurs états, influençant de fait l'état de l'environnement. Enfin, la dernière opération permet de récupérer la récompense, qui comme nous l'avons vu, peut être calculée de différentes façons, induite par la dernière action.

L'implémentation d'un tel composant est alors assez simple. Toutefois, notre expérimentation ne se déroulant pas en conditions réelles mais sur un corpus de données, il est nécessaire d'apporter quelques modifications à ce composant. D'une part, l'état de l'environnement ne peut être mesuré par les capteurs, mais récupéré à partir d'un corpus enregistré ou généré automatiquement, comme le montre la Figure III.2. D'autre part, la réalisation d'une action n'a pas d'impact sur le jeu de données pré-enregistré de même que l'utilisateur ne peut générer de récompense. Il est donc nécessaire de simuler le comportement de l'environnement et de définir une fonction de récompense. Nous implémentons pour cela deux algorithmes présentés dans l'Algorithme III.4.

Algorithme III.4 Algorithmes permettant de simuler la dynamique de l'environnement sur un corpus pré-déterminé.

```
function performAction(predictedAction)
    if predictedAction est trueAction ∨ numberOfTries > triesThreshold then
        moveToNextDataSample
        numberOfTries ← 0
    else
        numberOfTries ← numberOfTries + 1
    end if
end function
function getReward
    if predictedAction était trueAction then
        reward ← +1
    else
        reward ← −1
    end if
    return reward
end function
```

Dans la première fonction de cet algorithme, la notion de tentatives ($numberOfTries$) apparait. Cette valeur permet de simuler la patience de l'utilisateur, autorisant le système

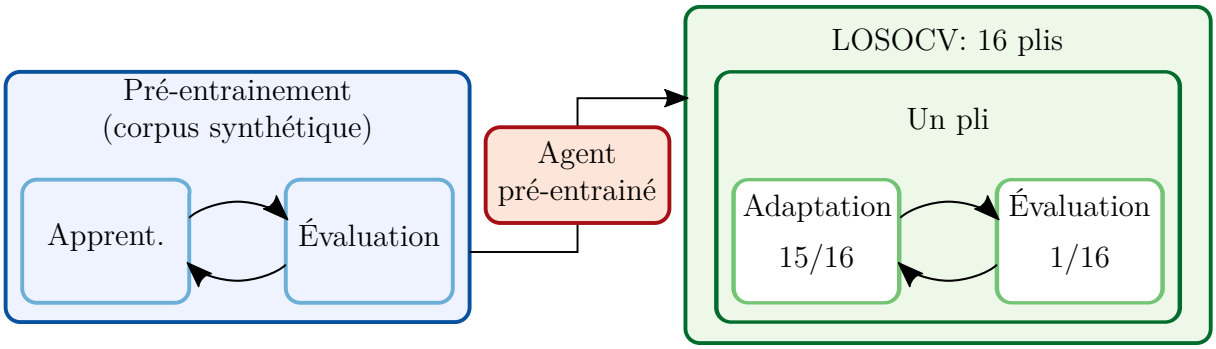


FIG. III.3 : Déroulement de l'expérimentation

à réessayer si l'action exécutée n'est pas celle attendue. La fonction de récompense est ici très simpliste, retournant une récompense positive si l'action est exactement celle attendue, et négative sinon. D'autres fonctions ont été testées mais ne semblent pas impacter de manière notable les performances du système.

III.2 Expérimentation préliminaire

Après avoir mis en place l'ensemble des composants, nous avons cherché à évaluer notre système pour valider la pertinence d'une telle approche. Le déroulement de cette expérimentation est détaillé Section III.2.1. Les données utilisées sont présentées ensuite Section III.2.2, avant l'étude des résultats Section III.2.3. Ces résultats ont également donné lieu à deux publications, dans une conférence nationale pour jeunes chercheurs sur l'intelligence artificielle (BRENON, PORTET et VACHER 2016a) et dans une conférence internationale sur les environnements intelligents (BRENON, PORTET et VACHER 2016b).

III.2.1 Déroulement

L'expérimentation se décompose en deux phases, représentées sur la Figure III.3, une phase de pré-entrainement et une phase de validation croisée. Pendant la première phase, un modèle est initialisé à partir de données synthétiques générées par un ensemble de règles logiques. Tout au long de cet entrainement, le modèle est évalué sur un corpus similaire (donnant une performance d'apprentissage) après chaque ère d'apprentissage. Ce modèle pré-entrainé est ensuite évalué lors d'une phase de validation croisée. Cette phase utilise la méthode nommée *Leave One Subject Out Cross Validation* (LOSOCV) qui adapte le modèle à partir des données issues de tous les sujets sauf un et qui évalue le modèle sur les données du sujet laissées de côté. Cette adaptation est répétée autant de fois qu'il y a de sujets, en utilisant chaque fois un sujet différent pour l'évaluation. Comme lors du pré-apprentissage, l'évaluation est effectuée après chaque ère d'apprentissage. Les performances de chacune des adaptations sont ensuite moyennées pour obtenir

les performances globales de notre système.

Une telle approche nous a semblé pertinente pour différentes raisons. Tout d'abord, le temps de convergence d'un système d'apprentissage par renforcement est généralement long, et au vu du peu de données réelles disponibles, il nous semblait nécessaire de pré-entrainer le système. Ce pré-entrainement permet au système d'avoir un comportement cohérent dès le début de la phase de validation et donc de se concentrer sur l'adaptation à l'utilisateur plutôt que sur l'apprentissage d'un comportement de base. Ensuite, bien qu'une méthode comme LOSOCV ne semble pas être la plus adéquate pour évaluer la capacité d'un système à s'adapter à un utilisateur, chacun des jeux de données peut être vu comme les données issues d'un seul et même utilisateur. En effet, chacun des sujets ayant suivi le même scénario expérimental, les données enregistrées sont similaires et peuvent donc être considérées comme issues d'un même utilisateur réalisant une même tâche mais de différentes manières.

Certains paramètres sont également modifiés entre les deux phases. Lors de la phase de pré-entrainement, nous ne cherchons pas à simuler l'interaction entre le système et un utilisateur réel. L'utilisation du paramètre $numberOfTries$ n'est donc pas nécessaire, et la valeur du paramètre $triesThreshold$ est fixée à une valeur infinie laissant le système explorer tant qu'il le souhaite jusqu'à trouver la bonne action. Lors de la phase de validation, le comportement est légèrement modifié en autorisant jusqu'à trois essais au système lors de l'adaptation, mais ne lui laissant qu'une seule chance (seulement 1 essai) lors de l'évaluation. L'utilisation de ces paramètres limite par la même occasion la nécessité d'exploration du système lors de la phase de pré-entrainement. En effet, les erreurs répétées influencent les valeurs des états et le système va donc rapidement changer son choix et ce, jusqu'à trouver la bonne action. Lors de la phase de validation, nous ne souhaitons pas laisser le système explorer. En effet, lors de l'exploration, l'agent agit aléatoirement et pourrait causer des désagréments à l'utilisateur. Ainsi, tout au long de cette expérimentation, la valeur de ϵ est fixée à 0, rendant la méthode ϵ-*greedy* inutile en sélectionnant continuellement l'action avec la valeur maximale.

III.2.2 Données

Pour cette expérimentation, nous utilisons un corpus de données librement accessible, le corpus « Interactions » du projet SWEETHOME (VACHER, LECOUTEUX et coll. 2014)[1]. Ce corpus a été enregistré dans l'appartement DOMUS (Figure III.4, GALLISSOT et coll. 2013) déjà évoqué dans la Section II.1.2. Cet appartement de 30 mètres carrés comprend une cuisine, une chambre, une salle de bain et un bureau. Toutes ces pièces sont équipées avec

1. http ://sweet-home-data.imag.fr/

FIG. III.4 : L'appartement intelligent instrumenté DOMUS

différents capteurs et actionneurs tels que des détecteurs de mouvement infrarouges, des capteurs de contacts ou encore des caméras (utilisées uniquement pour l'annotation des données). De plus, 7 microphones sont placés au plafond afin d'enregistrer le flux audio. Au total, près d'une centaine de capteurs permettent de percevoir l'état de l'appartement.

Pour récolter ce corpus, des participants ont été recrutés pour exécuter un scénario de la vie quotidienne dans l'appartement. Il leur était demandé de prononcer des ordres vocaux afin d'activer les différents services de l'habitat. Les différentes situations à jouer par les participants étaient :

1. nettoyer l'appartement ;
2. préparer et prendre un repas ;
3. discuter par visio-conférence ;
4. se détendre (par exemple en lisant) ;
5. faire une sieste.

Pour orienter les participants, la grammaire définissant les commandes vocales valides leur était fournie, en plus d'une description des différentes situations de la vie quotidienne. Le scénario a été pensé pour durer environ 45 minutes, mais aucune limite de temps n'était imposée lors de son exécution. Chaque participant recevait une liste d'actions à réaliser et les commandes vocales à prononcer qui devaient être interprétées par le contrôleur intelligent. Chaque commande devait être répétée au moins trois fois en cas d'erreur du système avant qu'une solution de type magicien d'Oz ne soit utilisée en secours. Au total,

16 personnes (7 femmes et 9 hommes) ont participé à cette expérimentation, permettant de récolter près de 9 heures de données. L'âge moyen des participants était de 38 ans, plus ou moins 13,6 (le plus jeune ayant 19 ans et le plus âgé 62 ans).

Toutefois, la variété de capteurs disponibles génère une grande disparité dans les données récoltées, et il est alors difficile d'obtenir une représentation unifiée permettant de prendre en compte l'ensemble des capteurs. De plus, les approches par renforcement sont peu efficaces lorsque la taille des ensembles d'états ou d'actions augmente trop. En effet, les algorithmes tels que le *Q-Learning* sont assurés de converger vers la stratégie optimale seulement quand le nombre d'interactions tend vers l'infini (Équation III.4). Cette particularité rend l'utilisation de l'apprentissage par renforcement pertinente pour des problèmes où les espaces de recherche sont relativement restreints. Ainsi, il est possible en un minimum d'interactions d'obtenir un comportement similaire à celui obtenu après une infinité d'interactions.

Ces contraintes nous ont poussé à utiliser une représentation de haut niveau des données d'entrée et de sortie. À partir des données issues des capteurs présents dans l'habitat, il est possible de déterminer des caractéristiques de plus haut niveau, telles que la localisation de l'habitant ou son activité courante. Ce travail d'inférence fut notamment réalisé par Chahuara, Fleury et coll. (2016) qui ont été capables de localiser l'utilisateur avec une exactitude statistique de 84 %, de déterminer son activité avec une exactitude de 85 % et de déterminer la commande prononcée avec une exactitude de 100 %. Toutefois, l'utilisation d'un tel système aurait généré du bruit dans les données d'entrée et donc compliqué la tâche de notre système. Nous avons donc fait le choix d'utiliser le résultat d'annotations, assurant une exactitude proche de 100 % pour chacune de ces trois caractéristiques.

Douze commandes différentes ont alors été définies et seulement trois pièces (la cuisine, la chambre et le bureau) étaient utilisées pendant la récolte du corpus. Enfin, 8 activités ont été annotées (Annexe A) plus une activité « inconnue ». Il existe alors un ensemble de $12 \times 3 \times 9 = 324$ états différents. L'ensemble d'actions dépend du nombre d'actionneurs et des commandes qu'ils supportent. Bien que la majorité des actions ne soient jamais réalisées, nous avons choisi de supporter un large ensemble d'actions. Tout actionneur utilisé au moins une fois était ajouté, avec l'ensemble des commandes qu'il supporte, à l'ensemble d'actions possibles, même si certaines commandes n'étaient jamais utilisées. Au total, 33 actions possibles sont supportées et listées Table III.1.

Ces annotations ont ensuite été mises en forme de manière à être facilement traitées par notre système. Ainsi, l'ensemble du corpus a été résumé en un ensemble de tuples de la forme :

```
command      location activity ->  action       device
```

Tab. III.1 : Actions possibles dans l'environnement

Commande	actionneur	Index
Fermer les volets	Volets de la chambre	0
Fermer les volets	Volets de la cuisine	1
Fermer les volets	Volets du bureau	2
Ouvrir les volets	Volets de la chambre	3
Ouvrir les volets	Volets de la cuisine	4
Ouvrir les volets	Volets du bureau	5
Fermer les rideaux	Rideaux de la chambre	6
Ouvrir les rideaux	Rideaux de la chambre	7
Éteindre la lumière	Toutes les lumières de la chambre	8
Éteindre la lumière	Lampes de chevet de la chambre	9
Éteindre la lumière	Plafonnier de la chambre	10
Éteindre la lumière	Toutes les lumières de la cuisine	11
Éteindre la lumière	Plafonnier de la cuisine	12
Éteindre la lumière	Lampe de l'évier de la cuisine	13
Éteindre la lumière	Plafonnier du bureau	14
Allumer la lumière	Toutes les lumières de la chambre	15
Allumer la lumière	Lampes de chevet de la chambre	16
Allumer la lumière	Plafonnier de la chambre	17
Allumer la lumière	Toutes les lumières de la cuisine	18
Allumer la lumière	Plafonnier de la cuisine	19
Allumer la lumière	Lampe de l'évier de la cuisine	20
Allumer la lumière	Plafonnier du bureau	21
Appeler du secours	Téléphone du bureau	22
Appeler la famille	Téléphone du bureau	23
Éteindre la radio	Radio de la chambre	24
Allumer la radio	Radio de la chambre	25
Donner la température	Haut-parleur de la chambre	26
Donner la température	Haut-parleur de la cuisine	27
Donner la température	Haut-parleur du bureau	28
Donner l'heure	Haut-parleur de la chambre	29
Donner l'heure	Haut-parleur de la cuisine	30
Donner l'heure	Haut-parleur du bureau	31

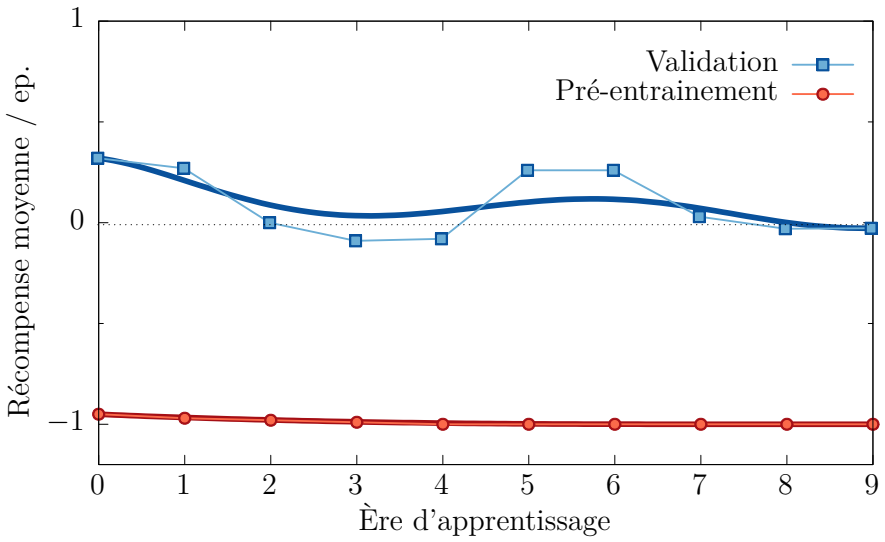


FIG. III.5 : Comparaison de la récompense moyenne lors des 10 séquences d'apprentissage

dont voici quelques exemples :

```
blind - open kitchen  none     ->  blind - open kitchen
light - on   kitchen  cook     ->  light - on   kitchen - sink
```

Le corpus synthétique utilisé lors de la phase de pré-entrainement respecte le même format. Ce corpus est généré en parcourant chacun des 324 états possibles du système et en lui associant une action et un actionneur à partir d'un certain nombre de règles logiques. Le corpus ainsi obtenu comporte 380 échantillons recouvrant la totalité des états possibles. Bien que ne représentant pas strictement la réalité des interactions entre un utilisateur et l'habitat, ce corpus permet d'entraîner un premier modèle de décision qui pourra être amélioré par la suite.

III.2.3 Résultats

La Figure III.5 présente l'évolution de la récompense moyenne obtenue pour chaque interaction lors des deux phases de cette expérimentation. Cette courbe permet de mettre en évidence la différence entre ces phases. Ainsi, la phase de pré-entrainement sert de phase exploratoire durant laquelle l'agent ne tient que peu compte de la récompense et explore les différentes actions disponibles. Ce comportement le pousse à commettre beaucoup d'erreurs ce qui explique la très faible récompense. À l'inverse, lors de la phase d'évaluation le système utilise immédiatement la connaissance qu'il a pour obtenir de meilleures récompenses. On remarque toutefois que la récompense n'atteint pas son maximum qui est de 1. Pour comprendre d'où vient cette sous-optimalité, il peut être intéressant de regarder la matrice de confusion associée à notre tâche de classification d'un contexte vers une décision.

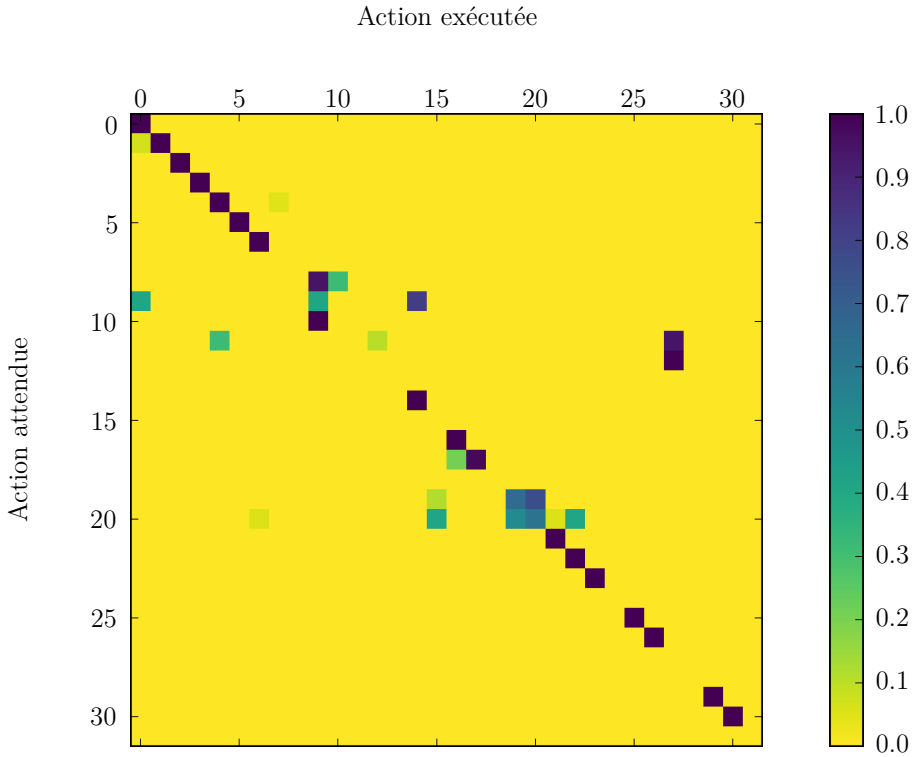


FIG. III.6 : Matrice de confusion pour la tâche de classification. Les index des différentes actions sont disponibles dans la Table III.1.

La matrice de confusion présentée Figure III.6 est une matrice carrée contenant autant de lignes et de colonnes que d'actions possibles. Elle associe l'action attendue (indexant les lignes) à l'action réellement exécutée (indexant les colonnes). Ainsi, chaque cellule $(i\,,j)$ contient le nombre d'interactions où l'action a_i était attendue et l'action a_j a été exécutée. Cette matrice de confusion est normalisée sur les lignes, c'est-à-dire que la somme des valeurs d'une ligne est égale à 1.

La diagonale (correspondant aux cellules où $i = j$) bien marquée de cette matrice est un indice de la bonne performance de notre système. On peut cependant noter quelques confusions qui font baisser les performances globales du système. Une étude plus approfondie montre que ces confusions sont liées à des actions sémantiquement très proches (allumer ou éteindre le plafonnier de la cuisine/la lampe de l'évier de la cuisine) et ne seraient pas nécessairement rebutantes dans le cas d'une utilisation en conditions réelles.

L'utilisation d'une matrice de confusion permet également d'extraire d'autres métriques plus synthétiques comme notamment la valeur de prédictivité positive (précision, *precision*) qui rapporte le nombre d'exemples correctement classés au nombre total d'exemples classés (moyennée pour chacune des classes). Notre système atteint alors une précision de l'ordre de 70 % prouvant une nouvelle fois que l'approche utilisée est applicable à notre tâche.

III.3 Synthèse

Au cours de ce chapitre, nous avons démontré la faisabilité d'un système de prise de décision adaptatif pour l'habitat intelligent basé sur l'apprentissage par renforcement. Le système que nous avons implémenté a été testé sur un ensemble de données issu d'un corpus réaliste enregistré dans un habitat intelligent. L'expérience rapportée, quoique restant assez artificielle, montre bien que les possibilités d'adaptation d'un système à un utilisateur sont possibles même avec une approche par renforcement aussi simple que le *Q-learning* tabulaire.

Cependant une forte restriction a été appliquée à notre système concernant l'ensemble des états possibles. Dans ce travail, la décision est basée sur les informations de lieu et d'activité qui sont le résultat d'une inférence. Cette particularité implique deux points à prendre en compte. D'une part, il faut étudier la possibilité de faire remonter une récompense négative si celle-ci est due à une erreur d'inférence plutôt qu'à une erreur de classification. D'autre part, l'adaptation à l'habitat et aux capteurs repose désormais sur le système d'inférence et non plus sur notre système de décision. Il convient donc d'étudier la capacité de ce système d'inférence à s'adapter.

C'est pourquoi une autre solution est de se passer de cette étape d'inférence, en utilisant directement les informations produites par le réseau domotique. Cette solution fait resurgir deux problèmes déjà évoqués précédemment. D'une part, la définition d'un référentiel commun pour l'ensemble des données hétérogènes. D'autre part, l'utilisation d'un modèle de *Q-Learning* capable de gérer un grand nombre d'états. Dans le Chapitre IV, nous verrons comment il est possible de résoudre ces problèmes et présenterons alors notre nouveau système de prise de décision.

Chapitre IV

Vers une approche profonde

> La seule façon de découvrir les limites du possible,
> c'est de s'aventurer un peu au-delà, dans l'impossible.
>
> Arthur C. Clarke

Nous venons de le voir, l'apprentissage par renforcement est une méthode prometteuse pour le développement d'un habitat intelligent adaptatif. Toutefois, le système présenté dans le chapitre précédent dépend des performances d'un système d'inférence pour restreindre l'espace de recherche. En supprimant cette phase d'inférence, il serait possible de développer un seul et unique système. Ceci limiterait de fait la complexité globale et permettrait de tirer profit de l'apprentissage par renforcement pour permettre au système de s'adapter à l'habitat et aux capteurs. Supprimer cette phase d'inférence fait cependant surgir deux principaux problèmes :

- les données issues du réseau domotique étant très diverses, il est nécessaire de développer une représentation universelle dans laquelle projeter l'ensemble des données ;
- dans le cas de l'utilisation directe des données issues des capteurs, l'espace des états possibles (combinaisons possibles des valeurs des capteurs) devient directement lié au nombre de capteurs, et pour un habitat tel que Domus, le nombre d'états atteint plusieurs milliers.

Ce dernier point est un problème récurrent soulevé par les approches par renforcement, et mobilise donc de nombreux chercheurs. Une récente avancée (Mnih et coll. 2015) vient ainsi proposer une solution à ce défi grâce aux capacités des réseaux de neurones profonds.

Dans la Section IV.1 nous commencerons par une introduction aux modèles neuronaux afin de mieux appréhender la description de la solution présentée ensuite. La Section IV.2 détaillera ensuite le système ARCADES (***A**daptive **R**einforced **C**ontext-**A**ware **De**ep **De**cision **S**ystem*) que nous avons mis au point, reposant sur l'apprentissage profond par renforcement.

IV.1 L'apprentissage profond par renforcement

L'apprentissage profond par renforcement (*Deep Reinforcement Learning*) est une méthode d'apprentissage tirant parti à la fois des spécificités de l'apprentissage par renforcement (apprentissage incrémentiel, utilisation d'un signal de récompense) et de l'apprentissage neuronal profond. Dans cette section, nous commencerons par introduire les modèles neuronaux et leur fonctionnement de manière à mieux comprendre les avantages qu'ils peuvent apporter. Nous présenterons ensuite une récente implémentation de l'apprentissage profond par renforcement qui a démontré la faisabilité d'une telle approche.

IV.1.1 Les modèles neuronaux

Les ***réseaux de neurones*** (*Neural Nets* – NN) sont des modèles fondés sur de petites unités de calculs, les neurones. Un neurone (Figure IV.1) est capable de modéliser un problème linéaire en appliquant une fonction d'activation prédéfinie f, au résultat du produit vectoriel entre un vecteur d'entrée $\mathbf{x}$ et un vecteur interne de poids $\mathbf{w}$ (Annexe B, MCCULLOCH et PITTS 1943). Cette définition du neurone est une tentative de MCCULLOCH et PITTS (respectivement neurobiologiste et mathématicien) de représenter le fonctionnement interne des neurones biologiques, qu'ils expérimentèrent dans leur système MCP.

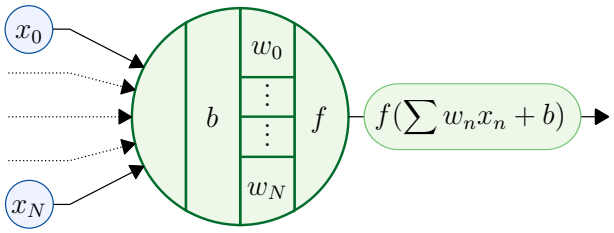


FIG. IV.1 : Fonctionnement d'un neurone dans les systèmes neuronaux artificiels

Toutefois, leur système laisse de côté une particularité importante des neurones biologiques, leur capacité à apprendre. Il faudra attendre les travaux de ROSENBLATT (1958), qui mettra au point le *Perceptron*, pour incorporer cette capacité. Le Perceptron est un système n'utilisant qu'un seul neurone, mais un neurone capable d'apprendre ses poids à partir d'exemples, c'est-à-dire par apprentissage supervisé.

Par la suite, Minsky et Papert (1969) pointeront du doigt les limitations des perceptrons et notamment leur capacité à ne représenter que des fonctions linéaires. Mais leur publication ne considère pas de modèle composé de plusieurs perceptrons. En effet, en plaçant plusieurs perceptrons côte à côte, on obtient un réseau de neurones à une couche. Puis, en cascadant et reliant plusieurs de ces couches il est possible d'obtenir un réseau de neurones à plusieurs couches, ou un perceptron multi-couches (*Multi-Layer Perceptron* – MLP).

Presque vingt ans après Minsky et Papert, Cybenko (1989) apportera la preuve des capacités exceptionnelles que peuvent fournir ces perceptrons multi-couches. Il énoncera ainsi la *propriété d'approximation universelle* (Wang et Raj 2017).

- Un MLP de deux couches peut représenter de manière exacte toute fonction booléenne.
- Un MLP de deux couches peut représenter avec une exactitude arbitraire toute fonction continue.
- Un MLP de trois couches peut représenter avec une exactitude arbitraire toute fonction arbitraire.

Cette propriété théorise les capacités des réseaux de neurones, mais au prix de couches contenant un nombre de neurones exponentiel. La possibilité de conserver la puissance de représentation des réseaux neuronaux tout en limitant le nombre de neurones nécessaires se pose alors. Les recherches à ce sujet vont donc étudier l'impact de la profondeur d'un réseau et mener à l'émergence de ***l'apprentissage profond***.

IV.1.1.1 Apprentissage par rétropropagation

Dès les années 1980, Yao (1985) et Håstad (1986) vont démontrer que si des MLP avec peu de couches nécessitent un trop grand nombre de neurones, un MLP avec plus de couches pourra représenter les mêmes fonctions mais avec un nombre de neurones plus faible. Formellement, une fonction pouvant être calculée avec un nombre polynomial n de neurones répartis sur k couches nécessite un nombre exponentiel e^n de neurones si le réseau se limite à $k-1$ couches. Plus récemment, Eldan et Shamir (2015) ont démontré que l'accroissement de la profondeur d'un réseau est bien plus pertinent que l'accroissement de sa largeur (le nombre maximal de neurones dans une couche). Les tentatives pour adopter des modèles profonds (composés d'au moins trois couches, une couche d'entrée, une couche cachée et une couche de sortie) se sont alors multipliées. Toutefois, l'augmentation du nombre de couches pose un problème pour l'apprentissage des paramètres de ces neurones.

En effet, la technique la plus utilisée pour déterminer les paramètres optimaux des neurones repose sur ***l'algorithme de descente du gradient stochastique*** (*Stochastic Gradient Descent* – SGD, LeCun, Y. Bengio et G. Hinton 2015) permettant d'optimiser les poids du réseau afin de minimiser la fonction de coût (utilisée pour mesurer la performance du réseau). Une telle approche nécessite de calculer le gradient à la sortie du réseau et de le rétropropager à travers celui-ci afin de modifier les poids proportionnellement à l'erreur obtenue.

Bien que des techniques de rétropropagation soient déjà connues, celles-ci sont coûteuses en temps et en puissance de calcul. En 1962, Dreyfus (1962) détailla une méthode reposant sur le théorème de dérivation des fonctions composées permettant de réduire considérablement la complexité du calcul et de la rétropropagation du gradient. Cette technique de *back-propagation (backprop)* a par la suite été appliquée avec succès (Werbos 1982 ; LeCun 1985) avant d'être popularisée par Rumelhart, McClelland et The P. D. P. Research Group (1986) comme méthode de référence pour l'apprentissage des réseaux de neurones.

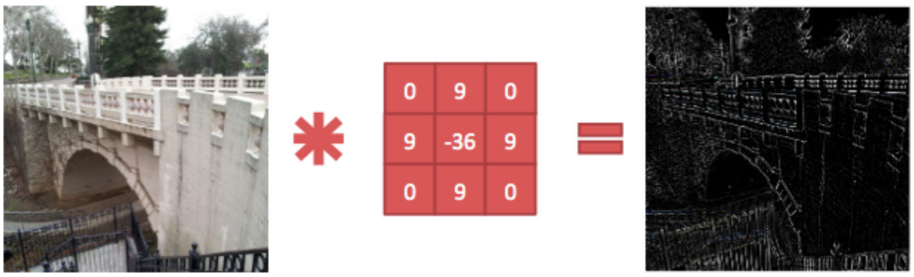
Les architectures des réseaux ont alors pu se complexifier, le nombre de couches internes se multipliant. Le terme d'**apprentissage profond** s'est alors imposé pour mettre en exergue l'importance théorique de la profondeur d'un réseau de neurones, et pour marquer le fait qu'il est désormais possible d'entraîner des réseaux plus profonds que ce qu'il n'a jamais été fait jusqu'à présent.

Pourtant, dès le début des années 1980, Fukushima (1980) avait déjà proposé une architecture composée de plusieurs couches et qui introduisait ce que l'on connaît aujourd'hui comme les réseaux convolutifs.

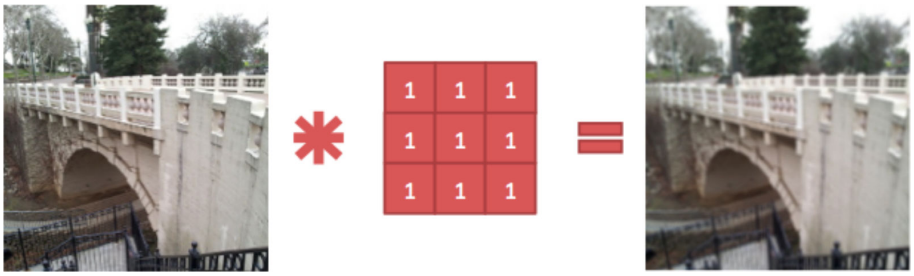
IV.1.1.2 Réseaux convolutifs et auto-encodeurs

Les découvertes dans le domaine de la neurobiologie ont démontré à la fin des années 1950 que différentes parties du cerveau entrent en jeu pour la reconnaissance de formes (Wiesel et Hubel 1959). Après avoir été perçus par la rétine, les signaux lumineux sont traités par le cortex visuel primaire puis secondaire. Ces cortex vont principalement détecter des caractéristiques simples comme les bords présents dans l'image perçue, les couleurs, l'orientation, etc. Ces caractéristiques sont par la suite traitées pour en extraire des formes qui seront ensuite utilisées par le lobe temporal pour identifier les objets présents dans la scène.

Ces découvertes vont inspirer Fukushima (1980) qui va développer le *Neocognitron*. Ce système neuronal à plusieurs couches introduit notamment des couches convolutives et de sous-échantillonnage qui imitent le fonctionnement des cortex visuels. Les couches de



(a) Convolution pour la détection de bords



(b) Convolution pour flouter une image

FIG. IV.2 : Exemples de résultats de convolutions en fonction de la matrice de convolution (WANG et RAJ 2017)

convolutions appliquent une simple opération mathématique entre un signal d'entrée et une matrice de convolution, ce qui permet de modifier les données initiales (Figure IV.2, Annexe B). Il est ainsi possible d'imiter le fonctionnement des cortex visuels, en utilisant des matrices de convolutions adéquates, comme notamment la détection de contours (Figure IV.2a). Les couches de sous-échantillonnage permettent, pour leur part, d'assurer une certaine résistance au bruit.

Par la suite, les réseaux convolutifs (*Convolutionnal Neural Network* – CNN) ont été démocratisés par LECUN, BOTTOU et coll. (1998) qui les ont utilisés avec succès pour la reconnaissance de caractères. Le défi *ImageNet*, dont l'objectif est de classer un ensemble d'images, a alors vu émerger les approches utilisant les réseaux convolutifs. Le premier grand succès des CNN lors de ce concours date de 2012 (KRIZHEVSKY, SUTSKEVER et G. E. HINTON 2012). En utilisant la puissance des processeurs graphiques (*Graphics Processing Unit* – GPU), KRIZHEVSKY et coll. ouvraient la voie à de nombreux autres systèmes (VGG, ResNet, etc.) (WANG et RAJ 2017 ; KARPATHY 2015).

L'utilisation de plusieurs couches de convolutions permet d'obtenir une représentation abstraite des données d'entrée. La Figure IV.3 montre comment à partir de données brutes, une matrice de pixels, les différentes couches du réseau permettent de découvrir des motifs de plus en plus complexes et abstraits.

Cette spécificité n'est pas intrinsèquement liée aux CNN, mais aux réseaux de neurones profonds plus généralement. Elle est particulièrement mise en évidence dans le cadre des

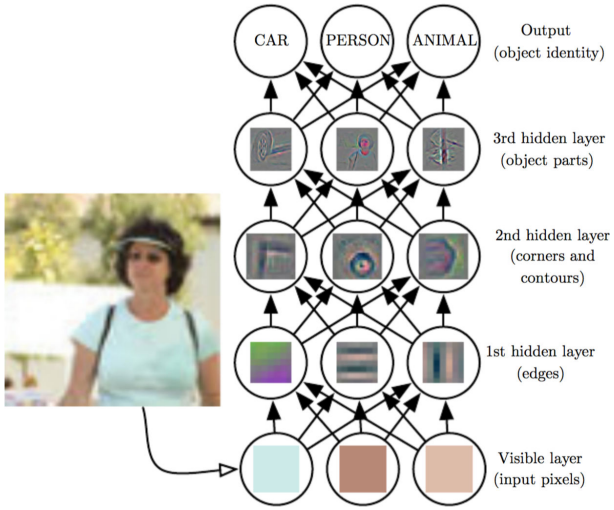


FIG. IV.3 : Exemple de la décomposition d'une image par un réseau de neurones à 5 couches (ZEILER et FERGUS 2014)

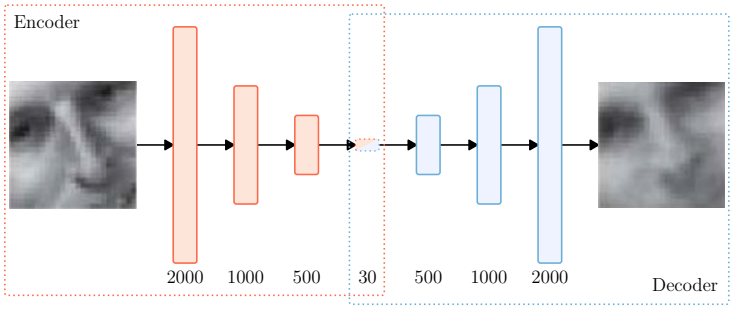


FIG. IV.4 : Exemple d'auto-encodeur permettant de représenter une image de taille 25 × 25 en un vecteur de 30 caractéristiques (G. E. HINTON et SALAKHUTDINOV 2006)

auto-encodeurs (Y. BENGIO 2009) et notamment des auto-encodeurs creux (*sparse autoencoders*). Ces réseaux de neurones profonds respectent une architecture symétrique dans laquelle la couche $n+1$ est moins large que la couche n (Figure IV.4). Ils sont ensuite entraînés en utilisant les mêmes données en entrée et en sortie. Cette phase d'apprentissage non-supervisé[1] permet alors de déterminer une fonction de compression des données similaire aux approches par analyse en composantes principales (G. E. HINTON et SALAKHUTDINOV 2006).

Il s'agit ici d'une des capacités principales des réseaux de neurones. En utilisant une architecture composée de plusieurs couches, le réseau est capable de déterminer par lui-même les caractéristiques pertinentes à partir de données de grande dimension. Cette spécificité permet donc de limiter l'impact de la sélection d'attributs généralement réalisée par un expert du domaine en laissant le réseau construire les caractéristiques discriminantes.

1. Il s'agit en fait d'un apprentissage supervisé mais dont les étiquettes sont directement obtenues à partir des données d'entrée et qui ne nécessite dont pas d'étiquetage de la part d'un expert.

IV.1.2 Deep Q-Network – DQN

L'apprentissage profond par renforcement cherche donc à utiliser un réseau de neurones profond de manière à prendre en charge un espace d'entrée de grande taille. De plus, nous l'avons vu, un réseau de neurones profond est également capable d'approximer toute fonction arbitraire (Cybenko 1989) et peut donc être utilisé pour approximer la fonction de Q-valeur d'un MDP, en lieu et place d'un tableau associatif. Toutefois, les premières tentatives reposant sur cette approche rencontraient des problèmes, le réseau ne convergeant pas vers la fonction de Q-valeur optimale (Tsitsiklis et Roy 1997).

Une des premières implémentations fructueuses de cette méthode est le *Deep Q-Network* introduit par Mnih et coll. (2015). Ce système a été testé sur une tâche de décision concernant des jeux vidéos de la console Atari2600 et obtient des performances comparables à un joueur humain sur 49 jeux. Schématiquement, à partir d'une suite d'images de l'écran de la console Atari (les 4 dernières images), le système choisit une action parmi 18 pour maximiser sa récompense, calculée à partir de l'évolution du score. Pour réaliser cela, les images de l'écran sont traitées par un réseau de neurones convolutif. Le vecteur résultant passe alors au travers d'un réseau de neurones totalement connectés qui fournit les valeurs de chacun des Q-états possibles à partir de l'état d'entrée. Une stratégie de type *ϵ-greedy* sélectionne ensuite l'action à exécuter. Après plusieurs millions d'interactions, le système est alors à même de concourir contre un humain et même de le surpasser dans la plupart des jeux.

Dans ces travaux, les 4 images fournies en entrée du réseau sont monochromes (256 niveaux de gris) et ont une taille de $84\,\text{px} \times 84\,\text{px}$. L'ensemble des entrées possibles à donc une taille de $256^{4\times84\times84} = 2^{225\,792} \approx 1{,}5 \cdot 10^{67\,970}$, bien au-delà des 324 états de notre système. Pour corriger le problème de convergence rencontré par la majorité des méthodes de renforcement profond, Mnih et coll. (2015) ont dû utiliser différentes méthodes que nous allons détailler.

Pour entrainer un réseau neuronal, la majorité des approches utilisent une fonction de coût permettant de calculer une perte entre la valeur retournée par le réseau et la vérité terrain. Dans le cas de l'apprentissage profond par renforcement cette vérité terrain n'existe pas. Nous avons d'une part des Q-valeurs fournies par le réseau et d'autre part une récompense évaluant l'action réalisée. La solution consiste alors à calculer une nouvelle Q-valeur, plus juste, pour l'action exécutée. On utilise pour cela l'Équation III.4 rappelée ci-dessous.

$$q = R(s') + \gamma max_{a'} Q_k(s', a') \quad \text{d'où} \quad \begin{aligned} Q_{k+1}(s,a) &\leftarrow (1-\alpha)Q_k(s,a) + \alpha q \\ &= Q_k(s,a) + \alpha(q - Q_k(s,a)) \end{aligned}$$

Il est alors possible de déterminer le coût de l'erreur réalisée en faisant la différence entre l'ancienne ($Q_k(s\,,a)$) et la nouvelle ($Q_{k+1}(s\,,a)$) Q-valeur. Cette solution, calquée sur le

fonctionnement du *Q-learning* tabulaire, n'est cependant pas satisfaisante, amenant généralement le réseau à osciller entre différentes fonctions de Q-valeur, sans jamais converger vers la fonction optimale. Pour palier ce problème, Mnih et coll. (2015) introduisent la notion de réseau cible, appelé *target-Q*. Ce réseau est en fait un clone du réseau *Q-Network*, qui permet d'approximer la fonction de Q-valeur, mais qui est mis à jour moins régulièrement (après un nombre `target_q` d'interactions). Ainsi, l'équation de mise à jour peut être reformulée :

$$q = R(s') + \gamma max_{a'} Q_{k-x}(s', a') \quad \text{d'où} \quad \begin{aligned} Q_{k+1}(s,a) &\leftarrow (1-\alpha)Q_k(s,a) + \alpha q \\ &= Q_k(s,a) + \alpha(q - Q_k(s,a)) \end{aligned}$$

Où $x \in [0\,;\texttt{target\_q}]$. Par ce biais, Mnih et coll. (2015) retardent l'impact des modifications de la fonction de Q-valeur sur les Q-valeurs cibles.

Une autre méthode permettant d'améliorer la convergence du système est l'utilisation de la technique de *réutilisation de l'expérience* (*experience replay*, Lin 1993). Cette technique utilise une mémoire dans laquelle chaque expérience de l'agent $\langle s\,, a\,, r\,, s' \rangle$ est sauvegardée. Au moment de l'apprentissage, les nouvelles Q-valeurs sont calculées à partir d'expériences tirées aléatoirement dans cette mémoire. En limitant l'impact des récentes expériences (puisque l'expérience utilisée n'est pas nécessairement une des dernières), cette technique permet d'éviter au réseau de stagner dans des minimas locaux. En outre, elle apporte également d'autres avantages. En réutilisant potentiellement plusieurs fois une même expérience, cette technique permet de limiter la quantité de données nécessaire. Enfin, l'apprentissage à partir d'exemples consécutifs peut créer un biais causé par les corrélations entre ces exemples. En tirant aléatoirement les exemples ceux-ci sont décorrélés, ce qui améliore la performance globale de l'apprentissage.

Ces techniques permettent alors au réseau de converger plus certainement vers la fonction de Q-valeur optimale en dépit d'un temps de convergence plus long. Pour contre-balancer cet allongement du temps de convergence, d'autres méthodes, utilisées pour les réseaux de neurones en général et pas spécifiquement pour l'apprentissage profond par renforcement, sont appliquées. Notamment, la méthode d'apprentissage repose sur une propagation du gradient par mini-lots. L'apprentissage par *mini-lots* (*mini-batch*) permet de limiter les divergences de gradient entre deux étapes d'apprentissage. En effet, la méthode de rétropropagation présentée Section IV.1.1.1 était initialement appliquée pour chaque échantillon d'apprentissage, mais pouvait alors rencontrer des échantillons contradictoires générant des gradients opposés. Lors d'une approche par *mini-lots*, un ensemble $\mathcal{B}$ d'échantillons est traité par le réseau, et on calcule alors le gradient sur l'ensemble de ces échantillons, lissant ainsi les disparités inter-échantillons afin de représenter plus justement l'ensemble du corpus.

De la même manière, une méthode d'optimisation de la descente du gradient est appliquée. Il est fait ici usage d'une méthode non-publiée mais largement évoquée et éprouvée par la communauté (Golge 2015 ; Raffel 2016 ; Ruder 2016) appelée RMSProp (G. E. Hinton 2015). Ainsi, le gradient ∇ calculé à partir d'un mini-lot $\mathcal{B}$ est pondéré par ses variations passées selon un hyper-paramètre α, limitant ainsi ses fluctuations. De plus, les méthodes traditionnelles nécessitent la sélection d'un hyper-paramètre définissant la vitesse d'annihilation du taux d'apprentissage η. Un choix mal avisé peut alors réduire la qualité de l'apprentissage. Avec RMSProp, la moyenne courante du carré des gradients ($MeanSquare$), qui est croissante, est utilisée pour pondérer le taux d'apprentissage η. Il n'est alors plus nécessaire de réaliser de coûteux tests pour déterminer le paramètre optimal. Les Équations IV.1 et IV.2 mettent en parallèle la méthode classique de mise à jour des paramètres w d'un réseau de neurones et la méthode RMSProp.

$$\text{SGD classique :} \quad w = w - \eta \cdot \nabla \tag{IV.1}$$

$$\begin{aligned} \text{RMSProp :} \quad & w = w - \frac{\eta}{\sqrt{MeanSquare_t}} \cdot \nabla \\ \text{où} \quad & MeanSquare_t = \alpha MeanSquare_{t-1} + (1-\alpha)\nabla^2 \\ & 0 < \alpha < 1 \end{aligned} \tag{IV.2}$$

C'est en utilisant ces méthodes conjointement que le DQN a réussi à atteindre de telle performances. Mnih et coll. (2015) ont ainsi proposé une solution fonctionnelle tirant profit à la fois de l'apprentissage par renforcement et de l'apprentissage profond (gestion d'un grand nombre d'entrées, extraction des caractéristiques pertinentes). Cette réussite nous à alors servi de base pour la réalisation du système Arcades.

IV.2 Le système Arcades

Le système Arcades est une implémentation de l'apprentissage profond par renforcement appliquée à la tâche de prise de décision dans un habitat intelligent. En ce sens, il est une évolution du système présenté Chapitre III, reprenant les mécanismes du *Deep Q-Network*. Il est la principale contribution de cette thèse et son code source est librement accessible sur Internet[2].

Le système de prise de décision développé par Mnih et coll. (2015) présente une solution au problème énoncé au début de ce chapitre concernant la taille de l'ensemble d'entrées d'un système de renforcement. Il ne propose en revanche aucune piste concernant la

2. https ://brenona.gricad-pages.univ-grenoble-alpes.fr/arcades/

représentation de données multimodales dans un espace de référence. Dans la première partie de cette section nous détaillerons le choix qui a été fait de projeter l'ensemble de ces données sous la forme d'une image, avant de décrire plus en détails l'architecture d'Arcades et son fonctionnement.

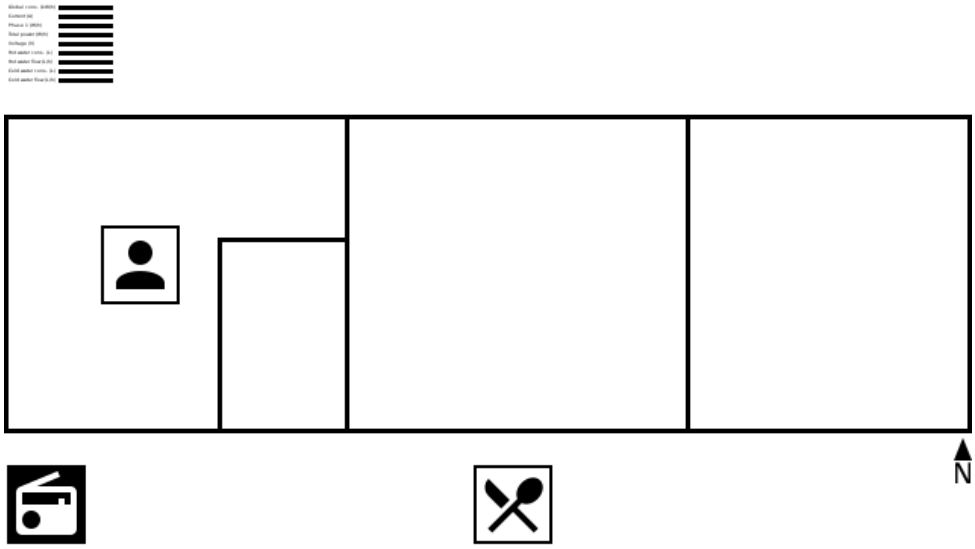
IV.2.1 Une représentation graphique des données

Un des défis du traitement de données est la définition d'un espace de référence dans lequel toutes les données peuvent être représentées et manipulées. Dans le cadre des habitats intelligents, les données sont multimodales : des séries temporelles continues (comme la consommation d'eau ou la température) jusqu'aux informations sémantiques événementielles (comme une commande vocale ou le changement d'état d'un capteur). Il est alors problématique de trouver une représentation s'accommodant à la fois de données bas-niveau issues de capteurs (mouvement dans une pièce) et d'événements haut-niveau (l'activité actuelle de l'utilisateur). Aucun consensus ne semble aujourd'hui avoir été trouvé.

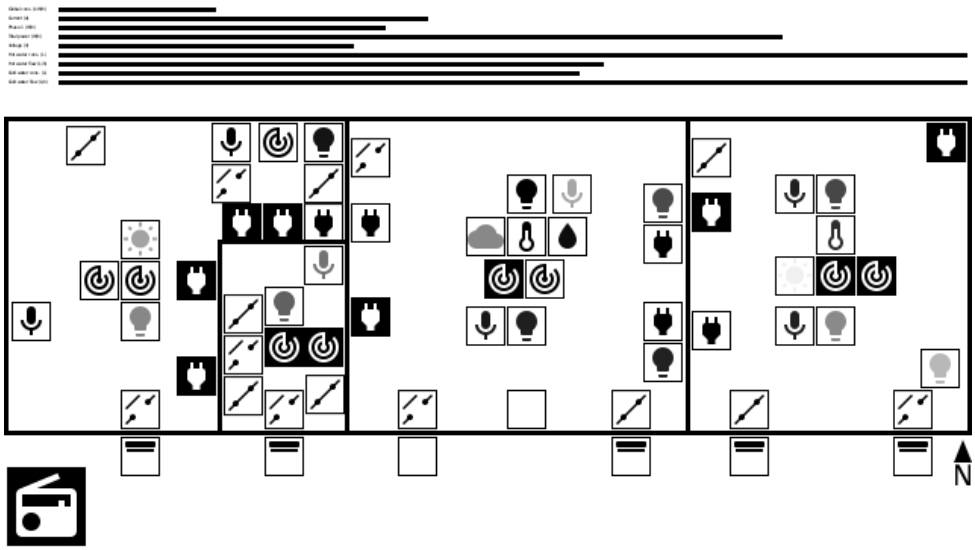
Au vu des performances des réseaux de neurones, et notamment des réseaux convolutifs, concernant le traitement d'image, nous avons décidé de représenter l'ensemble de nos données à travers une image. Nous avons alors choisi d'utiliser une **représentation graphique** en projetant à tout instant l'ensemble des informations sur une carte bidimensionnelle de l'habitat. Les capacités des réseaux convolutifs permettront alors de construire les caractéristiques pertinentes de l'environnement, un *contexte*, à partir de cette représentation brute.

L'utilisation d'une telle approche apporte de nombreux avantages.

- Une représentation graphique permet de prendre en charge de nombreuses données de différents types et peut donc aisément prendre en charge de nouveaux capteurs.
- La taille de l'image résultante n'est pas directement liée au nombre de capteurs ou à leur type. Ainsi, l'ajout, la suppression ou la modification de capteurs n'impactera pas le format d'entrée de l'agent.
- Les informations spatiales sont intrinsèquement incluses dans une image.
- L'apprentissage profond peut utiliser de manière efficace la forte redondance des données (une valeur d'un capteur étant représentée par un ensemble de pixels et différents capteurs pouvant fournir une même information) ;
- Un humain peut voir et comprendre l'image sans nécessiter de connaissances particulières.



(a) Exemple d'image générée à partir des annotations



(b) Exemple d'image générée à partir des données brutes issues des capteurs

FIG. IV.5 : Exemples d'images générées pour la représentation de l'état de l'environnement

La Figure IV.5 présente deux exemples d'images générées pour l'habitat Domus considéré dans nos recherches. Dans les deux cas, les murs et cloisons de l'habitat sont représentés par d'épaisses lignes noires. La Figure IV.5a représente des informations contextuelles issues du processus d'annotation. On y retrouve ainsi l'activité de l'utilisateur, la commande vocale prononcée et la localisation de l'utilisateur au sein de l'appartement. Ici, il est possible d'en déduire que l'utilisateur demande à éteindre la radio pendant qu'il mange dans la cuisine. Cette même information peut également être présentée non plus à l'aide des données inférées de haut-niveau, mais à partir des données brutes des capteurs. L'image résultante est présentée Figure IV.5b où chaque icône représente un capteur. Par exemple, l'utilisation d'une lampe est représentée par une ampoule noire sur fond blanc, à l'inverse d'une ampoule blanche sur fond noir représentant une lampe éteinte. La partie supérieure de l'image contient des jauges rendant compte de la consommation d'électricité et d'eau. Enfin, l'imposante icône en bas à gauche est le résultat du système de reconnaissance vocale. Chaque icône correspond alors au dernier état connu pour le capteur associé. À partir de cette image, on peut en déduire que l'utilisateur se trouve dans la cuisine et plus précisément proche de la chambre (les détecteurs de mouvement sont activés) dont la porte est ouverte. L'environnement est globalement bruyant comme le montre la forte activation des microphones. Tous les volets sont ouverts sauf un dont l'état n'est pas connu. Il existe une icône dédiée pour les différents types de capteurs, tels que :

- les capteurs binaires : détecteurs de présence, contacts de portes et fenêtres, état des volets, état des lampes, etc. ;
- les capteurs multinomiaux : activité courante, commande vocale ;
- les capteurs continus : niveau des lampes, niveau de bruit, température, humidité ;
- les jauges : consommation électrique, consommation d'eau, etc.

Une liste exhaustive des icônes et de leurs variantes est présentée dans l'Annexe A.

Plusieurs facteurs ont motivé nos choix quant à la représentation de l'appartement et des capteurs. Ainsi, l'image et les icônes ont été pensées de manière à maximiser les contrastes et la redondance, en utilisant un ensemble limité de composantes de base (lignes orthogonales, cercles), pour faciliter l'apprentissage du réseau convolutif. De plus, la volonté des utilisateurs de comprendre les choix du système, démontrée par Zaidenberg, Reignier et Mandran (2010), nous a poussé à réaliser des images compréhensibles par un humain. Enfin, la nécessité de générer de telles images continuellement, pour rendre compte de l'évolution de l'environnement, nous a mené à une implémentation reposant sur le format SVG (*Scalable Vector Graphics*). Ce format ouvert très largement répandu (notamment

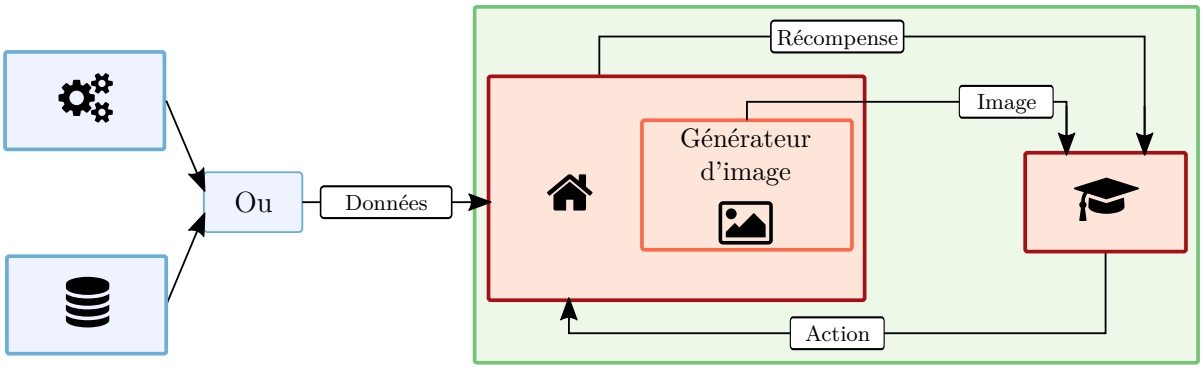


FIG. IV.6 : Architecture du système ARCADES

sur Internet) a été développé et est maintenu par le *World Wide Web Consortium* (W3C)[3]. Issu de la norme XML (*eXtended Markup Language*), il permet de générer ou de modifier aisément une image, et ce, de manière automatique.

L'utilisation d'une image permet donc de répondre à de nombreuses problématiques (représentation commune, adaptation à la topologie ou aux capteurs, etc.). Notre précédent système, présenté Chapitre III, a donc été modifié afin d'incorporer ces adaptations.

IV.2.2 Fonctionnement

La Figure IV.6 reprend l'architecture du système de décision dans l'habitat intelligent présentée Section III.1.2 en faisant apparaitre en rouge les composants que nous avons fait évoluer (l'environnement et l'agent).

Les modifications liées à l'environnement sont relativement limitées et se concentrent sur la génération de l'image. Ainsi, chaque seconde (ou lorsqu'un ordre vocal valide est reconnu, selon le mode de fonctionnement), l'interface récupère l'ensemble des valeurs des capteurs à l'instant t. Ces valeurs sont ensuite projetées sur l'image (les autres capteurs restant dans leur dernier état connu). L'image est alors passée à l'agent qui se chargera de réaliser des étapes de pré-traitement si nécessaire.

L'agent, en revanche a subi de nombreuses modifications. Dans un premier temps, il a fallu remplacer la table d'association par un réseau de neurones. La première architecture du réseau (Architecture IV.A1[4]) a été calquée sur celle donnée par MNIH et coll. (2015). Lors de l'utilisation de cette architecture, les images d'entrée était redimensionnées, d'une résolution de 639 px × 360 px à 84 px × 84 px. Ce pré-traitement était acceptable sur des images contenant peu d'informations (images générées à partir d'informations inférées),

3. https ://www.w3.org/Graphics/SVG/
4. $C_n[f+p/s]$ représente une couche convolutive de n filtres de taille $f \times f$, avec un bourrage de p pixels et un intervalle de s pixels. FC_n représente une couche de neurones totalement connectés de largeur n. Plus de détails dans l'Annexe B.

mais n'était pas applicable lorsque la totalité des capteurs était représentée. En effet, la perte d'information était telle, que l'image résultante était à la fois illisible par l'humain et inutilisable par le système.

Nous avons donc fait le choix de redimensionner les images vers une résolution de 256 px × 256 px. Il s'agit d'une définition communément utilisée pour la reconnaissance d'images par les réseaux de neurones et l'image résultante reste exploitable par l'humain et le système. Cette augmentation de la résolution induit indubitablement une augmentation de la taille de l'ensemble des entrées possibles, et il a donc fallu adapter le réseau. Pour prendre en charge cette nouvelle quantité de données, une couche de convolution a été ajoutée au réseau, résultant dans l'Architecture IV.A2.

$$C_{32}[8+0/4] \to C_{64}[4+0/2] \to C_{64}[3+0/1] \to FC_{512} \to FC_{18} \tag{IV.A1}$$

$$C_{16}[8+2/4] \to C_{32}[6+1/3] \to C_{64}[3+0/2] \to C_{64}[3+0/1] \to FC_{512} \to FC_{33} \tag{IV.A2}$$

Cette nouvelle architecture, présentée Figure IV.7, utilise alors quatre couches de convolutions, permettant de réduire l'image de base en des cartes d'activations de tailles : 64×64, 21×21, 10×10 puis 8×8 pixels. La dernière couche utilisant 64 filtres, les cartes d'activations sont ensuite réarrangées en un vecteur de dimension $64 \times 8 \times 8 = 4096$. Ce vecteur passe ensuite au travers de deux couches de neurones totalement connectés (où un neurone d'une couche est connecté à l'ensemble des neurones de la couche précédente) de taille 512 et 33, correspondant au nombre d'actions possibles. Assez classiquement, les différentes couches du réseau sont séparées par une couche de non-linéarité de type `ReLU` qui, pour toute entrée x, retourne 0 si x est négatif, x sinon.

Ainsi, à partir d'une image donnée, représentant l'état de l'environnement s, ce réseau de neurones génère un vecteur. Ce vecteur contient 33 valeurs réelles non bornées, représentant les Q-valeurs de chacune des 33 actions possibles. Bien que la majorité des réseaux de neurones aient recours à des couches non-paramétriques en sortie pour normaliser les valeurs, de type `SoftMax` – permettant de borner les valeurs entre 0 et 1 – ou `Log` – permettant de limiter la perte de précision pour les valeurs proche de 0 –, ces couches sont absentes de cette architecture (et l'étaient de celle de Mnih et coll. (2015)). En outre, leur utilisation affecte de manière négative les performances du système.

Une fois le réseau défini, nous avons adapté les différents algorithmes qui utilisaient jusqu'alors la table d'association (Algorithmes III.2 et III.3).

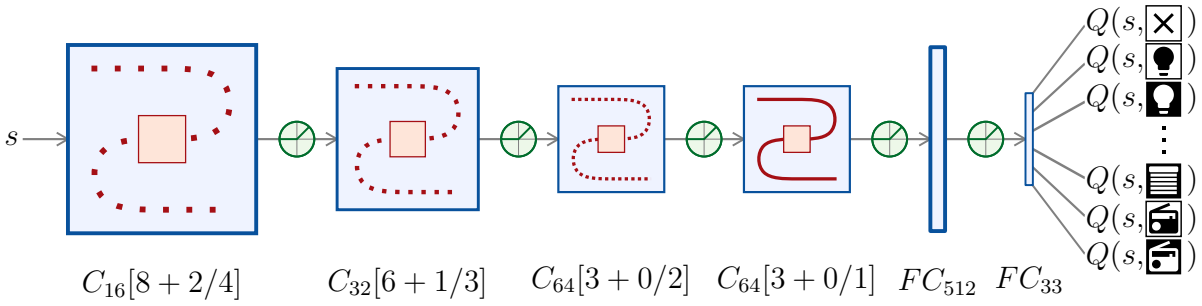


FIG. IV.7 : Architecture de notre réseau de neurones

La sélection d'actions (Algorithme IV.1) suit la même logique que précédemment, mais l'état n'est pas utilisé comme un index de tableau mais comme l'entrée du réseau. Ce réseau fournit en sortie un ensemble de valeurs correspondant à l'ensemble des Q-valeurs étant donnée l'entrée. Une nouvelle fois, une stratégie de type *ϵ-greedy* permet de sélectionner l'action avec la Q-valeur la plus élevée ou une action aléatoire.

Algorithme IV.1 Stratégie de décision de l'agent

Require: $\epsilon \in [0\,;1]$
  function GETACTION
    $x \sim \mathcal{U}([0\,,1])$
    if $x \leq \epsilon$ **then**
      action $\sim \mathcal{U}(\mathcal{A})$ ▷ Action aléatoire
    else
      Q-values $\leftarrow$ Q-**Network**.FORWARD(state)
      action $\leftarrow$ argmax(**Q**-values)
    end if
    return action
  end function

La fonction d'apprentissage (Algorithme IV.2) est en revanche plus complexe. Elle doit introduire les différents mécanismes du DQN, présentés Section IV.1.2. Ainsi, chaque interaction $\langle s_t\,, a_t\,, r_t\,, s_{t+1}\rangle$ expérimentée par l'agent est enregistrée dans une mémoire $\mathcal{I}$. Au moment de l'apprentissage, un lot $\mathcal{B}$ d'interactions est tiré aléatoirement dans la mémoire $\mathcal{I}$ et utilisé pour entrainer le réseau. La récompense obtenue étant connue pour chacun des échantillons de $\mathcal{B}$, il est possible de calculer une Q-valeur cible pour chacun des états s_t du lot grâce à l'Équation IV.3.

$$\text{target-}Q = r_t + \gamma \max_a Q(s_{t+1}, a) \tag{IV.3}$$

C'est lors de cette étape qu'intervient le réseau cible *target-Q* évoqué plus tôt et qui permet de retarder la propagation des récompenses. Cette particularité permet d'améliorer la stabilité de l'algorithme comparée aux approches standards. En effet, la modification d'une valeur $Q(s_t\,, a)$ modifie généralement la valeur $Q(s_{t+1}\,, a)$ pour toutes les actions a et modifie donc la valeur cible, pouvant provoquer des oscillations voire des divergences.

L'utilisation d'une ancienne version du réseau permet de limiter ces risques en retardant l'impact d'une récompense sur le calcul des valeurs cibles Mnih et coll. (2015). Finalement, la différence entre ces valeurs cibles et les valeurs actuelles des états est utilisée comme perte et permet de calculer le gradient. Le gradient peut alors être optimisé par l'algorithme RMSProp, avant d'être retropropagé dans le réseau pour modifier ses paramètres.

Algorithme IV.2 Méthode d'apprentissage de l'agent
Notation : $\mathcal{B}_{\mathbf{s_t}}$ est le vecteur regroupant la composante s_t de chacun des échantillons.

function LEARN
  $\mathcal{B} \leftarrow$ GETSAMPLES($minibatchSize$)
    ▷ Chaque échantillon est de la forme $\langle s_t\,, a_t\,, r_t\,, s_{t+1}\rangle$
  Q-values $\leftarrow \Big(\ Q$-**Network**.FORWARD$(\mathcal{B}_{\mathbf{s_t}})\Big)\Big[\mathcal{B}_{\mathbf{a_t}}\Big]$
    ▷ Contient la Q-valeur de chaque Q-état de $\mathcal{B}$
    ▷ (0 pour tous les Q-états non présents dans le lot).
  targets-values $\leftarrow \mathcal{B}_{\mathbf{r_t}} + \gamma \max\Big(\ \mathbf{target}$-$Q$.FORWARD$(\mathcal{B}_{\mathbf{s_{t+1}}})\ \Big)$
    ▷ Valeurs cibles calculées grâce au réseau cible
  loss $\leftarrow$ **targets-values** $-$ **Q-values**
  gradient $\leftarrow$ Q-**Network**.BACKWARD$(\mathcal{B}_{\mathbf{s_t}}\,,$ **loss**$)$
  updatedGradient $\leftarrow$ RMSPROP(**gradient**)
  Q-**Network**.$parameters \leftarrow$ Q-**Network**.$parameters +$ **updatedGradient**
end function

IV.3 Synthèse

Dans ce chapitre, nous avons présentés notre contribution, le système Arcades. Celui-ci répond aux deux problématiques soulevées.

D'une part, nous avons défini une représentation commune pour l'ensemble de nos données. À travers une image, il nous est maintenant possible de prendre en compte à la fois des données de bas et de haut niveau. De plus, cette représentation est résistante aux modifications de topologies ou de capteurs, permettant au système de s'adapter aisément en cas de modifications. Enfin, cette représentation compréhensible par un humain, permet à l'utilisateur de mieux appréhender et surveiller le fonctionnement du système.

D'autre part, l'utilisation d'un réseau neuronal comme système d'approximation de la fonction de Q-valeur permet au système Arcades de gérer un ensemble d'entrées de grande taille. Grâce à cela, il nous est maintenant possible de supprimer l'étape d'inférence qui était jusqu'alors nécessaire, et d'utiliser directement les données issues du réseau de capteurs.

Les réseaux de neurones sont des modèles très utilisés aujourd'hui, qui ne cessent de repousser les limites de l'apprentissage automatique. Leur capacité à construire les ca-

ractéristiques représentatives des données de manière non-supervisée permet de limiter l'impact d'une représentation inadéquate ou la mise en place de systèmes d'inférences en amont.

Dans le prochain chapitre, nous commencerons Section V.1 par évaluer les performances d'ARCADES et le comparerons, lorsque cela est possible, au système présenté Chapitre III. Par la suite, dans un souci de compréhension plus approfondie, nous présenterons des expérimentations complémentaires visant à comprendre ce qui permet à un tel système de fonctionner. Nous analyserons pour cela quelles sont les données discriminantes, comment le réseau construit les caractéristiques pertinentes et quelles sont les informations que l'on peut extraire de ces caractéristiques.

Chapitre V

Résultats et analyses

> L'intelligence est la capacité
> à s'adapter au changement.
>
> Stephen Hawking

L'approche à base d'apprentissage profond par renforcement est une approche novatrice pour la prise de décision en habitat intelligent. En ce sens, aucune autre publication, à notre connaissance, ne propose de données et de métriques, ni ne détaille le fonctionnement interne d'une telle approche, avec laquelle comparer notre expérimentation. Toutefois, Chahuara, Portet et Vacher (2017) proposent un système de prise de décision reposant sur les réseaux logiques de Markov (*Markov Logic Network* – MLN). Ce système ayant également été évalué sur le corpus « Interaction », ses performances pourront servir de point de comparaison.

Dans ce chapitre, nous commençons, Section V.1, par présenter les différentes expérimentations réalisées et comparons les résultats obtenus à ceux présentés Section III.2.3 ainsi qu'à ceux obtenus par Chahuara, Portet et Vacher (2017). Par la suite, Section V.2, nous analysons plus en détail le fonctionnement interne du système, notamment en étudiant la façon dont le réseau décompose l'image qui lui est fournie et les caractéristiques qui en sont extraites. Nous cherchons alors à savoir si ces caractéristiques sont similaires à celles qu'un expert aurait pu produire.

V.1 Évaluation du modèle profond

Comme pour le premier système, l'objectif ici est de montrer la capacité du système à s'adapter à son utilisateur. Nous suivons pour cela le même protocole, rappelé Figure V.1.

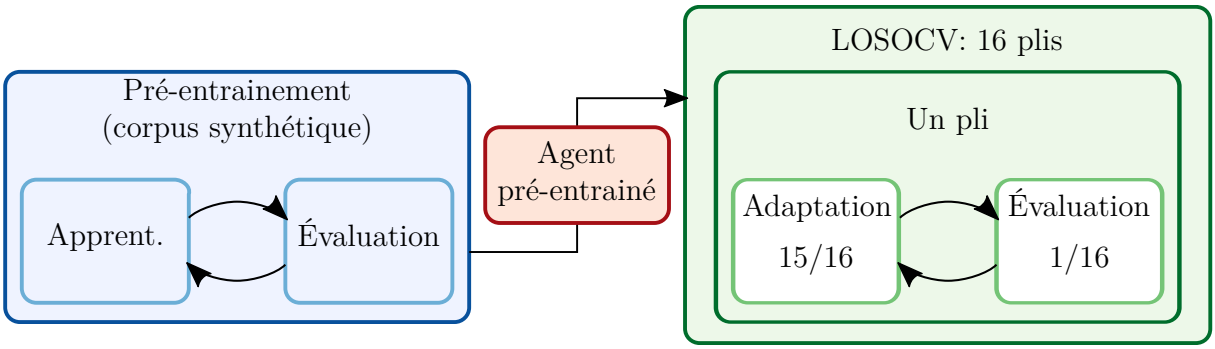


FIG. V.1 : Protocole d'apprentissage et d'évaluation

Ce protocole est réalisé deux fois, pour deux expérimentations différentes : une première utilisant les données annotées et une seconde sur les données brutes issues des capteurs.

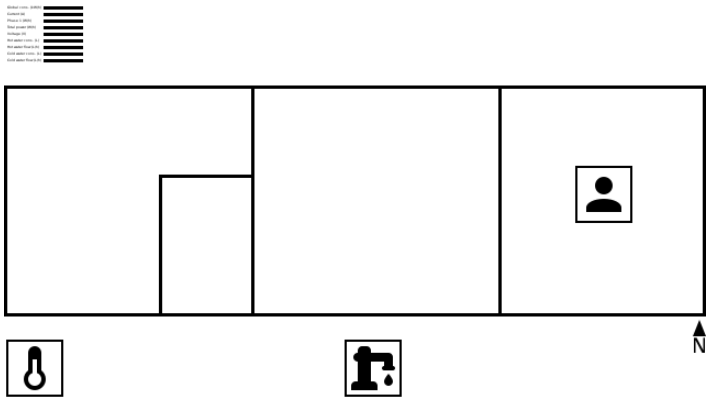
V.1.1 Expérimentation 1 : Apprentissage sur données annotées

Notre premier objectif a été de valider la nouvelle approche à l'aide de données équivalentes à celles utilisées précédemment. Ces données, que l'on appelle *données annotées*, regroupent donc les informations de localisation, d'activité et la commande prononcée par l'utilisateur ; ces informations étant suffisantes pour choisir la bonne action dans la majorité des cas. Pour cette première expérimentation, nous avons cherché à coller au plus près au système de MNIH et coll. (2015) afin de limiter les risques d'échecs qui pourraient être liés aux choix des hyper-paramètres. Ainsi, le réseau neuronal suit la même architecture, rappelée Architecture V.A1, alimenté par des images de 84 px × 84 px résultant du redimensionnement d'images comme celle présentée en Figure V.2a. La Figure V.2b montre que malgré le redimensionnement nécessaire pour permettre au réseau de traiter l'image, les informations présentes initialement restent compréhensibles par un humain.

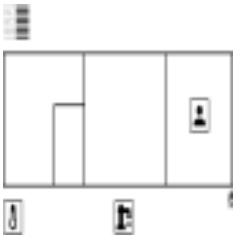
$$C_{32}[8+0/4] \rightarrow C_{64}[4+0/2] \rightarrow C_{64}[3+0/1] \rightarrow FC_{512} \rightarrow FC_{18} \tag{V.A1}$$

Tout comme pour notre précédent système, nous gardons à l'esprit que le système a pour vocation d'être utilisé par de réels utilisateurs dans un environnement interactif. En ce sens, un temps d'adaptation trop long serait un réel frein à l'adoption d'un tel système. En revanche, le phase de pré-entrainement peut être arbitrairement longue puisqu'elle ne nécessite aucune interaction réelle.

Dans notre cas, ce pré-entrainement est réalisé sur 300 000 interactions, ce qui est du même ordre de grandeur que pour notre précédent système. La mesure de la performance d'apprentissage est réalisée sur 5000 interactions totalement indépendantes (tirées aléatoirement dans l'ensemble des états possibles) toutes les 6000 interactions. Les paramètres utilisés lors de cette phase d'entrainement sont compilés dans la Table V.1a.



(a) Image avant pré-traitement, telle que présentée par l'environnement.



(b) Image après pré-traitement, telle que présentée au réseau de neurones.

FIG. V.2 : Exemple de pré-traitement appliqué aux images générées.

Une fois le modèle pré-entrainé, nous l'adaptons lors de la seconde phase. Chaque pli de la validation croisée est réalisé sur un total de 300 000 interactions issues du corpus de données réelles. L'évaluation du modèle est également réalisée toutes les 6000 interactions, sur 5000 interactions issues des données non-utilisées pour l'apprentissage. Lors de cette phase d'adaptation, nous mesurons également les performances d'apprentissage, qui peuvent indiquer un problème d'apprentissage comme notamment le sur-apprentissage (performances sur le corpus d'apprentissage très élevées mais faibles sur le corpus d'évaluation). Les paramètres sont une nouvelle fois compilés dans la Table V.1b.

TAB. V.1 : Paramètres de l'expérimentation 1.

(a) Phase de pré-entrainement

Paramètre	Valeur
Tag git	`synthetic-annot-1`
Données	synthétiques, annotées
$totalLearningSteps$	$300 \cdot 10^3$
$learningSteps$	6000
$evaluationSteps$	5000

(b) Phase de validation

Paramètre	Valeur
Tag git	`real-annot-1`
Données	réelles, annotées
$totalLearningSteps$	$300 \cdot 10^3$
$learningSteps$	6000
$evaluationSteps$	5000

Les performances de notre système sont exprimées à partir de différentes métriques.

Métrique d'apprentissage par renforcement La *récompense moyenne par épisode* est la valeur objectif de l'agent, celle qu'il cherche à maximiser. Il s'agit de la moyenne des sommes des récompenses obtenues au cours de chacun des épisodes (un ensemble indépendant de τ interactions). L'utilisation de cette métrique en lieu et place de la somme totale des récompenses permet de comparer les performances de différents agents quel que soit le nombre d'interactions d'évaluations réalisées. La plage de valeurs de cette métrique dépend principalement de la fonction de récompense. Dans cette expérimentation, la fonction de récompense renvoi -1 à chaque erreur de l'agent et $+1$ lorsque celui-ci

TAB. V.2 : Exemple de matrice de confusion faisant apparaitre les vrais positifs/négatifs et les faux positifs/négatifs pour la classe 2.

		Prédictions			
		$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$
Échantillons	$\mathcal{C}_1$	tn	fp	tn	tn
	$\mathcal{C}_2$	fn	tp	fn	fn
	$\mathcal{C}_3$	tn	fp	tn	tn
	$\mathcal{C}_4$	tn	fp	tn	tn

sélectionne la bonne action, mettant du même coup un terme à l'épisode. La récompense par épisode peut alors prendre des valeurs comprises entre $-\tau$ si l'agent ne sélectionne jamais la bonne action et $+1$ si l'agent sélectionne la bonne action du premier coup.

Métriques de classification Notre système pouvant être vu comme un système de classification, les mesures habituellement utilisées pour ces systèmes peuvent être réalisées. Ainsi, comme pour la précédente expérimentation, nous enregistrons une *matrice de confusion* à chaque étape de l'apprentissage. À partir de cette matrice, nous extrayons la valeur de prédictivité positive (précision, *precision*), mais également, le rappel et le score F1, dont nous rappelons les formules ci-dessous (BLEIK et GAUHER 2016).

$$\textbf{précision par classe} = \frac{tp_{\mathcal{C}_i}}{tp_{\mathcal{C}_i} + fp_{\mathcal{C}_i}} \qquad \textit{précision} = \overline{\textbf{précision par classe}}$$

$$\textbf{rappel par classe} = \frac{tp_{\mathcal{C}_i}}{tp_{\mathcal{C}_i} + fn_{\mathcal{C}_i}} \qquad \textit{rappel} = \overline{\textbf{rappel par classe}}$$

$$\textbf{F1 par classe} = \frac{2 \times \textbf{précision par classe} \times \textbf{rappel par classe}}{\textbf{précision par classe} + \textbf{rappel par classe}}$$

$$F1 = \overline{\textbf{F1 par classe}}$$

Où $tp_{\mathcal{C}_i}$, $tn_{\mathcal{C}_i}$, $fp_{\mathcal{C}_i}$ et $fn_{\mathcal{C}_i}$ représentent respectivement le nombre d'exemples de type vrai positif, vrai négatif, faux positif et faux négatif respectivement pour la classe $\mathcal{C}_i$ (Table V.2) et $\overline{\mathbf{x}}$ la moyenne des valeurs du vecteur $\mathbf{x}$.

En plus des valeurs à la fin de chacune des phases, la récompense par épisode ainsi que le score F1 sont présentés sous forme de graphiques afin de rendre compte de leur évolution au cours de l'apprentissage.

Les résultats sur données synthétiques, présentés Figure V.3, atteignent le plateau maximal très rapidement, dénotant un apprentissage parfait du corpus d'entrainement totalement déterministe et donc un comportement cohérent du système. La phase de validation (Figure V.4) ne permet pas, par la suite, de mettre en exergue la capacité du système à s'adapter. En effet, dès le début de cette phase le système exhibe de bonnes performances seulement diminuées par un comportement non prédictible de l'utilisateur sur un ensemble restreint d'exemples. Malgré cela, le système ARCADES stagne avec une précision de 80 % supérieure à celle obtenue par notre approche tabulaire. De plus, le score F1 de 79 % est lui largement supérieur aux 46 % d'alors, dénotant une bien meilleure spécificité de notre système. Ces performances peuvent également être mises en parallèle de celles obtenues par CHAHUARA, PORTET et VACHER (2017), qui atteignent une précision de 85 % sur un corpus comportant davantage d'incertitudes (liées à l'étape d'inférence réalisée).

Il est intéressant de noter que les performances lors de l'apprentissage stagnent à des valeurs intermédiaires. Ce comportement est lié au facteur d'exploration (l'ϵ de la stratégie ϵ-*greedy*), qui est diminué de 0,99 à 0,5 durant les 100 000 premières interactions d'apprentissages (une action sur deux est donc choisie aléatoirement) mais qui est fixé à 0 (aucune décision aléatoire) lors de l'évaluation.

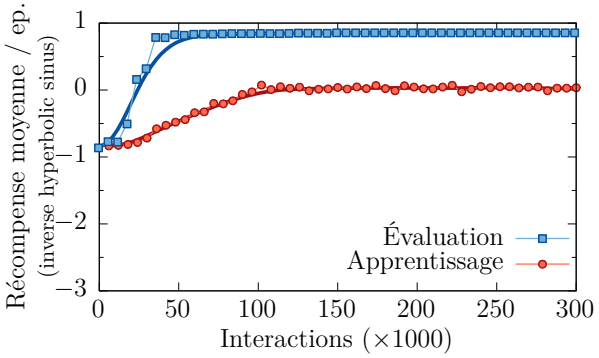
Ces premiers résultats nous permettent donc de valider cette approche à base de réseau neuronal. Toutefois, cette expérimentation ne tire pas pleinement parti des capacités des réseaux de neurones et se limite donc à la représentation des états via trois caractéristiques de haut niveau (commande, localisation et activité). Il nous reste alors à vérifier que le système puisse maintenant être utilisé dans un cadre moins supervisé en utilisant les capacités de l'apprentissage profond.

V.1.2 Expérimentation 2 : Apprentissage sur données brutes issues des capteurs

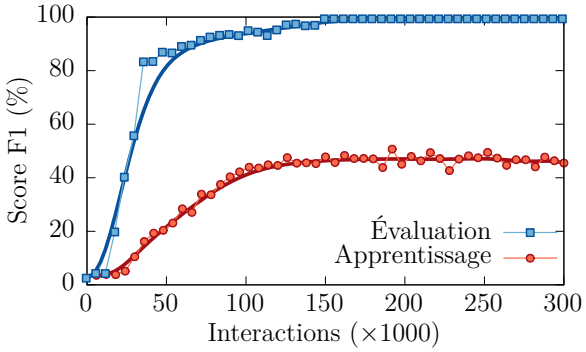
Dans cette seconde expérimentation, les données d'entrée sont des images représentant l'ensemble des capteurs de l'habitat et leur valeur (comme présentée Figure IV.5b). Toutefois, la représentation de 81 capteurs sur une carte réduite à 84×84 rend l'ensemble inutilisable pour l'utilisateur. Nous avons donc choisi d'utiliser une image plus grande

(a) Scores après entrainement

Récompense par épisode	+1
Précision	100 %
Rappel	100 %
Score F1	100 %



(b) Évolution de la *récompense moyenne par épisode*

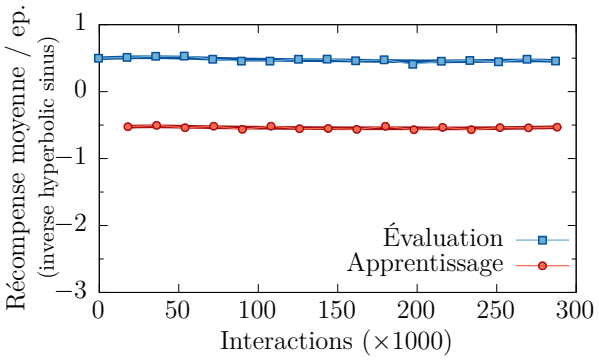


(c) Évolution du *score F1*

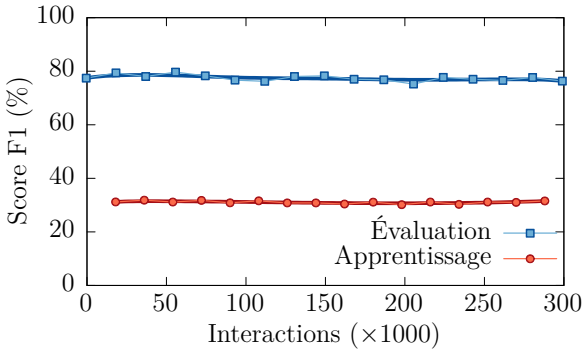
FIG. V.3 : Expérimentation 1 – Résultats de la phase de *pré-entrainement*

(a) Scores après entrainement

Récompense par épisode	0,52
Précision	80 %
Rappel	78 %
Score F1	79 %



(b) Évolution de la *récompense moyenne par épisode*



(c) Évolution du *score F1*

FIG. V.4 : Expérimentation 1 – Résultats de la phase de *validation*

redimensionnée à 256×256 pixels qui est une taille largement utilisée pour le traitement d'image par des réseaux de neurones. Le réseau a alors été adapté, pour prendre en charge ce surplus de données, afin de correspondre au modèle précédemment présenté (Architecture IV.A2) et rappelé Architecture V.A2.

$$C_{16}[8+2/4] \rightarrow C_{32}[6+1/3] \rightarrow C_{64}[3+0/2] \rightarrow C_{64}[3+0/1] \rightarrow FC_{512} \rightarrow FC_{33} \quad \text{(V.A2)}$$

Une autre particularité de l'utilisation des données brutes des capteurs est la non-synchronisation des différents flux. En effet, nous utilisons ici des données issues de trois sources différentes :

- le réseau KNX reliant l'ensemble des capteurs simples ;
- les microphones qui utilisent un réseau séparé ;
- le résultat de l'annotation pour la commande vocale.

Bien que ces trois flux aient été alignés manuellement, le résultat ne peut être aussi parfait qu'avec un corpus uniquement composé d'annotations ou généré automatiquement. Ce type de problème est particulièrement adapté aux modèles tels que les réseaux de neurones récurrents (*Recurrent Neural Networks* – RNN, Lipton, Berkowitz et Elkan 2015) qui permettraient également de prendre en charge des dépendances temporelles plus longues ou complexes (par exemple si une action de l'après-midi dépend d'un état rencontré le matin). Toutefois, leur utilisation aurait induit un sur-coût non négligeable de développement et est au-delà de l'objectif de ce travail de thèse.

Une solution de contournement consiste à fournir au réseau un historique d'une longueur donnée pour prendre une décision en lieu et place d'une unique visualisation à l'instant t. Ainsi, deux secondes avant une prise de décision, l'état de l'environnement est fourni à l'agent. L'agent doit alors combiner les trois images (les deux en amont de la décision et celle à l'instant de la décision) avant de la fournir au réseau. La Figure V.5 présente les trois images de base et le résultat du pré-traitement par l'agent. Chaque image initiale n'étant composée que de noir et blanc, celles-ci sont réduites à des images monochromes (1 seul canal). Leur combinaison génère alors une image à 3 canaux, interprétés comme des canaux rouge, vert et bleu. Ainsi, sur les images résultantes, le canal rouge correspond à l'état il y a 2 secondes, le vert il y a une seconde et le bleu l'état actuel. Les parties blanches ou noires sont celles communes au cours de ces trois secondes.

L'utilisation d'états ne nécessitant pas d'action (les deux secondes avant la prise de décision) fait alors apparaitre le cas de l'action consistant à ne rien faire. En effet, l'utilisation

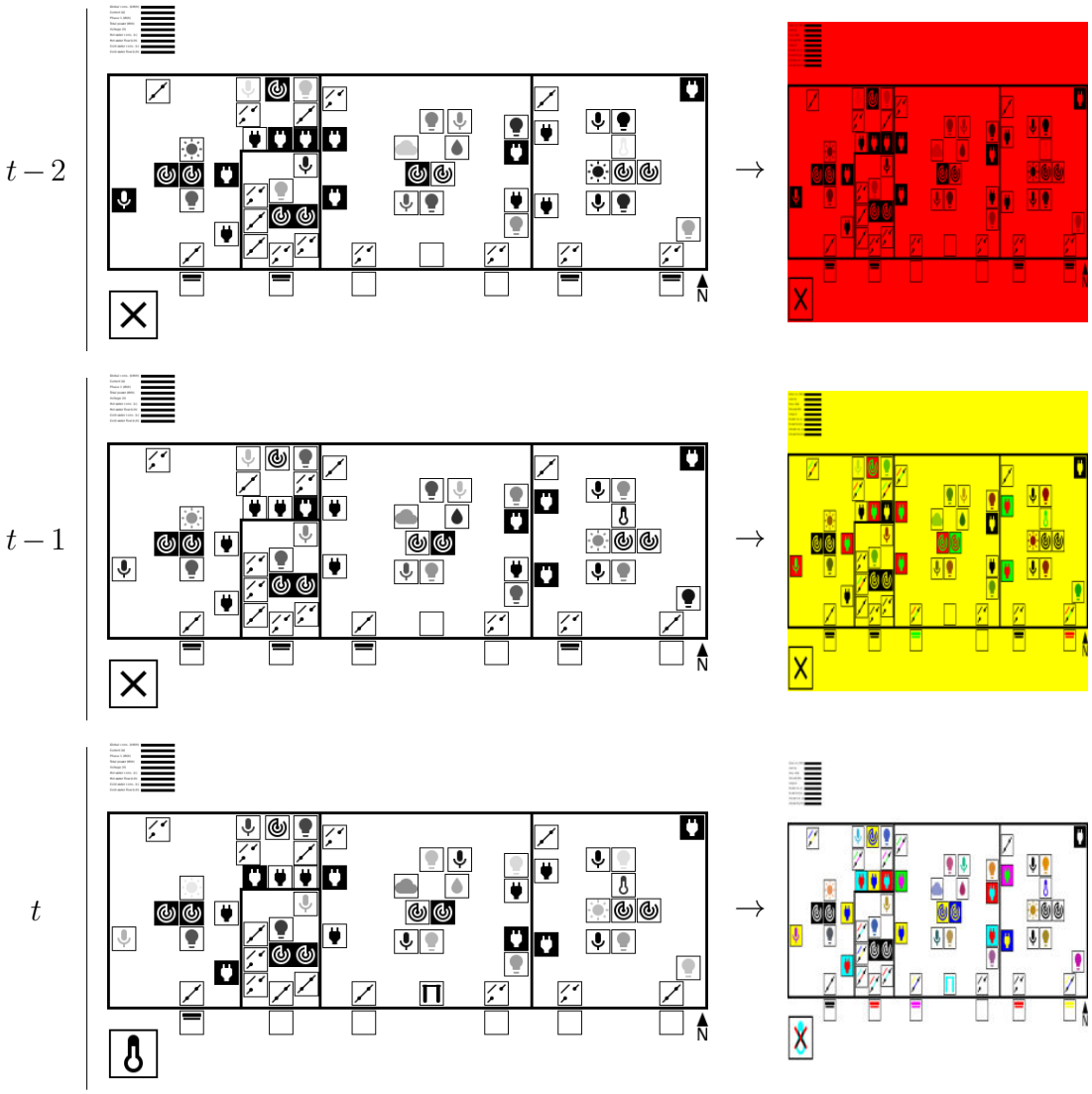


FIG. V.5 : Exemple de pré-traitement d'un ensemble de trois images.

du système de manière réactive (seulement lorsqu'une commande vocale valide est détectée) permettait de ne pas tenir compte de ce cas-là. Le cas « Ne rien faire » apparaissant 2 fois avant chacune des autres actions, cette classe d'(in)action est donc largement majoritaire. En utilisant la même fonction de récompense que précédemment le système apprend très rapidement à ne jamais rien faire, ce qui est un comportement normal pour un système statistique comme un réseau de neurones. Ce n'est en revanche pas le comportement attendu. Pour résoudre ce problème de classes non-équilibrées, une solution classique consiste à pondérer la fonction de coût selon la représentativité de la classe. Nous nous sommes basé sur une telle approche afin de définir une nouvelle fonction de récompense présentée Algorithme V.1. Un nouvel argument nommé `no_action_penalty` permet de définir la valeur de la récompense lorsque le système choisit de ne rien faire alors qu'une action est attendue. Dans notre cas nous utilisons une valeur dépendante du nombre d'actions :

`no_action_penalty` $= -2 \times \Big(|\mathcal{A}| - 1\Big) \times$ `history_length`, où `history_length` correspond au nombre de secondes fournies comme historique à l'agent. De la même manière, nous utilisons une faible récompense négative même lorsqu'aucune action n'est attendue afin d'inciter le système à essayer d'agir.

Algorithme V.1 Fonction de récompense permettant de prendre en compte la classe majoritaire *Ne rien faire*.

```
function getReward
    if true_action est Ne rien faire then
        if provided_action est Ne rien faire then
            reward ← −0,005
        else
            reward ← −1
        end if
    else
        if provided_action est Ne rien faire then
            reward ← no_action_penalty
        else if true_action = provided_action then
            reward ← +1
        else
            reward ← −1
        end if
    end if
    return reward
end function
```

L'utilisation d'une fonction de récompense pondérée a permis de corriger le problème de classe majoritaire. Toutefois, le temps d'apprentissage alors nécessaire était excessivement long. Il s'agit ici de l'un des inconvénients des réseaux de neurones souvent évoqué (Pritzel et coll. 2017). Les limitations liées au matériel disponible nous ont obligé à chercher un moyen de tester notre approche sans pour autant nécessiter plusieurs semaines d'entrainement. La solution la plus évidente a été de réduire les espaces de recherche en limitant d'une part les états possibles et d'autre part les actions disponibles.

Restriction des ensembles d'états et d'actions Nous nous sommes alors concentrés sur la prise de décision dans les cas les plus fréquents et les plus ambigus. La majorité des interactions présentes dans le corpus Sweethome prend place dans la cuisine ou la chambre de l'appartement, ne laissant que peu d'exemples d'interactions dans le bureau. De même, la majorité des interactions sont liées à la gestion des lumières, des volets et du système multimédia. Ainsi, lors de la génération du corpus synthétique pour la phase de pré-entrainement, nous ne générons que des exemples pour lesquels :

- l'utilisateur se situe soit dans la cuisine soit dans la chambre ;

TAB. V.3 : Actions possibles dans un environnement restreint

Commande	Actionneur	Index
Allumer la lumière	Lumière de l'évier de la cuisine	1
Allumer la lumière	Plafonnier de la cuisine	2
Allumer la lumière	Lampes de chevet	3
Allumer la lumière	Plafonnier de la chambre	4
Éteindre la lumière	Lumière de l'évier de la cuisine	5
Éteindre la lumière	Plafonnier de la cuisine	6
Éteindre la lumière	Lampes de chevet	7
Éteindre la lumière	Plafonnier de la chambre	8
Allumer la radio	Dans la chambre	9
Éteindre la radio	Dans la chambre	10
Ouvrir les volets	Volets de la cuisine	11
Ouvrir les volets	Volets de la chambre	12
Fermer les volets	Volets de la cuisine	13
Fermer les volets	Volets de la chambre	14
Ne rien faire	nulle part	15

- la commande prononcée a trait à la gestion de la lumière, des volets ou de la radio.

Enfin, les activités possibles de l'utilisateur sont aussi limitées aux principales activités se déroulant dans les deux pièces principales, qui sont : l'utilisateur est en train de manger, l'utilisateur est en train de dormir ou l'utilisateur est en train de ranger.

De telles restrictions permettent par la suite de limiter les actions possibles pour l'agent aux actions se déroulant dans la cuisine ou la chambre et ayant un rapport avec la lumière, les volets ou la radio. Un nouvel ensemble d'actions, présenté Table V.3, a alors été défini et chaque décision qui ne correspondait pas à l'une de ces actions était remplacée par l'action attendue *Ne rien faire*. Cette classe d'action étant déjà largement majoritaire, l'impact d'un tel choix est négligeable quant à la représentativité des classes mais limite grandement le nombre d'actions possibles, le diminuant de 33 actions initialement à seulement 15.

L'utilisation d'ensembles restreints pourrait laisser supposer que notre système n'est pas capable de passer à l'échelle et donc que l'approche par réseau de neurones n'est pas pertinente comparée à une approche tabulaire classique. Toutefois, bien que le nombre d'états différents possibles soit limité, ceci n'influence pas la taille des entrées de l'agent qui restent des images de $639 \times 360 \times 3$ px. Nous pensons donc que le système ne perd pas ici sa capacité à passer à l'échelle. De la même manière, la réduction du nombre d'actions disponibles permet de grandement réduire le temps d'apprentissage, en limitant l'effet de l'exploration par l'agent des actions non maximales. Ainsi, rien ne laisse penser

que le système ne puisse gérer un plus grand nombre d'actions à condition que le temps d'apprentissage lors de la phase de pré-entrainement ne soit pas un frein.

Toutefois, malgré une amélioration des performances lors des premières expérimentations utilisant ces ensembles restreints, le système ne semblait pas pouvoir converger, les performances du système oscillant aux alentours de (20 ± 10) % de score F1. Afin de corriger ce problème, nous avons cherché les hyper-paramètres qui influençaient un tel comportement et tenté de les optimiser.

Optimisation des hyper-paramètres D'après Mnih et coll. (2015), deux hyper-paramètres entrent principalement en jeu pour paramétrer la vitesse de convergence du réseau : `target_q` et `minibatch_size`. Ces deux paramètres sont directement liés à la méthode d'apprentissage du réseau et leur influence a déjà été évoquée Section IV.1.2. En effet, la valeur de `target_q` est le nombre d'interactions durant lesquelles une ancienne version du réseau de décision est utilisée. En utilisant une valeur nulle, les adaptations du modèle sont immédiatement répercutées sur le calcul des valeurs cibles. Même si un tel comportement semble souhaitable a priori, il s'avère que plus que de faire accélérer la convergence, une valeur trop basse du `target_q` limite la stabilité du réseau et peut même le pousser à diverger. À l'inverse, une valeur trop élevée de `target_q` aurait un impact néfaste sur la vitesse d'apprentissage, en retardant davantage la prise en compte des dernières interactions. Le second paramètre, `minibatch_size`, permet de définir la taille des lots utilisés pour l'apprentissage, c'est-à-dire la taille de l'ensemble $\mathcal{B}$. Globalement l'utilisation d'un lot de grande taille permet une meilleure convergence, l'idéal étant d'utiliser l'ensemble des exemples disponibles (apprentissage par lot, *batch learning*). Notre système devant apprendre à partir d'exemples fournis en continu, une telle approche n'est pas envisageable. En revanche l'utilisation de mini-lots permet de s'approcher d'un résultat similaire en lissant le gradient sur un ensemble d'exemples. Ainsi, une valeur élevée du paramètre `minibatch_size` améliore la convergence du réseau mais au prix d'un temps d'exécution plus long. En effet, l'ensemble des exemples présents dans $\mathcal{B}$ doit passer une première fois à travers le réseau, puis être rétro-propagé de manière à calculer le gradient. Le temps d'apprentissage est donc directement lié à la taille de $\mathcal{B}$, selon la formule :

$$tps_{learning} \propto 2 \cdot |\mathcal{B}|$$

Nous avons donc réalisé différentes expérimentations avec différentes valeurs des paramètres afin de mesurer leur impact sur deux principales métriques, le score F1 et le temps nécessaire à l'exécution. Les graphiques en Figure V.6 montrent alors l'évolution de ces métriques en fonction des hyper-paramètres utilisés. La tendance la plus évidente est l'amélioration du score F1, de même que le temps d'exécution avec l'augmentation de la valeur

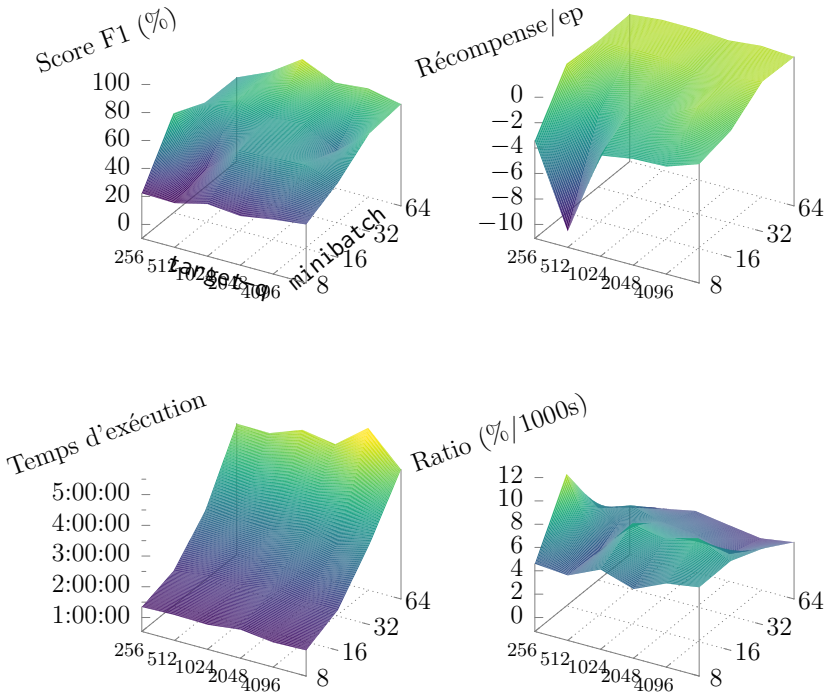


FIG. V.6 : Évolution du score F1 et du temps d'exécution en fonction de la valeur de `target_q` et `minibatch_size`

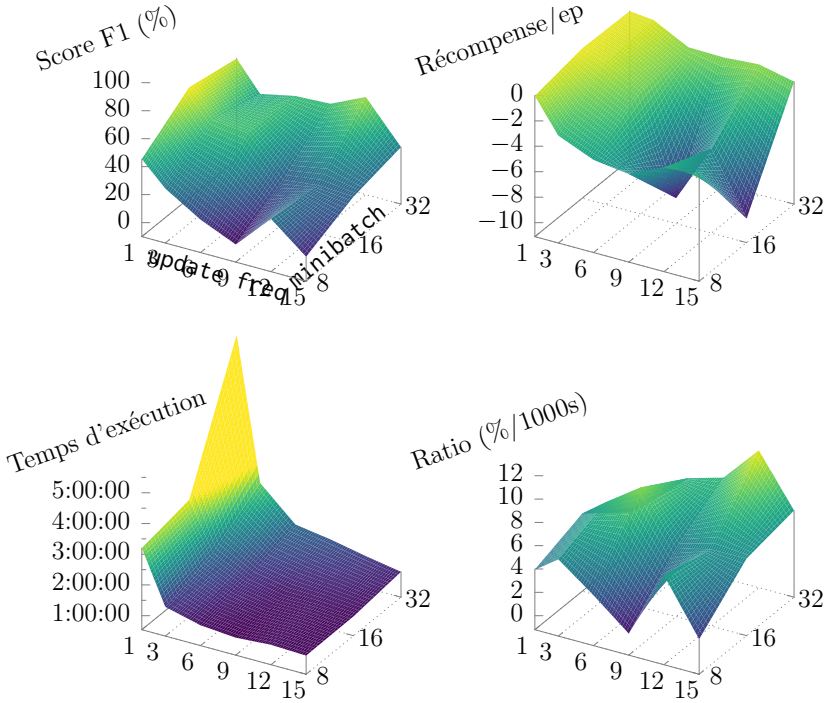


FIG. V.7 : Évolution du score F1 et du temps d'exécution en fonction de la valeur de `update_freq` et `minibatch_size`

de `minibatch_size`. Afin de déterminer une valeur permettant un bon compromis entre performance et rapidité, nous traçons également une courbe mettant en rapport le score et le temps d'exécution. Bien que cette courbe exhibe un pic pour une valeur de `target_q` faible, ce résultat semble davantage lié aux conditions de l'expérimentation (faible charge du serveur) et ne semble pas représentatif de la tendance générale. Nous avons ainsi choisi d'utiliser une valeur élevée pour `target_q` et une valeur de `minibatch_size` de 16.

Toutefois, de meilleures performances peuvent être obtenues avec une taille de lots plus élevée. Pour limiter le temps d'exécution, un autre hyper-paramètre peut être modifié, `update_freq`. Ce paramètre définit le nombre d'interactions effectuées avant de lancer une phase d'apprentissage (équivalent à la valeur de *maxSteps* dans l'Algorithme III.1), ce qui le lie au temps d'exécution selon la formule :

$$tps_{learning} \propto 1/updateFreq$$

Il est ainsi possible de conserver un temps d'exécution raisonnable tout en augmentant la taille des lots utilisés pour l'apprentissage et donc en améliorant la capacité de convergence du réseau. Les résultats obtenus avec les différentes valeurs sont résumés par la Figure V.7.

L'évolution des différents paramètres est résumé Table V.4.

Tab. V.4 : Évolution des hyper-paramètres

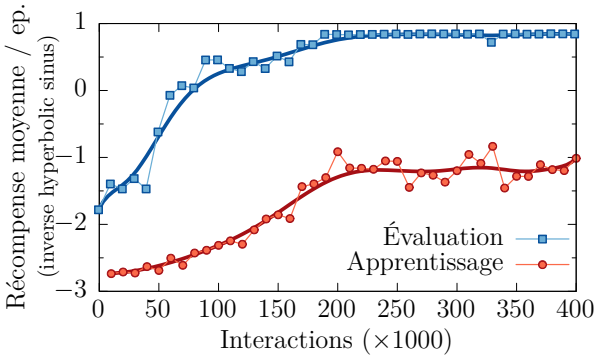
	Valeur initiale	Optimisation 1	Optimisation 2
tag git	`graphic-labelled-sensor-5`	`hyperparam-optim-1`	`hyperparam-optim-2`
`target_q`	1024	4096	4096
`minibatch_size`	64	16	32
`update_freq`	3	3	12

Une fois le système totalement adapté à nos besoins, il nous était alors possible de lancer une nouvelle expérimentation pour valider notre approche.

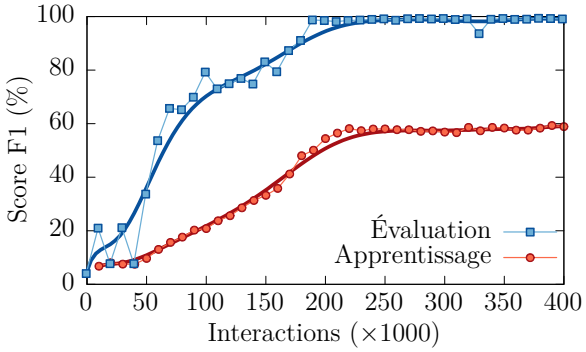
Expérimentation Nous avons alors exécuté une phase de pré-entrainement sur un corpus synthétique. Ce corpus est ici davantage bruité que lors de la première expérience. En effet, nous ne contraignons une valeur que pour un sous-ensemble de capteurs qui nous semblent pertinents pour la prise de décision et générons des données aléatoires pour les autres. Le système demande alors un plus grand nombre d'interactions avant d'atteindre un plateau. Nous réalisons donc un total de 400 000 interactions en évaluant notre agent toutes les 10 000 interactions (Table V.5). Le modèle généré est ensuite adapté lors d'une phase de validation croisée comportant également 400 000 interactions dans chaque pli et en évaluant le système toutes les 10 000 interactions.

(a) Scores après entrainement

Récompense par épisode	0,98
Précision	100 %
Rappel	100 %
Score F1	100 %



(b) Évolution de la *récompense moyenne par épisode*



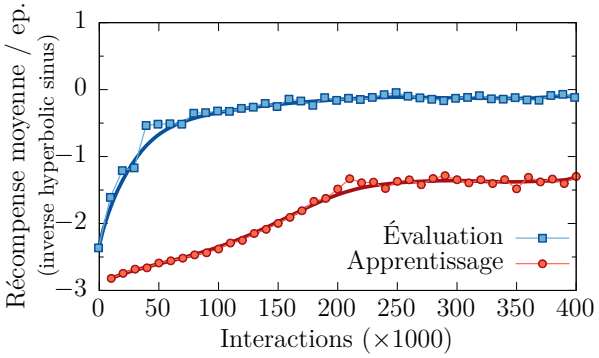
(c) Évolution du *score F1*

FIG. V.8 : Expérimentation 2 – Résultats de la phase de *pré-entrainement* sur environnement restreint

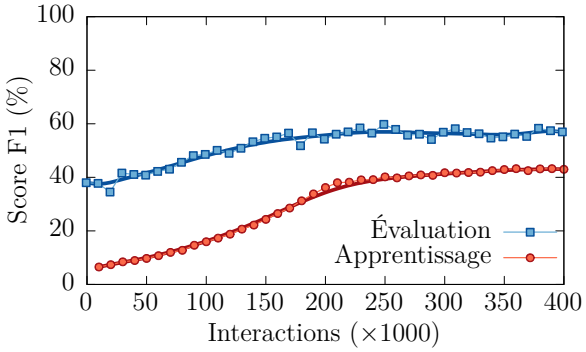
---

(a) Scores après entrainement

Récompense par épisode	−0,09
Précision	54 %
Rappel	62 %
Score F1	58 %



(b) Évolution de la *récompense moyenne par épisode*



(c) Évolution du *score F1*

FIG. V.9 : Expérimentation 2 – Résultats de la phase de *validation* sur environnement restreint

TAB. V.5 : Paramètres de l'expérimentation 2.

(a) Phase de pré-entrainement

Paramètre	Valeur
Tag git	`synthetic-raw-1`
Données	synthét., brutes, restreintes
totalLearningSteps	$400 \cdot 10^3$
learningSteps	$10 \cdot 10^3$
evaluationSteps	5000

(b) Phase de validation

Paramètre	Valeur
Tag git	`real-raw-1`
Données	réelles, brutes, restreintes
totalLearningSteps	$400 \cdot 10^3$
learningSteps	$10 \cdot 10^3$
evaluationSteps	5000

Les graphiques dans la Figure V.8 exhibent les mêmes caractéristiques que dans l'expérimentation 1. Lors de la phase de pré-entrainement, le système commence par agir de manière aléatoire lors d'une phase très exploratoire. Après quoi, les connaissances acquises lui permettent d'atteindre un plateau.

La phase de validation (Figure V.9), quant à elle, montre bien la capacité d'adaptation du système. À partir de performances relativement dégradées (score F1 de moins de 40 %), le système s'adapte aux nouvelles données avant d'atteindre un nouveau plateau avec un score F1 de près de 60 %. Bien que ces performances soient moindres comparées aux résultats de la première expérimentation, cette approche reste meilleure que la performance de référence établie par notre précédent système (Chapitre III) qui atteignait un score F1 de 46 %.

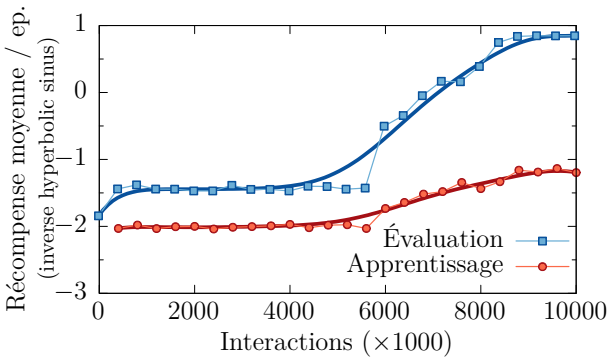
Expérimentation 2 bis – Suppression des restrictions Pour rappel, les résultats que nous venons de présenter ont été obtenus en limitant à la fois le nombres d'états possibles ($|\mathcal{S}|$) et le nombre d'actions possibles ($|\mathcal{A}|$). Cette restriction nous a permis de réaliser de nombreux tests afin de déterminer le plus justement possible la valeur optimale des hyper-paramètres d'ARCADES. L'expérimentation suivante s'est déroulée sur la totalité des ensembles d'états et d'actions, permettant de vérifier la capacité du système à passer à l'échelle.

La taille des images fournies au système ne change pas lors de cette expérimentation (256 px × 256 px), en revanche il existe une plus grande variété des images. En effet, sur un ensemble restreint, des images étaient fournies uniquement quand l'utilisateur se trouvait dans la chambre ou la cuisine. Ici, des images sont également fournies lorsque l'utilisateur est dans le bureau, générant alors de nouvelles images possibles. Le même mécanisme s'applique pour les autres éléments du contexte (activité, commande prononcée). L'ensemble d'actions disponibles passe lui de 15 (Table V.3) à 33 actions (Table III.1 plus « Ne rien faire »).

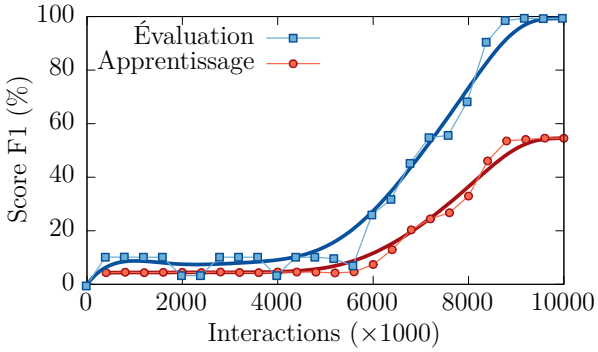
On réalise ici les deux mêmes étapes que précédemment. Le pré-entrainement s'effectue sur $10\,000\,000 = 10 \cdot 10^6$ d'interactions (contre $400\,000 = 400 \cdot 10^3$ précédemment), avec

(a) Scores après entrainement

Récompense par épisode	0,99
Précision	100 %
Rappel	100 %
Score F1	100 %



(b) Évolution de la *récompense moyenne par épisode*



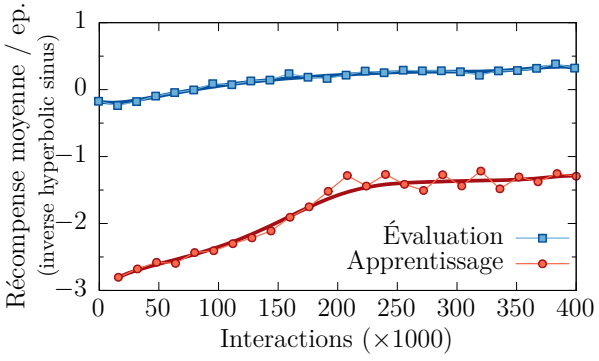
(c) Évolution du *score F1*

FIG. V.10 : Expérimentation 2 bis – Résultats de la phase de *pré-entrainement* sur environnement non-restreint

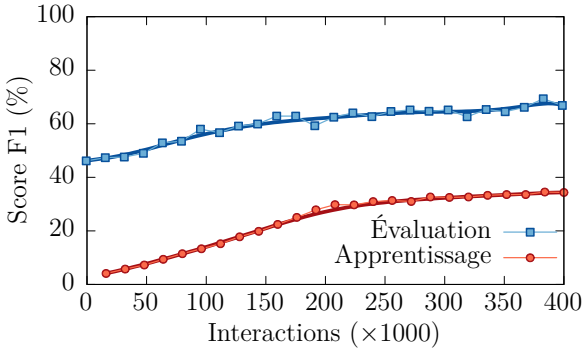
---

(a) Scores après entrainement

Récompense par épisode	0,36
Précision	68,91 %
Rappel	66,36 %
Score F1	67,49 %



(b) Évolution de la *récompense moyenne par épisode*



(c) Évolution du *score F1*

FIG. V.11 : Expérimentation 2 bis – Résultats de la phase de *validation* sur environnement non-restreint

TAB. V.6 : Paramètres de l'expérimentation 2 bis.

(a) Phase de pré-entrainement

Paramètre	Valeur
Tag git	`synthetic-raw-full-1`
Données	synthét., brutes
$totalLearningSteps$	$10 \cdot 10^6$
$learningSteps$	$100 \cdot 10^3$
$evaluationSteps$	5000

(b) Phase de validation

Paramètre	Valeur
Tag git	`real-raw-full-1`
Données	réelles, brutes
$totalLearningSteps$	$400 \cdot 10^3$
$learningSteps$	4000
$evaluationSteps$	5000

une évaluation toute les $100 \cdot 10^3$ interactions (Table V.6). Il est intéressant de noter que cette phase de pré-entrainement est extrêmement longue comparée aux précédentes. Il faut en effet plus $8 \cdot 10^6$ interactions (près de 100 jours à raison d'une décision par seconde) pour que le système apprenne un comportement cohérent (Figure V.10) contre environ $200 \cdot 10^3$ sur dans l'environnement restreint (Figure V.8). Cette différence peut s'expliquer par deux facteurs. D'une part, l'augmentation du nombre d'actions disponibles ($|\mathcal{A}|$) oblige le système à explorer davantage, afin de trouver la meilleure action. D'autre part, ARCADES doit également évaluer l'utilité de l'information fournie par des capteurs inutiles jusqu'alors, notamment ceux présents dans la chambre. Réalisée de manière non interactive, cette phase de pré-entrainement a tout de même duré près de 4 jours et 16 heures.

La phase de validation a pour sa part été plus courte, exécutée sur $400 \cdot 10^3$ interactions comme pour l'expérimentation sur données restreintes. Le temps d'adaptation est alors comparable au temps d'adaptation obtenu précédemment, de l'ordre de $200 \cdot 10^3$ à $300 \cdot 10^3$ interactions (Figure V.11). De plus, les performances finales du système sont également meilleures qu'en utilisant un environnement restreint (0,36 contre $-0{,}09$ de récompense moyenne).

La matrice de confusion présentée en Figure V.12 permet également de comparer cette approche neuronale à notre approche tabulaire détaillée dans le Chapitre III. On y voit une nouvelle fois une diagonale nettement marquée, concordante avec les scores obtenus, faisant apparaitre des confusions récurrentes concernant les mêmes actions que précédemment. En revanche, on peut constater une meilleure performance concernant les actions 11 et 12 ainsi qu'une dispersion plus continue du bruit. Cette particularité tend à montrer que l'approche neuronale a une meilleure capacité de généralisation que l'approche tabulaire expérimentée en premier lieu.

V.1.3 Synthèse

À travers ces expérimentations, notre approche a prouvé sa validité pour la réalisation d'un système de décision adaptatif. Ainsi, notre système est capable :

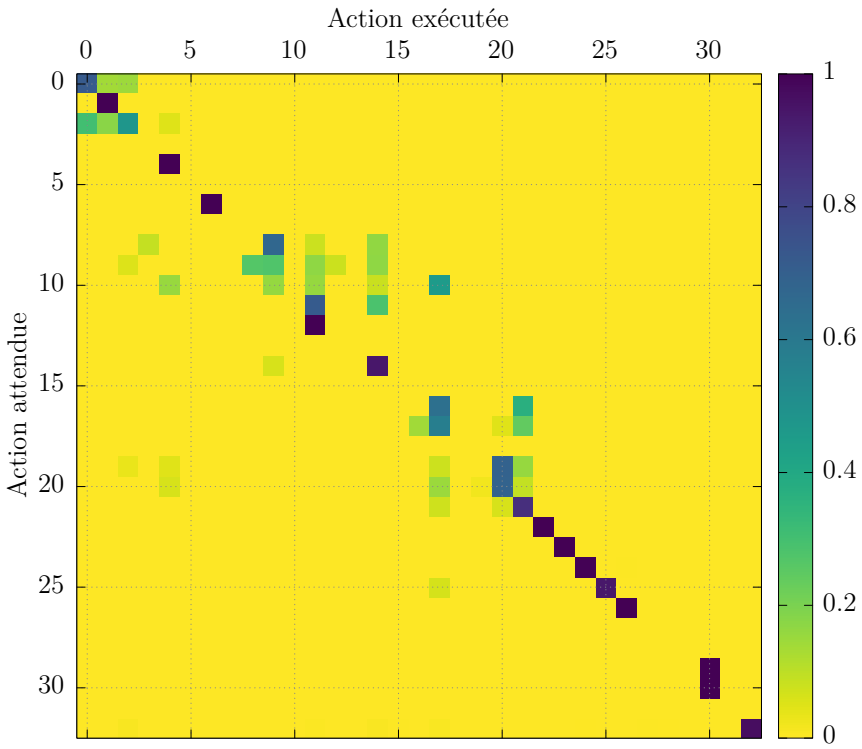


FIG. V.12 : Matrice de confusion pour la tâche de classification. Les index des différentes actions sont disponibles dans la Table III.1.

- de supporter un ensemble d'entrées de grande taille ;
- de générer une représentation globale, pour l'ensemble des données, compréhensible par l'utilisateur ;
- d'apprendre le comportement adéquat vis-à-vis des préférences de l'utilisateur ;
- d'adapter son comportement continuellement pour toujours converger vers le comportement optimal.

Avec des performances sur données annotées meilleures que celles obtenues par notre précédent système et comparables à celle de CHAHUARA, PORTET et VACHER (2017), le système ARCADES se place comme une solution performante pour la prise de décision. Les résultats obtenus par la suite, sur les données brutes issues des capteurs, restent raisonnablement bons et prouvent la capacité du système à tirer profit des données brutes des capteurs, telles quelles, sans nécessité d'inférence et ce, seulement à partir de signaux de renforcement. S'ils ne peuvent être suffisants pour envisager une application directe d'ARCADES, ils ouvrent cependant la voie à de nouvelles méthodes reposant sur l'apprentissage profond par renforcement. Certaines améliorations, détaillées Chapitre VI, peuvent ainsi être envisagées afin d'améliorer l'utilisabilité du système (temps de convergence, meilleure adaptation, méthode d'interaction).

En revanche, comme souvent pour les systèmes neuronaux, il est difficile de comprendre les informations utilisées ou extraites par le système lors de sa prise de décision. Il est alors intéressant d'étudier plus en détail les modèles résultants de ces différents apprentissages pour en comprendre de manière plus approfondie les mécaniques sous-jacentes. Les résultats de ces études sont ainsi présentés dans la section suivante.

V.2 Expérimentations complémentaires

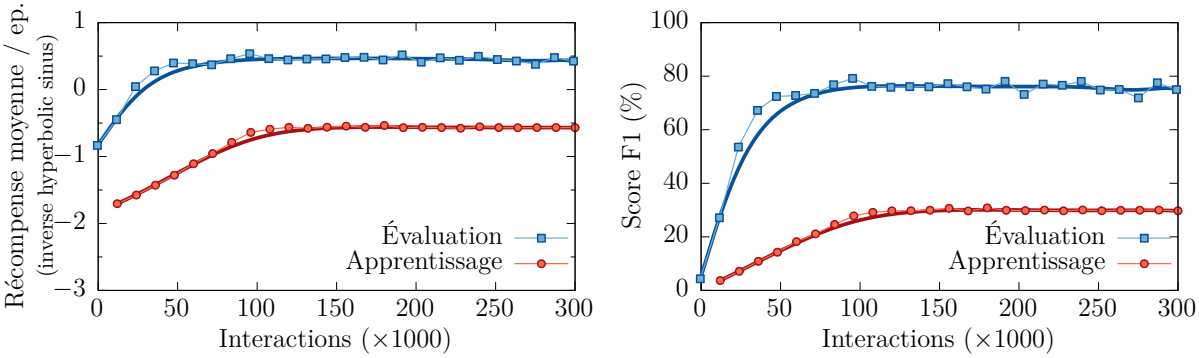
Contrairement à la table de valeurs obtenues grâce au *Q-Learning* tabulaire, les paramètres de notre réseau de neurones sont plus difficiles à interpréter. Cette remarque est régulièrement adressée aux modèles neuronaux et différentes méthodes ont été élaborées afin de mieux appréhender leur fonctionnement et notamment celui des réseaux convolutifs (Karpathy 2015). Dans cette section, nous présenterons les résultats d'expérimentations complémentaires visant à mieux comprendre le fonctionnement de notre système.

Dans un premier temps (Section V.2.1), nous étudierons l'utilité de pré-entrainer notre modèle sur des données artificielles. Par la suite (Section V.2.2), nous montrerons que si Arcades est capable de s'adapter à l'utilisateur, il est également capable de s'adapter à des changements dans la topologie des capteurs. Enfin, nous chercherons à savoir (Section V.2.3) si les caractéristiques générées par le réseau concordent avec celles fabriquées par les experts du domaine.

V.2.1 Impact du pré-entrainement

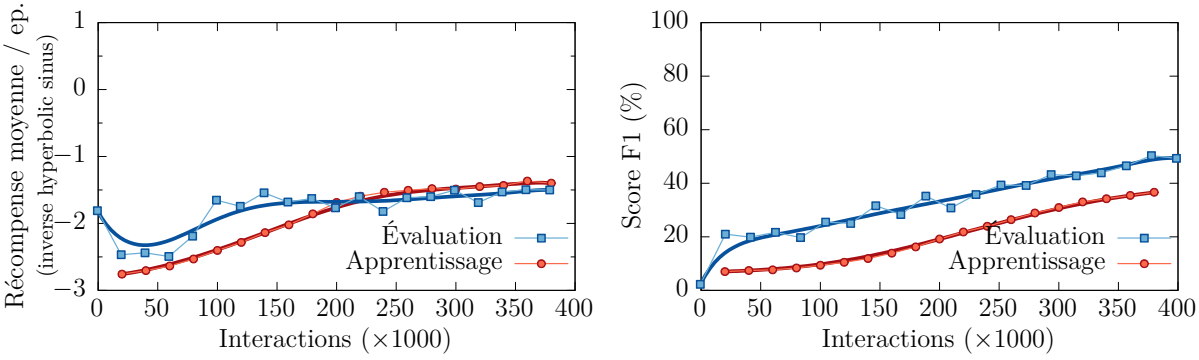
Chacune des expérimentations présentées précédemment repose sur une étape de pré-entrainement permettant au système d'être initialisé avec un comportement cohérent. Si l'expérimentation 2 bis semble montrer qu'une telle étape préliminaire permet au système de s'adapter en un temps raisonnable quelle que soit la taille des ensembles de recherches ($\mathcal{A}$ et $\mathcal{S}$), un tel avantage n'est pas aussi explicite sur les expérimentations réalisées en amont. En effet, au cours de la première expérimentation nous avons vu qu'après l'acquisition de ce comportement le système ne parvient pas, sur les données réelles, à s'adapter plus justement à l'utilisateur. En outre, le temps de convergence de la phase de pré-entrainement est relativement faible, moins de 100 000 interactions. Nous avons donc chercher à voir quels étaient les impacts de cette étape de pré-entrainement sur le temps de convergence de la phase de validation.

Pour cela, nous avons réalisé la phase de validation à partir d'un modèle qui n'avait pas été pré-entrainé. Les résultats présentés Figure V.13 sont issus de l'expérience utilisant des images générées à partir des annotations (tag git : `real-annot-nopretrain`), tandis que



(a) Récompense moyenne par épisode (b) Score F1

FIG. V.13 : Résultats sur données réelles annotées



(a) Récompense moyenne par épisode (b) Score F1

FIG. V.14 : Résultats sur données réelles brutes issues des capteurs

la Figure V.14 montre les résultats pour des images à base de données brutes issues des capteurs dans un environnement restreint (tag git : `real-raw-nopretrain`).

On peut noter que dans le premier cas, le pré-entrainement permet de limiter le temps « d'adaptation », mais n'influence pas les performances du système *in fine*. En effet, le système converge très rapidement vers des valeurs similaires à celles obtenues à partir d'un modèle pré-entrainé, sans les dépasser ou être bien en deçà. Il semblerait donc que dans cette configuration le pré-entrainement n'ait aucun effet néfaste sans pour autant être un facteur primordial d'efficacité. Son omission ne fait qu'augmenter le temps de convergence et ceci de manière assez légère.

À l'inverse, les résultats obtenus à partir des données brutes des capteurs semblent indiquer un réel intérêt dans l'utilisation d'un corpus de données générées. Alors que l'adaptation après pré-entrainement atteint une performance acceptable (supérieure à un score F1 de 46 %) en à peine 100 000 interactions, il en faut plus de 300 000 pour un modèle qui n'a pas été pré-entrainé. De même, les performances finales du système sont largement inférieures à celles obtenues lors de la validation après pré-entrainement. On voit donc que l'utilisation d'une phase de pré-entrainement a un réel bénéfice dans le cas où

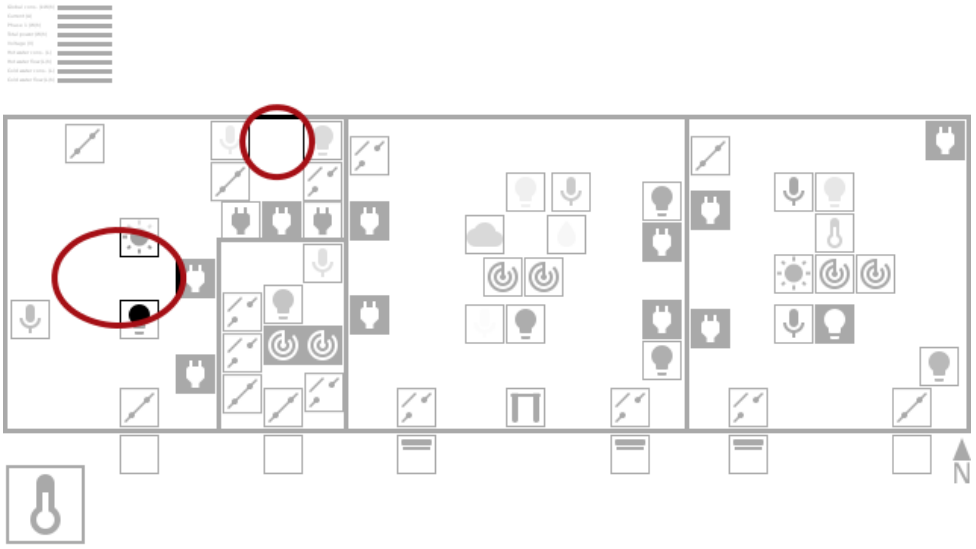


FIG. V.15 : Exemple d'image d'entrée avec trois capteurs (détecteurs de mouvement) manquants.

nous utilisons des données issues directement des capteurs, corroborant ainsi l'hypothèse formulée sur la base des résultats de l'expérimentation 2 bis.

Les expérimentations présentées jusqu'ici se sont concentrées sur l'adaptation à l'utilisateur, ce qui n'est qu'un seul des objectifs de nos travaux.

V.2.2 Adaptation au matériel

Un des objectifs de notre système était l'adaptation à l'habitat et aux capteurs. Nous cherchons ici à vérifier la capacité d'ARCADES à adapter son modèle dans le cas de disparitions de capteurs. Cette disparition peut apparaitre en cas de désinstallation d'un capteur mais également en cas de panne.

Pour cela, nous réalisons quatre expérimentations différentes. Pour chacune d'elle, nous utilisons un modèle pré-entrainé (sur des données brutes synthétiques (tag git : `synthetic-raw-full-1`)). Nous générons ensuite de nouvelles données synthétiques dans lesquelles certains capteurs sont absents, comme le montre la Figure V.15.

La première expérimentation (tag git : `faulty-PIR-full`) ne tient plus compte des trois détecteurs de mouvement présents dans la cuisine. La Figure V.16 présente les résultats qui mettent une nouvelle fois en évidence la capacité du système à s'adapter. Après une baisse de performances de l'ordre de 30 % de score F1 lors des premières évaluations, le système atteint rapidement son niveau d'origine.

Dans les deux autres expériences, les microphones de la cuisine ne fournissent plus d'informations acoustiques (Figure V.17, tag git : `faulty-microphone-full`), puis ce sont les capteurs d'ouverture de porte du placard et du réfrigérateur qui sont inactifs (Figure V.18, tag

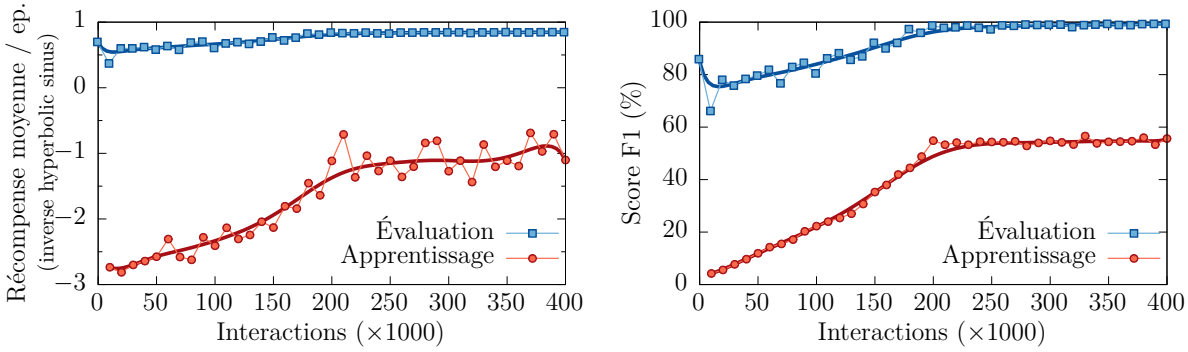


FIG. V.16 : Performances sans détecteur de mouvement

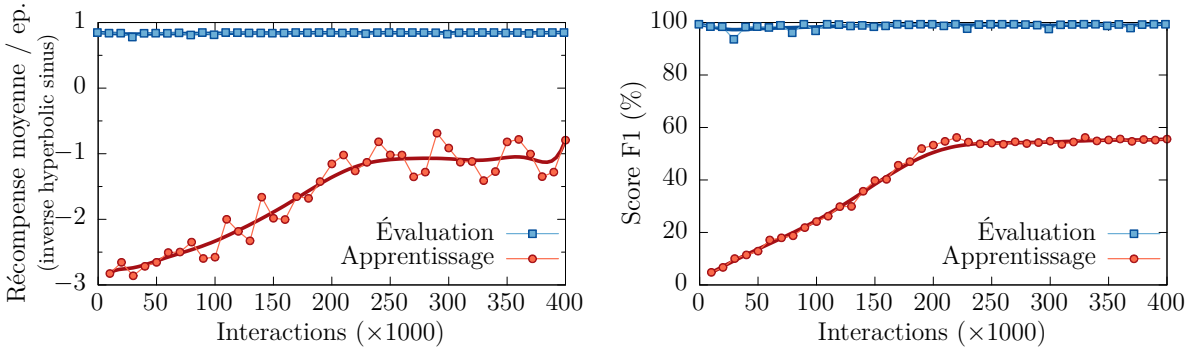


FIG. V.17 : Performances sans bruit acoustique

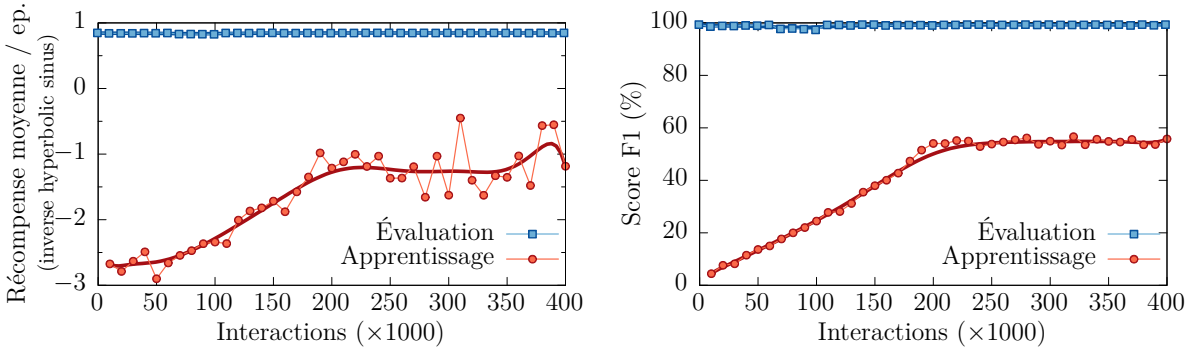


FIG. V.18 : Performances sans détecteur d'ouverture de porte

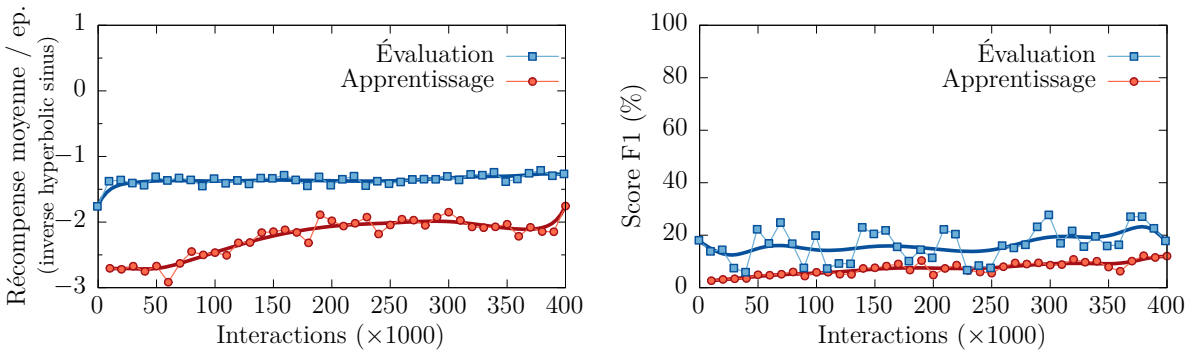


FIG. V.19 : Performances sans reconnaissance de commande

git : `faulty-switches-full`). Ces derniers sont particulièrement actifs lorsque l'utilisateur est en train de ranger ou nettoyer l'appartement, et on s'attend ainsi à ce que leur absence ait un effet néfaste sur les performances. En revanche, la très légère baisse de performance (pouvant être assimilée à du bruit) semble indiquer qu'aucune de ces informations n'est utilisée pour la prise de décision.

La quatrième et dernière expérimentation (Figure V.19, tag git : `faulty-command-full`) simule la panne du système de reconnaissance automatique de la parole, en supprimant l'icône en bas à gauche de l'image (Figure IV.5b). En revanche, les images sont toujours fournies au système deux secondes avant qu'une action soit attendue. Il semblerait que cette information ait été correctement assimilée par le système lors de son préentrainement, puisque les performances ne sont pas aussi impactées que ce que l'on aurait pu prévoir. Il en résulte tout de même une nette détérioration des performances globales mais qui restent supérieures au hasard (score F1 de l'ordre de 3 % pour le hasard). On note en particulier la grande instabilité qui laisse penser que le système ne peut pas déduire l'intention de l'occupant uniquement à partir des données des capteurs (c'est-à-dire uniquement à partir des informations contextuelles).

Ces constats sur l'utilité présumée de tel ou tel capteur pour la prise de décision nous ont poussées à réaliser une nouvelle expérimentation pour montrer quelles informations sont extraites par le système pour construire les caractéristiques lui permettant de prendre sa décision.

V.2.3 Caractéristiques des états

Au cours de son passage à travers le réseau, l'image d'entrée subit différentes transformations qui permettent de mettre en exergue certaines de ses particularités. Ces caractéristiques sont alors résumées, en sortie de la partie convolutive, en un unique vecteur de grande dimension (Karpathy 2015). Les couches cachées totalement connectées vont ensuite appliquer différentes transformations à l'espace de représentation de manière à rendre les données linéairement séparables (LeCun, Y. Bengio et G. Hinton 2015). La couche de sortie classifie alors les exemples comme pourrait le faire un ensemble de machines à vecteur support. Ainsi, les représentations obtenues à ces deux endroits du réseau semblent particulièrement intéressantes à étudier, permettant de mettre en évidence les caractéristiques extraites et la façon dont le réseau groupe les exemples de manière à les rendre linéairement séparables.

L'étude de vecteurs de grandes tailles (4096 et 512 dimensions dans le cas de la dernière version du système Arcades) n'est pas triviale. Une solution est alors de projeter ces vecteurs dans des espaces de moindre dimension (généralement 2 ou 3) tout en conservant certaines propriétés. Nous allons utiliser ici la projection ***t-SNE*** (Maaten et G. Hinton

2008) qui est l'une des plus utilisées aujourd'hui dans le domaine des réseaux neuronaux et qui permet de projeter des vecteurs dans un espace bi-dimensionnel tout en conservant la distance euclidienne entre deux vecteurs.

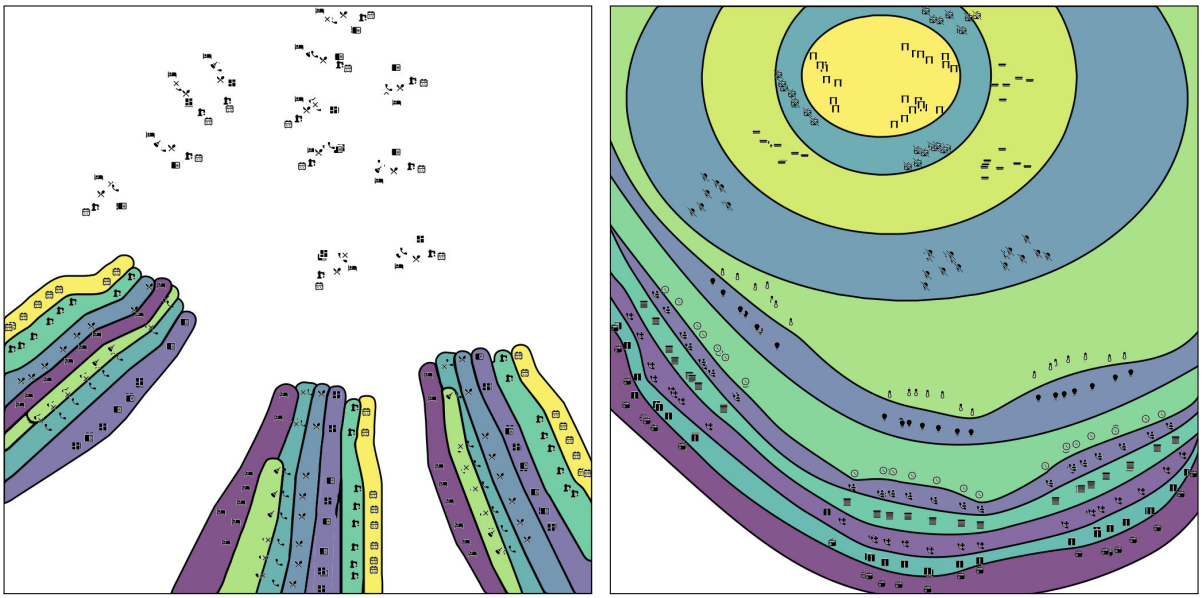
Les expérimentations décrites ci-après suivent toutes le même déroulement. À partir d'un modèle pré-entrainé, nous fournissons un ensemble d'images au système. Pour chaque image, les représentations intermédiaires obtenues en sortie de la partie convolutive et après la dernière couche cachée du réseau sont sauvegardées. Dans le même temps, les informations liées à l'image d'entrée (localisation, activité, commande, action attendue, capteurs activés) sont également sauvegardées. Les deux ensembles de vecteurs sont ensuite chacun projetés dans un espace bi-dimensionnel, permettant d'obtenir des coordonnées x et y. Ces coordonnées permettent de générer une image dans laquelle on présente les informations liées au vecteur (une image par type d'information).

La Figure V.20 présente les résultats de la projection des vecteurs extraits par le réseau obtenu à la fin de l'expérimentation 1, après convolution d'images générées à partir de données synthétiques annotées. On y retrouve les 324 états possibles agencés selon certaines caractéristiques. Ainsi, dans la partie basse de la Figure V.20a on note très nettement que l'activité de l'utilisateur est identifiée, permettant d'aligner les vecteurs représentant les images associées à la même activité. Un agencement en cercles concentriques peut également être discerné sur la Figure V.20b représentant les commandes prononcées par l'utilisateur. Enfin, la localisation de l'utilisateur au moment de l'énonciation de la commande, présentée Figure V.20c (une couleur différente pour chaque pièce), est également très bien identifiée, permettant de regrouper les vecteurs selon cette caractéristique. L'extraction de telles caractéristiques sur un corpus annoté était prévisible puisqu'il s'agit des informations disponibles sur l'image (une icône pour chacune des caractéristiques).

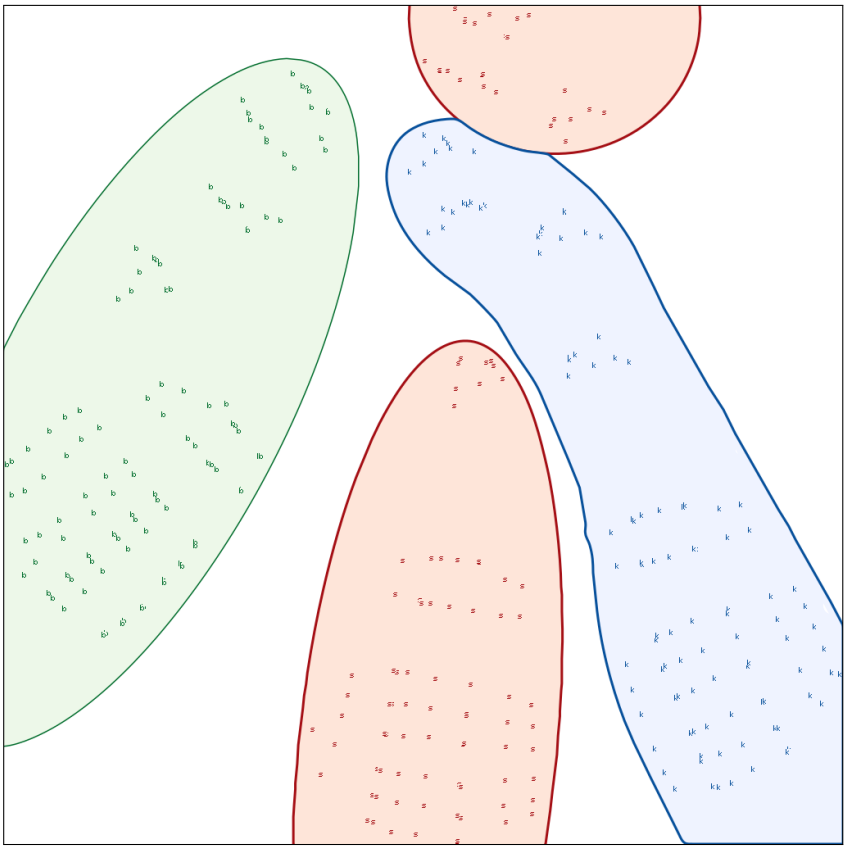
Les mêmes représentations peuvent être générées pour des données issues de données réelles brutes (Figure V.21) traitées par le modèle appris lors de l'expérimentation 2 bis. On constate alors que ces caractéristiques ne sont pas aussi bien identifiées. Si des groupes apparaissent pour la localisation et la commande, l'activité ne semble pas être une caractéristique pertinente pouvant être extraite de l'image.

Ces résultats permettent de confirmer le choix régulièrement fait d'utiliser la localisation de l'utilisateur comme élément du contexte. À l'inverse, l'étiquetage sémantique de l'activité ne semble pas pertinent et des classifications plus générales, comme le niveau d'activité de l'utilisateur, pourraient être plus adaptées.

Une autre projection peut également se faire à partir des vecteurs de la dernière couche cachée du réseau. Il s'agit alors de la dernière représentation intermédiaire avant que le système ne classe la donnée d'entrée. On ne présente ici que les résultats pour les données



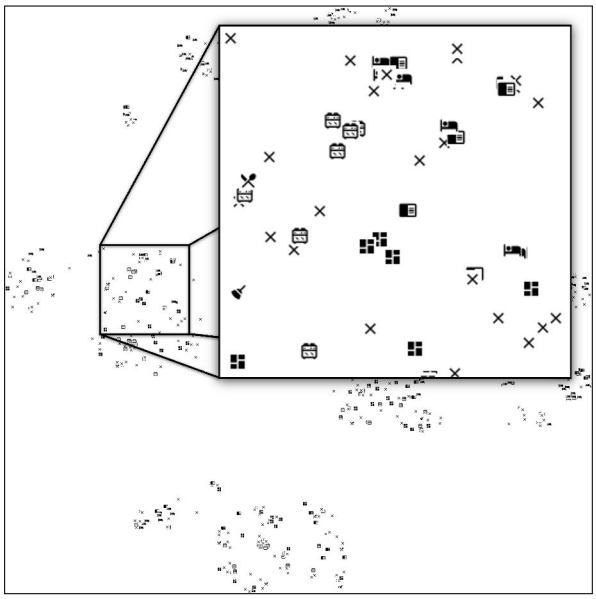
(a) Activité de l'utilisateur (b) Commande prononcée par l'utilisateur



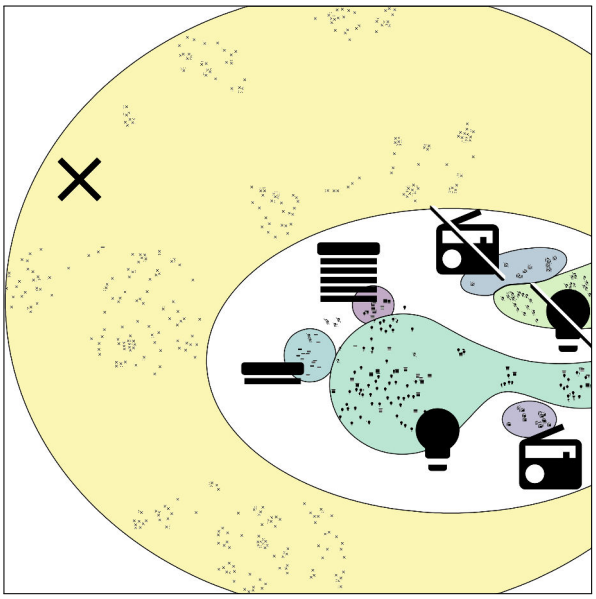
(c) Localisation de l'utilisateur

FIG. V.20 : Projection t-SNE bi-dimensionnelle des vecteurs en sortie de convolution associée à différentes caractéristiques connues des données.

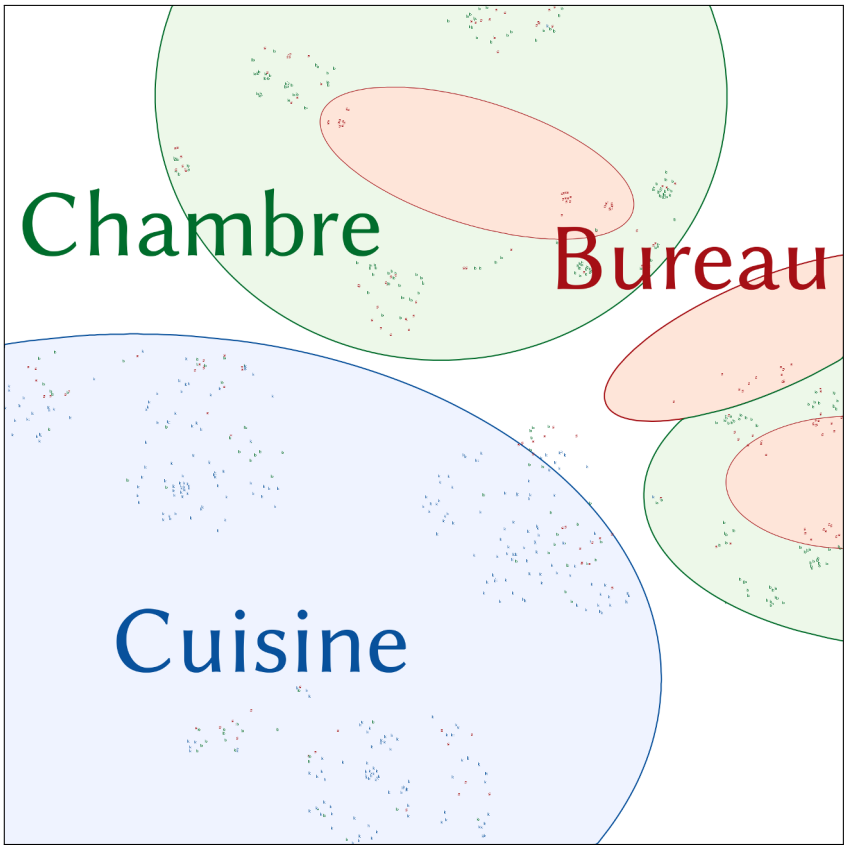
Les vecteurs sont issus du traitement par le réseau d'images générées reprenant les mêmes informations que les annotations (commande, localisation, activité).



(a) Activité de l'utilisateur



(b) Commande prononcée par l'utilisateur



(c) Localisation de l'utilisateur

FIG. V.21 : Projection t-SNE bi-dimensionnelle des vecteurs en sortie de convolution associée à différentes caractéristiques connues des données.

Les vecteurs sont issus du traitement par le réseau d'images générées à partir de données brutes réelles du corpus SWEETHOME.

réelles brutes. Sur la Figure V.22, on associe à chaque projection l'action attendue de la part du système. Malgré la prédominance des actions de type « Ne rien faire », on constate que d'autres groupes sont formés regroupant les actions de même type, et au même endroit (identifié par la couleur de l'icône). Cette projection permet de rendre compte de la capacité d'un réseau neuronal à transformer l'espace des données de manière à former des groupes linéairement séparables permettant d'identifier les différentes classes.

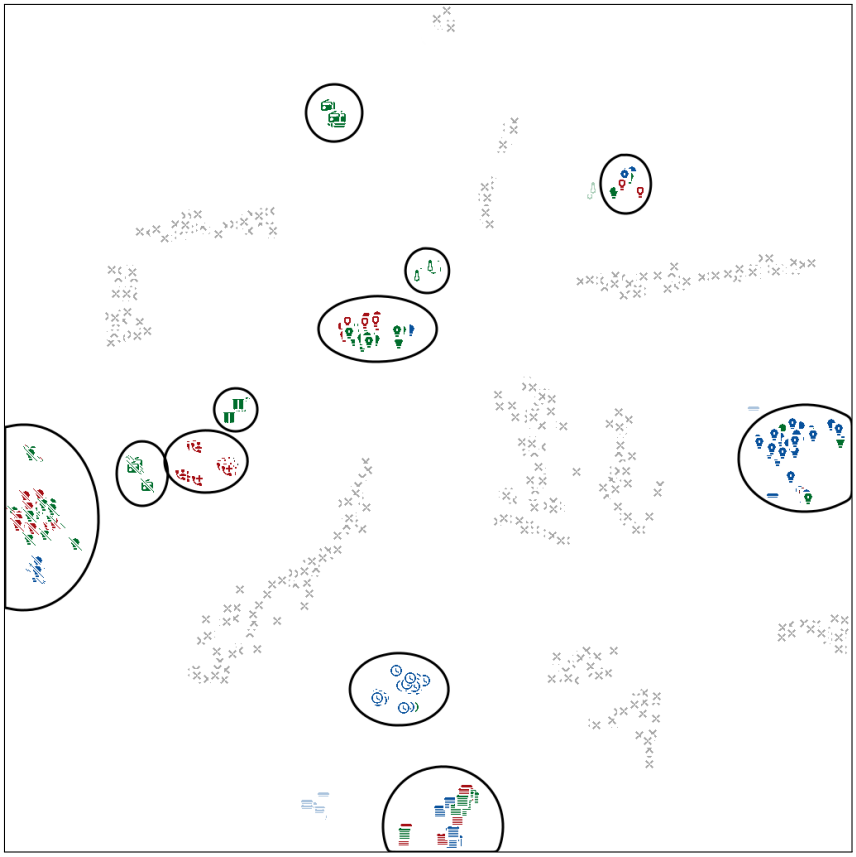


FIG. V.22 : Projection t-SNE bi-dimensionnelle des vecteurs à la dernière couche cachée associée à l'action attendue.
Les vecteurs sont issus du traitement par le réseau d'images générées à partir de données brutes réelles du corpus SWEETHOME.

Chapitre VI

Conclusion

Les souvenirs de nos vies, de nos œuvres
et de nos actes se poursuivront à travers d'autres.

Rosa Parks

Afin d'améliorer l'acceptation de systèmes domotiques il est nécessaire de les rendre plus simples à configurer, utiliser et adapter. Pour apporter une solution à ces problématiques, les habitats intelligents tendent à appliquer les méthodes d'intelligence artificielle pour permettre aux systèmes d'apprendre de leurs interactions avec l'utilisateur. Ces approches soulèvent cependant différentes questions quant à la représentation des données et à la méthode d'adaptation peu intrusive. Dans la Section VI.1 nous résumons les différentes contributions présentées tout au long de ce mémoire, apportant une réponse novatrice à ces questions. La Section VI.2 revient quant à elle sur les limites de nos approches, faisant émerger de nouvelles questions, et propose différentes voies d'améliorations.

VI.1 Contributions

Au cours de ce mémoire de thèse, nous avons présenté une nouvelle approche pour la prise de décision adaptative en habitat intelligent. Après avoir exposé l'état actuel des recherches dans ce domaine, nous avons détaillé le fonctionnement de notre approche reposant sur l'apprentissage par renforcement. Une grande partie de la thèse a été consacrée à la compréhension des mécanismes sous-jacents à une telle méthode d'apprentissage et à son utilisation dans le cadre d'un habitat intelligent. De cette première étape a émergé un premier système de prise de décision (Chapitre III). Ce système a été évalué sur un corpus de données réalistes et a exhibé des performances suffisantes pour donner lieu à

deux publications dans des conférences centrées sur l'intelligence artificielle (Brenon, Portet et Vacher 2016a) et sur les environnements intelligents (Brenon, Portet et Vacher 2016b).

Par la suite, nous avons cherché à améliorer ce système en repoussant les limites qu'il rencontrait. Après la consultation d'une bibliographie liée aux réseaux de neurones et à l'apprentissage profond, nous avons appliqué une nouvelle approche totalement novatrice. Cette extension de notre précédent système fait le choix de ne plus représenter le contexte de manière explicite mais de le laisser émerger des données de manière non-supervisée. Ce choix nous permet également de proposer une nouvelle méthode de représentation des données liées à l'habitat intelligent reposant sur une représentation graphique de l'habitat et des capteurs. Le système Arcades alors développé a été évalué de manière analogue au précédent et a montré des performances bien supérieures et de nombreux avantages. Différentes analyses ont alors permis d'éclairer le fonctionnement de ce système et de fournir des informations quant aux informations pertinentes pour la caractérisation du contexte d'une interaction.

Les contributions liées au système Arcades sont ainsi nombreuses. Dans un premier temps, la totalité du code source du système est disponible en ligne[1], permettant aux futurs chercheurs de l'améliorer ou de l'adapter à de nouveaux problèmes. Dans un second temps, après que notre première expérimentation ait permis de démontrer la pertinence de l'apprentissage par renforcement dans le cadre des habitats intelligents, le système Arcades a montré que l'apprentissage profond par renforcement était également applicable à ce cas d'usage. Par la suite, les dernières analyses ont démontrées que si l'utilisation de la localisation dans la définition du contexte est évidente, la définition de l'activité de l'utilisateur doit pour sa part être approfondie afin de définir une caractéristique réellement discriminante.

VI.2 Limites et perspectives

Si le système Arcades présente des résultats intéressants il ouvre surtout la voie à de nombreuses questions de recherche et évolutions possibles.

Avant tout, la méthode d'interaction entre l'utilisateur et le système n'a pas été traitée dans ces travaux, le signal de renforcement étant généré de manière automatique. Il n'en demeure pas moins que les pistes de réflexions sont nombreuses à ce sujet. D'une part, comme l'avait fait Mozer (2005), en analysant les actions de l'utilisateur sur les interrupteurs suite à une décision. D'autre part, grâce aux microphones, en permettant à

1. https ://brenona.gricad-pages.univ-grenoble-alpes.fr/arcades/

l'utilisateur d'interagir vocalement avec le système et donc de valider ou non une décision à l'aide d'un ensemble de mots clés (« Non », « Très bien », « Encore », etc.).

Ensuite, la quantité de données nécessaire pour obtenir un modèle pertinent est déraisonnable pour une utilisation directe avec un utilisateur naïf. Si la phase de pré-entrainement est une première solution à ce problème, il conviendrait d'en étudier d'autres. L'utilisation d'un système expert, en parallèle du système apprenant, lui permettrait d'être suppléer par une solution éprouvée le temps de l'apprentissage. Les travaux de Pritzel et coll. (2017) visent tout particulièrement le problème de la quantité de données nécessaire en proposant une nouvelle approche, basée uniquement sur l'apprentissage profond par renforcement, nommée *Neural Episodic Control*. De la même manière, Usunier et coll. (2016) proposent une nouvelle stratégie d'exploration qui permet d'obtenir de meilleures performances que la stratégie ϵ-*greedy* habituelle. Le transfert de connaissances d'un modèle vers un autre pourrait également être une autre façon d'améliorer notre approche. En ce sens, des expérimentations au sein d'un autre habitat à partir des modèles présentés ici permettraient de vérifier la faisabilité d'une telle méthode.

Enfin, des améliorations quant aux choix de représentation peuvent aussi être envisagées. Notre système n'utilise aujourd'hui que des images en noir et blanc, et ne considère la présence que d'une seule personne dans l'habitat. L'utilisation de couleurs pourrait permettre la représentation d'informations plus nuancées mais également rendre compte d'évolutions liées à plusieurs utilisateurs évoluant dans un même environnement. La représentation des actions peut également évoluer. En effet chaque décision est ici atomique, comprenant à la fois le type d'action et l'actionneur. La récompense générée ne peut alors pas discriminer l'une ou l'autre des informations. De plus, cette représentation est très liée aux actionneurs disponibles et limite donc l'adaptabilité du système aux évolutions topologiques (nombre ou type d'actionneurs) de l'environnement. L'utilisation d'un réseau à multiples sorties, comme c'est le cas pour différents systèmes reposant sur des réseaux de neurones (Mirowski et coll. 2016 ; Szegedy et coll. 2014), permettrait de décorréler ces deux composantes.

Somme toute, les réseaux de neurones, en apportant un changement de paradigme ont assurément permis d'améliorer les performances de nombreux systèmes de classification ou de contrôle. Leur fonctionnement n'est cependant pas encore totalement compris et chaque avancée en ce sens permet de réaliser de nouvelles prouesses (Huszár 2017 ; Zhang, S. Bengio et coll. 2016 ; Zhang, Q. Liao et coll. 2017 ; Shwartz-Ziv et Tishby 2017 ; Krueger et coll. 2017). Il n'en demeure pas moins que leur réussite est grandement liée aux énormes masses de données annotées disponibles aujourd'hui. Un des défis actuels est donc de rendre leurs performances d'apprentissage raisonnable pour les cas d'usages ne disposant que de peu de données ou demandant une interaction humaine.

Bibliographie

Abowd, Gregory D., Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith et Pete Steggles (1999). « Towards a Better Understanding of Context and Context-Awareness ». Anglais. In : *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. HUC 99. Karlsruhe, Germany : Springer-Verlag, p. 304-307. isbn : 3-540-66550-1. url : http ://dl.acm.org/citation.cfm ?id=647985.743843 (cf. p. 14, 15).

Aloise, Fabio, Piero Aricò, Francesca Schettini, A. Riccio, M. Risetti, Serenella Salinari, Donatella Mattia, Fabio Babiloni et Febo Cincotti (2011). « A new P300 no eye-gaze based interface : Geospell ». Anglais. In : *Proceedings of the International Conference on Bio-Inspired Systems and Signal Processing (BIOSIGNALS 2011)*. Sous la dir. de Fabio Babiloni, Ana L. N. Fred, Joaquim Filipe et Hugo Gamboa. SciTePress, p. 227-232. isbn : 9789898425355. url : http ://dblp.uni-trier.de/db/conf/biostec/biosignals2011.html#AloiseASRRSMBC11 (cf. p. 11).

Araya-López, Mauricio, Vincent Thomas, Olivier Buffet et François Charpillet (oct. 2010). « A Closer Look at MOMDPs ». In : *22nd International Conference on Tools with Artificial Intelligence (ICTAI 2010)*. Proceedings of the 22nd International Conference on Tools with Artificial Intelligence. Arras, France : IEEE. hal : inria-00535559 (cf. p. 24).

Attard, Judie, Simon Scerri, Ismael Rivera et Siegfried Handschuh (2013). « Ontology-based Situation Recognition for Context-aware Systems ». Anglais. In : *Proceedings of the 9th International Conference on Semantic Systems*. I-SEMANTICS 13. New York, NY, USA : ACM, p. 113-120. (Visité le 20/11/2013) (cf. p. 16).

Badii, Atta et Jérome Boudy (2009). « CompanionAble - integrated cognitive assistive & domotic companion robotic systems for ability & security ». Anglais. In : *1er Congrés of the Société Française des Technologies pour l'Autonomie et de Gérontechnologie (SFTAG09)*. Troyes, p. 18-20 (cf. p. 2).

Begue, Alisson (2013). « Connaissance des règles d'oubli de la pillule chez l'adolescente ». Mém. de mast. Faculté de médecine et de maïeutique Lyon Sud Charles Mérieux.

Bellman, Richard (1957). *Dynamic Programming*. Anglais. 1re éd. Princeton, NJ, USA : Princeton University Press. isbn : 9780691146683 (cf. p. 23).

Bengio, Yoshua (2009). « Learning Deep Architectures for AI ». Anglais. In : *Foundations and Trends® in Machine Learning* 2.1, p. 1-127. doi : 10.1561/2200000006 (cf. p. 50).

Berlink, Heider et Anna Helena Reali Costa (2015). « Batch Reinforcement Learning for Smart Home Energy Management ». Anglais. In : *Proceedings of the 24th International Conference on Artificial Intelligence*. IJCAI'15. Buenos Aires, Argentina, p. 2561-2567 (cf. p. 17).

Bleik, Said et Shaheen Gauher (2016). *Computing Classification Evaluation Metrics in R*. Anglais. url : http ://blog.revolutionanalytics.com/2016/03/com_class_eval_metrics_r.html (cf. p. 66).

Bonarini, Andrea et Vittorio Maniezzo (sept. 1991). « Integrating expert systems and decision-support systems : principles and practice. » Anglais. In : *Knowledge-Based Systems* 4.3, p. 172-176. doi : 10.1016/0950-7051(91)90006-N. url : http ://dblp.uni-trier.de/db/journals/kbs/kbs4.html#BonariniM91 (cf. p. 19).

Brenon, Alexis, François Portet et Michel Vacher (juin 2016a). « Étude préliminaire d'une méthode de prise de décision adaptative pour la commande vocale dans un habitat intelligent ». In : *RFIA-RJCIA*. Clermont Ferrand, France, p. 2-8. hal : hal-01326986 (cf. p. 37, 92).

— (sept. 2016b). « Preliminary Study of Adaptive Decision-Making System for Vocal Command in Smart Home ». Anglais. In : *12th International Conference on Intelligent Environments*. London, United Kingdom, p. 218-221. hal : hal-01345333 (cf. p. 37, 92).

Brown, P. J., J. D. Bovey et Xian Chen (oct. 1997). « Context-aware applications : from the laboratory to the marketplace ». Anglais. In : *IEEE Personal Communications* 4.5, p. 58-64. ISSN : 1070-9916. DOI : 10.1109/98.626984 (cf. p. 13).

Cassandra, Anthony R. (2016). *POMDP for Dummies*. Anglais. Consulted on 2016-05-17. URL : http ://pomdp.org/ (cf. p. 24).

Catarci, Tiziana, Febo Cincotti, Massimiliano De Leoni, Massimo Mecella et Giuseppe Santucci (jan. 2009). « Smart homes for all : Collaborating services in a for-all architecture for domotics ». Anglais. In : *Collaborative Computing : Networking, Applications and Worksharing*. Sous la dir. d'Elisa Bertino et James B. D. Joshi. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 10. Springer Berlin Heidelberg, p. 56-69. ISBN : 978-3-642-03353-7. DOI : 10.1007/978-3-642-03354-4_6. (Visité le 21/11/2012) (cf. p. 11).

Centre National de Ressources Textuelles et Lexicales (CNRTL) (2016). *Lexicographie*. URL : http ://www.cnrtl.fr/definition/ (visité le 23/01/2016) (cf. p. 13).

Chahuara, Pedro (mar. 2013). « Contrôle intelligent de la domotique à partir d'informations temporelles multisources imprécises et incertaines ». Thèse de doct. École Doctorale MSTII, Université de Grenoble. HAL : tel-00956372 (cf. p. 16, 19, 20).

Chahuara, Pedro, Anthony Fleury, François Portet et Michel Vacher (juil. 2016). « On-line Human Activity Recognition from Audio and Home Automation Sensors : comparison of sequential and non-sequential models in realistic Smart Homes ». Anglais. In : *Journal of ambient intelligence and smart environments* 8.4, p. 399-422. ISSN : 1876-1372. DOI : 10.3233/AIS-160386 (cf. p. 40).

Chahuara, Pedro, François Portet et Michel Vacher (2017). « Context-aware decision making under uncertainty for voice-based control of smart home ». Anglais. In : *Expert Systems with Applications* 75, p. 63-79 (cf. p. 16-18, 20, 63, 67, 80).

Charalampos, Doukas et Ilias Maglogiannis (2008). « Enabling human status awareness in assistive environments based on advanced sound and motion data classification ». Anglais. In : *Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments*, p. 1-8 (cf. p. 2).

Chen, Chao et Diane J. Cook (2012). « Behavior-Based Home Energy Prediction. » Anglais. In : *Intelligent Environments*. IEEE, p. 57-63. ISBN : 978-1-4673-2093-1. DOI : 10.1109/IE.2012.44. URL : http://dblp.uni-trier.de/db/conf/intenv/intenv2012.html#ChenC12 (cf. p. 12).

Chen, Luke, Chris D. Nugent, Maurice Mulvenna, Dewar Finlay, Xin Hong et Michael Poland (déc. 2008). « A Logical Framework for Behaviour Reasoning and Assistance in a Smart Home ». Anglais. In : *International Journal of Assistive Robotics and Mechatronics* 9.4, p. 20-34 (cf. p. 15).

Chikhaoui, Belkacem, Yazid Benazzouz et Bessam Abdulrazak (2009). « Towards a Universal Ontology for Smart Environments ». Anglais. In : *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services*. Kuala Lumpur, Malaysia : ACM, p. 80-87. ISBN : 978-1-60558-660-1. DOI : 10.1145/1806338.1806361 (cf. p. 16).

Christensen, Heidi, Iñigo Casanueva, Stuart Cunningham, Phil Green et Thomas Hain (2013). « homeService : Voice-enabled assistive technology in the home using cloud-based automatic speech recognition ». Anglais. In : *4th Workshop on Speech and Language Processing for Assistive Technologies*, p. 29-34 (cf. p. 2).

Cook, Diane J., Aaron S. Crandall, L. Thomas Brian et C. Krishnan Narayanan (2013). « CASAS : A Smart Home in a Box ». Anglais. In : *IEEE Computer* 46.6, p. 26-33 (cf. p. 12).

Cook, Diane J. et Sajal K. Das (2007). « How smart are our environments ? An updated look at the state of the art ». Anglais. In : *Pervasive and Mobile Computing*. 3^{e} sér. 3.2, p. 53-73 (cf. p. 12).

Coutaz, Joëlle et James L. Crowley (2008). *Plan « Intelligence Ambiante » : Défis et Opportunités*. Document de réflexion conjoint du comité d'experts "Informatique Ambiante" du département ST2I du CNRS et du Groupe de Travail "Intelligence Ambiante" du Groupe de Concertation Sectoriel (GCS3) du Ministère de l'Enseignement Supérieur et de la Recherche, DGRI A3 (cf. p. 8).

Cristoforetti, L., M. Ravanelli, M. Omologo, A. Sosi, A. Abad, M. Hagmueller et P. Maragos (2014). « The DIRHA simulated corpus ».

Anglais. In : *The 9th edition of the Language Resources and Evaluation Conference (LREC)*. Reykjavik, Iceland, p. 2629-2634 (cf. p. 2).

Cybenko, George (déc. 1989). « Approximation by superpositions of a sigmoidal function ». Anglais. In : *Mathematics of Control, Signals, and Systems* 2.4, p. 303-314. doi : 10.1007/bf02551274 (cf. p. 47, 51).

De Carolis, Berardina et Giovanni Cozzolongo (déc. 2004). « C@sa : Intelligent Home Control and Simulation ». Anglais. In : *International Conference on Computational Intelligence*. Sous la dir. d'Ali Okatan. International Conference on Computational Intelligence, ICCI 2004, December 17-19, 2004, Istanbul, Turkey, Proceedings. Istanbul, Turkey : International Computational Intelligence Society, p. 462-465. isbn : 975-98458-1-4 (cf. p. 20).

Dey, Anind K., Daniel Salber, Gregory D. Abowd et Masayasu Futakawa (1999). « The Conference Assistant : Combining Context-Awareness with Wearable Computing ». Anglais. In : *Proceedings of the 3rd IEEE International Symposium on Wearable Computers*. ISWC 99. Washington, DC, USA : IEEE Computer Society, p. 21-. isbn : 0-7695-0428-0. url : http ://dl.acm.org/citation.cfm ?id=519309.856496 (cf. p. 15).

Dreyfus, Stuart (août 1962). « The numerical solution of variational problems ». Anglais. In : *Journal of Mathematical Analysis and Applications* 5 (1). Sous la dir. d'Elsevier, p. 30-45. doi : 10.1016/0022-247X(62)90004-5 (cf. p. 48).

Ducatel, Ken, Marc Bogdanowicz, Fabiana Scapolo, Jos Leijten et Jean-Claude Burgelman (2001). *Scenarios for ambient intelligence in 2010*. Anglais. Office for official publications of the European Communities (cf. p. 1, 9).

Dumoulin, Vincent (2016). *A guide to convolution arithmetic for deep learning*. url : https ://github.com/vdumoulin/conv_arithmetic (visité le 12/09/2017) (cf. p. 121, 122).

Eldan, Ronen et Ohad Shamir (12 déc. 2015). « The Power of Depth for Feedforward Neural Networks ». Anglais. In : arXiv : 1512.03965v4 [cs.LG] (cf. p. 47).

Filho, G. et Tom J. Moir (2010). « From science fiction to science fact : a smart-house interface using speech technology and a photorealistic avatar ». Anglais. In : *International Journal of Computer Applications in Technology* 39.8, p. 32-39 (cf. p. 2).

Fukushima, Kunihiko (1980). « Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position ». Anglais. In : *Biological Cybernetics* 36.4, p. 193-202. ISSN : 1432-0770. DOI : 10.1007/BF00344251 (cf. p. 48).

Gallissot, Mathieu (avr. 2012). « Modéliser le concept du confort dans l'habitat Intelligent : du multisensoriel au comportement ». Thèse de doct. École Doctorale MSTII, Université de Grenoble. HAL : tel-00738342 (cf. p. 8, 9).

Gallissot, Mathieu, Jean Caelen, Francis Jambon et Brigitte Meillon (2013). « Une plateforme usage pour l'intégration de l'informatique ambiante dans l'habitat. L'appartement Domus ». In : *Technique et Science Informatiques (TSI)* 32.5, p. 547-574 (cf. p. 13, 38).

Gemmeke, Jort F., Bart Ons, Netsanet Tessema, Hugo Van Hamme, Janneke Van De Loo, Guy De Pauw, Walter Daelemans, Jonathan Huyghe, Jan Derboven, Lode Vuegen, Bert Van Den Broeck, Peter Karsmakers et Bart Vanrumste (2013). « Self-taught assistive vocal interfaces : an overview of the ALADIN project ». Anglais. In : *Interspeech 2013*, p. 2039-2043 (cf. p. 2).

Golge, Eren (fév. 2015). *Comparison : SGD vs Momentum vs RMSprop vs Momentum+RMSprop vs AdaGrad*. Anglais. URL : http ://www.erogol.com/comparison-sgd-vs-momentum-vs-rmsprop-vs-momentumrmsprop/ (cf. p. 53).

Gómez-Romero, Juan, Miguel A. Serrano, Miguel A. Patricio, Jesús García et José M. Molina (oct. 2012). « Context-based scene recognition from visual data in smart homes : an Information Fusion approach ». Anglais. In : *Personal and Ubiquitous Computing* 16.7, p. 835-857 (cf. p. 18, 19).

Hamill, Melinda, Vicky Young, Jennifer Boger et Alex Mihailidis (2009). « Development of an automated speech recognition interface for Personal Emergency Response Systems ». Anglais. In : *Journal of NeuroEngineering and Rehabilitation* 6 (cf. p. 2).

Håstad, Johan (1986). « Almost optimal lower bounds for small depth circuits ». Anglais. In : *Proceedings of the eighteenth annual ACM symposium on Theory of computing (STOC 1986)*. Association for Computing Machinery (ACM). DOI : 10.1145/12130.12132 (cf. p. 47).

Helaoui, Rim, Daniele Riboni et Heiner Stuckenschmidt (2013). « A Probabilistic Ontological Framework for the Recognition of Multilevel Human Activities ». Anglais. In : *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp 13. Zurich, Switzerland : ACM, p. 345-354. isbn : 978-1-4503-1770-2. doi : 10.1145/2493432.2493501 (cf. p. 16, 18).

Hinton, Geoffrey E. (fév. 2015). *Lecture 6e – rmsprop : Divide the gradient by a running average of its recent magnitude*. Anglais. url : http ://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (cf. p. 53).

Hinton, Geoffrey E. et Ruslan R. Salakhutdinov (juil. 2006). « Reducing the Dimensionality of Data with Neural Networks ». Anglais. In : *Science* 313.5786, p. 504-507. doi : 10.1126/science.1127647 (cf. p. 50).

Hinton, Geoffrey E., Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever et Ruslan R. Salakhutdinov (3 juil. 2012). « Improving neural networks by preventing co-adaptation of feature detectors ». Anglais. In : *CoRR*. arXiv : 1207.0580v1 [cs.NE] (cf. p. 122).

Holmes, Arran, Hakan Duman et Anthony Pounds-Cornish (20 fév. 2002). « The iDorm : Gateway to Heterogeneous Networking Environments ». Anglais. In : *Proceedings of the International ITEA Workshop on Virtual Home Environments*. Paderborn, Germany : ITEA Press, p. 20-37 (cf. p. 11).

Howard, Ronald A. et James E. Matheson (sept. 1981). « Influence Diagrams ». Anglais. In : *Readings on The Principles and Applications of Decision Analysis*. Sous la dir. de R.A. Howard et J.E. Matheson. T. 2. 3. Institute for Operations Research et the Management Sciences (INFORMS), p. 127-143. doi : 10.1287/deca.1050.0020 (cf. p. 19).

Huszár, Ferenc (2017). *Everything that Works Works Because itś Bayesian : Why Deep Nets Generalize ?* url : http ://www.inference.vc/everything-that-works-works-because-its-bayesian-2/ (visité le 19/09/2017) (cf. p. 93).

Intille, Stephen S. (avr. 2002). « Designing a home of the future ». Anglais. In : *IEEE Pervasive Computing* 1.2, p. 76-82. doi : 10.1109/mprv.2002.1012340 (cf. p. 10).

Ioffe, Sergey et Christian Szegedy (11 fév. 2015). « Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift ». Anglais. In : arXiv : 1502.03167v3 [cs.LG] (cf. p. 122).

Kaliouby, Rana El, Peter Robinson, Simeon Keates et Engineering Design Centre (2003). « Temporal Context and the Recognition of Emotion from Facial Expression ». Anglais. In : *Proceedings of the HCI International Conference*. American Psychological Association, p. 631-635 (cf. p. 13).

Karami, Abir-Beatrice, Anthony Fleury, Jacques Boonaert et Stéphane Lecoeuche (juin 2016). « User in the Loop : Adaptive Smart Homes Exploiting User Feedback — State of the Art and Future Directions ». Anglais. In : *Information* 7, p. 35. doi : 10.3390/info7020035 (cf. p. 18, 21).

Karpathy, Andrej (jan. 2015). *CS231n : Convolutional Neural Networks for Visual Recognition*. Anglais. Standford. url : https ://cs231n.github.io/ (cf. p. 49, 81, 85).

Katzouris, Nikos, Alexander Artikis et Georgios Paliouras (2014). « Event Recognition for Unobtrusive Assisted Living ». Anglais. In : *8th Hellenic Conference on AI*. Ioannina, Greece : Springer-Verlag, p. 475-488. doi : 10.1007/978-3-319-07064-3_41 (cf. p. 15, 18).

Kidd, Cory D., Robert Orr, Gregory D. Abowd, Christopher G. Atkeson, Irfan A. Essa, Blair MacIntyre, Elizabeth Mynatt, Thad E. Starner et Wendy Newstetter (1999). « The Aware Home : A Living Laboratory for Ubiquitous Computing Research ». Anglais. In : *Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*. Springer-Verlag, p. 191-198. doi : 10.1007/10705432_17 (cf. p. 10).

Klein, Dan et Pieter Abbeel (2013). *UC Berkeley CS188 Intro to AI*. Anglais. url : http ://ai.berkeley.edu/ (visité le 15/02/2017) (cf. p. 31).

Klein, Michael, Andreas Schmidt et Rolf Lauer (2007). « Ontology-Centred Design of an Ambient Middleware for Assisted Living : The Case of SOPRANO ». Anglais. In : *30th Annual German Conference on Artificial Intelligence (KI 2007)* (cf. p. 11).

KNUDSEN, Harold R. et Louis H. MUZEKARI (1983). « The effects of verbal statements of context on facial expressions of emotion ». Anglais. In : *Journal of Nonverbal Behavior* 7.4, p. 202-212. ISSN : 1573-3653. DOI : 10.1007/BF00986266 (cf. p. 13).

KOFLER, Mario J., Christian REINISCH et Wolfgang KASTNER (avr. 2012). « A semantic representation of energy-related information in future smart homes ». Anglais. In : *Energy and Buildings* 47, p. 169-179 (cf. p. 18, 19).

KRIZHEVSKY, Alex, Ilya SUTSKEVER et Geoffrey E. HINTON (2012). « ImageNet Classification with Deep Convolutional Neural Networks ». Anglais. In : *Advances in Neural Information Processing Systems 25*. Sous la dir. de F. PEREIRA, C. J. C. BURGES, L. BOTTOU et K. Q. WEINBERGER. Curran Associates, Inc., p. 1097-1105. URL : http ://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf (cf. p. 49, 120).

KRUEGER, David, Nicolas BALLAS, Stanislaw JASTRZEBSKI, Devansh ARPIT, Maxinder S. KANWAL, Tegan MAHARAJ, Emmanuel BENGIO, Asja FISCHER et Aaron COURVILLE (17 fév. 2017). « Deep Nets Don't Learn via Memorization ». In : *5th International Conference on Learning Representations (ICLR 2017)*. OpenReview : rJv6ZgHYg (cf. p. 93).

KUSIAK, Andrew (2007). « Innovation : The living laboratory perspective ». Anglais. In : *Computer-Aided Design and Applications* 4.6, p. 863-876. DOI : 10.1080/16864360.2007.10738518 (cf. p. 10).

LAGO, Paula, Frederic LANG, Claudia RONCANCIO, Claudia JIMÉNEZ-GUARÍN, Radu MATEESCU et Nicolas BONNEFOND (juin 2017). « The ContextAct@A4H real-life dataset of daily-living activities Activity recognition using model checking ». Anglais. In : *CONTEXT 2017*. T. 10257. Lecture Notes in Computer Science. Paris, France : Springer Verlag, p. 175-188. DOI : 10.1007/978-3-319-57837-8_14. HAL : hal-01551418 (cf. p. 12).

LEBELLEGO, G., N. NOURY, G. VIRONE, M. MOUSSEAU et J. DEMONGEOT (jan. 2006). « A model for the measurement of patient activity in a hospital suite ». Anglais. In : *IEEE Transactions on Information Technology in Biomedicine* 10.1, p. 92-99. DOI : 10.1109/titb.2005.856855 (cf. p. 13).

Lecouteux, Benjamin, Michel Vacher et François Portet (août 2011). « Distant Speech Recognition in a Smart Home : Comparison of Several Multisource ASRs in Realistic Conditions ». Anglais. In : *Interspeech 2011*. Florence, Italy, p. 2273-2276 (cf. p. 2).

LeCun, Yann (1985). « Une procédure d'apprentissage pour réseau à seuil asymmétrique (A learning scheme for asymmetric threshold networks) ». In : *Proceedings of Cognitiva 85, Paris, France*, p. 599-604 (cf. p. 48).

LeCun, Yann, Yoshua Bengio et Geoffrey Hinton (mai 2015). « Deep learning ». Anglais. In : *Nature* 521, p. 436-444 (cf. p. 48, 85).

LeCun, Yann, L. Bottou, Yoshua Bengio et P. Haffner (nov. 1998). « Gradient-based learning applied to document recognition ». Anglais. In : *Proceedings of the IEEE* 86.11, p. 2278-2324. ISSN : 0018-9219. DOI : 10.1109/5.726791 (cf. p. 49).

Lee, Seung-Hyun et Sung-Bae Cho (2012). « Fusion of Modular Bayesian Networks for Context-Aware Decision Making ». Anglais. In : *Hybrid Artificial Intelligent Systems*. Sous la dir. d'Emilio Corchado, Václav Snášel, Ajith Abraham, Michal Wozniak, Manuel Graña et Sung-Bae Cho. T. 7208. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, p. 375-384. ISBN : 978-3-642-28941-5. (Visité le 17/10/2012) (cf. p. 19).

Leong, Chui Yew, A. R. Ramli et T. Perumal (2009). « A rule-based framework for heterogeneous subsystems management in smart home environment ». Anglais. In : *IEEE Transactions on Consumer Electronics* 55.3, p. 1208-1213 (cf. p. 19).

Li, D. et S. K. Jayaweera (2014). « Reinforcement learning aided smart-home decision-making in an interactive smart grid ». Anglais. In : *2014 IEEE Green Energy and Systems Conference (IGESC)*, p. 1-6 (cf. p. 17).

Liao, Hsien-Chou et Chien-Chih Tu (2007). « A RDF and OWL-Based Temporal Context Reasoning Model for Smart Home ». Anglais. In : *Information Technology Journal* 6.8, p. 1130-1138. ISSN : 18125638 (cf. p. 16, 18).

Lin, L.-J. (1993). *Reinforcement Learning for robots using neural Networks*. Rapp. tech. DTIC Document (cf. p. 52).

Lipton, Zachary C., John Berkowitz et Charles Elkan (29 mai 2015). « A Critical Review of Recurrent Neural Networks for Sequence Learning ». In : arXiv : 1506.00019v4 [cs.LG] (cf. p. 69).

Maaten, Laurens J.P. van der et Geoffrey Hinton (nov. 2008). « Visualizing High-Dimensional Data Using t-SNE ». Anglais. In : *Journal of Machine Learning Research* (9), p. 2579-2605. URL : https ://lvdmaaten.github.io/tsne/ (cf. p. 85).

McCulloch, Warren S et Walter Pitts (1943). « A logical calculus of the ideas immanent in nervous activity ». Anglais. In : *The bulletin of mathematical biophysics* 5.4, p. 115-133 (cf. p. 46, 119).

Mileo, Alessandra, Davide Merico et Roberto Bisiani (2011). « Reasoning support for risk prediction and prevention in independent living ». Anglais. In : *Theory Pract. Log. Program.* 11.2-3, p. 361-395. ISSN : 1471-0684. DOI : 10.1017/S147106841000058X (cf. p. 15, 18).

Milion, Nathalie, Philippe Dard, Mireille Jandon, Olivier Leberre et Philippe Deliot-Lefèvre (2005). « Maison intelligente, que fait-elle pour nous ? » In : *Sciences* (4) (cf. p. 7).

Minsky, Marvin et Seymour Papert (1969). *Perceptrons : An Introduction to Computational Geometry*. Anglais. MIT Press. ISBN : 978-0262130431 (cf. p. 47).

Mirowski, Piotr, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J. Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran et Raia Hadsell (11 nov. 2016). « Learning to Navigate in Complex Environments ». Anglais. In : *CoRR*. arXiv : 1611.03673v3 [cs.AI] (cf. p. 93).

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski et coll. (2015). « Human-level control through deep reinforcement learning ». Anglais. In : *Nature* 518.7540, p. 529-533 (cf. p. 18, 21, 26, 45, 51-53, 57, 58, 60, 64, 73).

Moore, Philip, Bin Hu et Mike Jackson (2011). « Rule Strategies for Intelligent Context-Aware Systems : The Application of Conditional Relationships in

Decision-Support ». Anglais. In : *International Conference on Complex, Intelligent and Software Intensive Systems, CISIS 2011*. Seoul, Korea, p. 9-16. DOI : 10.1109/CISIS.2011.12 (cf. p. 19).

MORGENSTERN, Oskar et John von NEUMANN (1944). *Theory of Games and Economic Behavior*. Anglais. Sous la dir. de Princeton University PRESS (cf. p. 17).

MOZER, Michael C. (1998). « The neural network house : An environment that adapts to its inhabitants ». Anglais. In : *Proceedings of the American Association for Artificial Intelligence* (cf. p. 10, 16, 21).

— (2005). « Smart Environments : Technology, Protocols and Applications ». Anglais. In : *Smart Environments*. Sous la dir. de D. Cook & R. Das (EDS.) John Wiley & Sons, Inc. Chap. 12 - Lessons from an Adaptive Home, p. 271-294. ISBN : 9780471686590. DOI : 10.1002/047168659X.ch12 (cf. p. 16-18, 26, 28, 92).

NISHIYAMA, Takashi, Shinpei HIBIYA et Tetsuo SAWARAGI (2011). « Development of agent system based on decision model for creating an ambient space ». Anglais. In : *AI & Society* 26.3, p. 247-259. ISSN : 0951-5666. DOI : 10.1007/s00146-010-0305-3. URL : http ://www.springerlink.com/content/j64226q53803r30l/abstract/ (cf. p. 18, 20).

NORMAN, Donald A. (11 sept. 1998). *The Invisible Computer : Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution*. Anglais. MIT PR. 320 p. ISBN : 0262140659. URL : http ://www.ebook.de/de/product/3638741/donald_a_norman_the_invisible_computer_why_good_products_can_fail_the_personal_computer_is_so_complex_and_information_appliances_are_the_solution.html (cf. p. 10).

PASCOE, Jason (1998). « Adding Generic Contextual Capabilities to Wearable Computers ». Anglais. In : *Proceedings of the 2Nd IEEE International Symposium on Wearable Computers*. ISWC '98. Washington, DC, USA : IEEE Computer Society, p. 92-. ISBN : 0-8186-9074-7 (cf. p. 14).

PEETOOM, Kirsten K. B., Monique A. S. LEXIS, Manuela JOORE, Carmen D. DIRKSEN et Luc P. DE WITTE (2014). « Literature review on monitoring technologies and their outcomes in independently living elderly people ». Anglais. In : *Disability and*

Rehabilitation : Assistive Technology, p. 1-24. DOI : 10.3109/17483107.2014.961179 (cf. p. 8).

POPESCU, Mihail, Yun LI, Marjorie SKUBIC et Marilyn RANTZ (20-25 août 2008). « An acoustic fall detector system that uses sound height information to reduce the false alarm rate ». Anglais. In : *Proceeding of the 30th Annual International Conference of the IEEE (EMBS 2008)*, p. 4628-4631 (cf. p. 2).

PORTET, François, Michel VACHER, Caroline GOLANSKI, Camille ROUX et Brigitte MEILLON (2013). « Design and evaluation of a smart home voice interface for the elderly : acceptability and objection aspects ». Anglais. In : *Personal and Ubiquitous Computing* 17 (1), p. 127-144. ISSN : 1617-4909. DOI : 10.1007/s00779-011-0470-5. HAL : hal-00953242v1 (cf. p. 2).

PRITZEL, Alexander, Benigno URIA, Sriram SRINIVASAN, Adrià PUIGDOMÈNECH, Oriol VINYALS, Demis HASSABIS, Daan WIERSTRA et Charles BLUNDELL (2017). « Neural Episodic Control ». Anglais. In : *CoRR*. arXiv : 1703.01988v1 [cs.LG] (cf. p. 71, 93).

RAFFEL, Colin (fév. 2016). *Stochastic Optimization Techniques*. Anglais. URL : http://colinraffel.com/wiki/stochastic_optimization_techniques#rmsprop (cf. p. 53).

RASHIDI, Parisa, Diane J. COOK, Lawrence B. HOLDER et Maureen SCHMITTER–EDGECOMBE (2011). « Discovering Activities to Recognize and Track in a Smart Environment ». Anglais. In : *IEEE Trans. Knowl. Data Eng.* 23.4, p. 527-539. DOI : 10.1109/TKDE.2010.148 (cf. p. 12).

RICHARDSON, Matthew et Pedro DOMINGOS (2006). « Markov logic networks ». Anglais. In : *Machine Learning* 62.1-2, p. 107-136 (cf. p. 16, 20).

RODRÍGUEZ, Natalia Daz, Manuel P. CUÉLLAR, Johan LILIUS et Miguel Delgado CALVO-FLORES (2014a). « A fuzzy ontology for semantic modelling and recognition of human behaviour ». Anglais. In : *Knowledge-Based Systems* 66, p. 46-60. DOI : 10.1016/j.knosys.2014.04.016 (cf. p. 16, 18).

— (mar. 2014b). « A Survey on Ontologies for Human Behavior Recognition ». Anglais. In : *ACM Computing Surveys* 46.4 (43), p. 1-33. ISSN : 0360-0300. DOI : 10.1145/2523819 (cf. p. 15).

ROSENBLATT, Frank (nov. 1958). « The perceptron : a probabilistic model for information storage and organization in the brain ». Anglais. In : *Psychological Review* 65.6, p. 386-408 (cf. p. 46).

RUDER, Sebastian (jan. 2016). *An overview of gradient descent optimization algorithms*. Anglais. URL : http ://sebastianruder.com/optimizing-gradient-descent/index.html (cf. p. 53).

RUMELHART, David E., James L. MCCLELLAND et THE P. D. P. RESEARCH GROUP (1er jan. 1986). *Parallel Distributed Processing*. Anglais. MIT Press Ltd. 567 p. ISBN : 026268053X. URL : http ://www.ebook.de/de/product/9337112/david_e_rumelhart_james_l_mcclelland_the_pdp_research_group_parallel_distributed_processing.html (cf. p. 48).

RUSSEL, Stuart et Peter NORVIG (2009). *Artificial Intelligence : A Modern Approach*. Anglais. Sous la dir. de PEARSON. third. Pearson (cf. p. 17, 21, 24, 29).

SAKAMURA, Ken, éd. (1990). *TRON Project 1990*. Anglais. Springer Nature. DOI : 10.1007/978-4-431-68129-8 (cf. p. 10).

SCHAUL, Tom, Justin BAYER, Daan WIERSTRA, Yi SUN, Martin FELDER, Frank SEHNKE, Thomas RÜCKSTIESS et Jürgen SCHMIDHUBER (2010). « PyBrain ». Anglais. In : *Journal of Machine Learning Research*. URL : http ://pybrain.org/ (cf. p. 33).

SCHERER, Dominik, Andreas MÜLLER et Sven BEHNKE (15-18 sept. 2010). « Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition ». In : *20th International Conference Artificial Neural Networks (ICANN 2010)*. (Thessaloniki, Greece). Sous la dir. de Konstantinos DIAMANTARAS, Wlodek DUCH et Lazaros S. ILIADIS. T. 3. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 92-101 (cf. p. 122).

SCHILIT, Bill N., Norman ADAMS et Roy WANT (déc. 1994). « Context-Aware Computing Applications ». Anglais. In : *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*. IEEE Computer Society, p. 85-90. DOI : 10.1109/WMCSA.1994.16 (cf. p. 14).

Settles, Burr (2012). « Active learning ». Anglais. In : *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6.1, p. 1-114 (cf. p. 25, 28).

Shortliffe, Edward H. et Bruce G. Buchanan (1975). « A model of inexact reasoning in medicine ». Anglais. In : *Mathematical Biosciences* 23.3, p. 351-379. ISSN : 0025-5564. DOI : 10.1016/0025-5564(75)90047-4. URL : http://www.sciencedirect.com/science/article/pii/0025556475900474 (cf. p. 19).

Shwartz-Ziv, Ravid et Naftali Tishby (2 mar. 2017). « Opening the Black Box of Deep Neural Networks via Information ». In : *Computer Research Repository (CoRR)*. arXiv : 1703.00810v3 [cs.LG] (cf. p. 93).

Sondik, Edward J. (1971). « The optimal control of partially observable Markov processes ». Anglais. Thèse de doct. Stanford University (cf. p. 24).

Sutton, Richard S. et Andrew G. Barto (19 juin 2017). *Reinforcement Learning : An Introduction*. Anglais. Sous la dir. de The M. I. T. Press. 2e éd. URL : http://incompleteideas.net/sutton/book/the-book-2nd.html (visité le 18/09/2017). En prép. (cf. p. 23, 24, 29, 31, 33).

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke et Andrew Rabinovich (17 sept. 2014). « Going Deeper with Convolutions ». In : *Computer Vision and Pattern Recognition*. arXiv : 1409.4842v1 [cs.CV] (cf. p. 93).

Tsitsiklis, J. et B. V. Roy (1997). « An Analysis if Temporal-Difference learning with function approximation ». In : *IEEE Trans. Automat. Contr.* P. 674-690 (cf. p. 51).

Usunier, Nicolas, Gabriel Synnaeve, Zeming Lin et Soumith Chintala (2016). « Episodic Exploration for Deep Deterministic Policies : An Application to StarCraft Micromanagement Tasks ». Anglais. In : arXiv : 1609.02993v3 [cs.AI] (cf. p. 93).

Vacher, Michel, Sybille Caffiau, François Portet, Brigitte Meillon, Camille Roux, Elena Elias, Benjamin Lecouteux et Pedro Chahuara (mai 2015). « Evaluation of a Context-Aware Voice Interface for Ambient Assisted Living ». Anglais. In : *ACM Transactions on Accessible Computing* 7.2, p. 1-36. DOI : 10.1145/2738047. HAL : hal-01138090v3 (cf. p. 2, 13).

Vacher, Michel, Benjamin Lecouteux, Pedro Chahuara, François Portet, Brigitte Meillon et Nicolas Bonnefond (2014). « The Sweet-Home speech and multimodal corpus for home automation interaction ». Anglais. In : *The 9th edition of the Language Resources and Evaluation Conference (LREC)*. Reykjavik, Iceland, p. 4499-4506. hal : hal-00953006v1 (cf. p. 38).

Vacher, Michel, François Portet, Anthony Fleury et Norbert Noury (mar. 2011). « Development of Audio Sensing Technology for Ambient Assisted Living : Applications and Challenges ». Anglais. In : *International Journal of E-Health and Medical Communications* 2.1, p. 35-54. doi : 10.4018/jehmc.2011010103. hal : hal-00757407 (cf. p. 13).

Vlachostergiou, Aggeliki, Georgios Stratogiannis, George Caridakis, George Siolas et Phivos Mylonas (2016). « User Adaptive and Context-Aware Smart Home Using Pervasive and Semantic Technologies ». Anglais. In : *Journal of Electrical and Computer Engineering* 2016, p. 20. doi : 10.1155/2016/4789803 (cf. p. 13).

Waldner, Jean-Baptiste (15 fév. 2007). *Nano-Informatique & Intelligence Ambiante - Inventer l'ordinateur du XXIe siècle*, p. 304. isbn : 2746215160. Google Books : wXCiGQAACAAJ (cf. p. 8, 9).

Wang, Haohan et Bhiksha Raj (24 fév. 2017). « On the Origin of Deep Learning ». Anglais. In : *CoRR*. arXiv : 1702.07800v4 [cs.LG] (cf. p. 47, 49).

Watkins, Christopher John Cornish Hellaby (1989). « Learning from Delayed Rewards ». Anglais. Thèse de doct. King's College (cf. p. 32).

Weiser, Mark (sept. 1991). « The Computer for the 21st Century ». Anglais. In : *SIGMOBILE Mob. Comput. Commun. Rev.* 3.3, p. 3-11. issn : 1559-1662. doi : 10.1145/329124.329126 (cf. p. 1, 8, 9).

Werbos, Paul J. (1982). « Applications of advances in nonlinear sensitivity analysis ». Anglais. In : *System Modeling and Optimization : Proceedings of the 10th IFIP Conference New York City, USA, August 31 – September 4, 1981*. Sous la dir. de R. F. Drenick et F. Kozin. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 762-770. isbn : 978-3-540-39459-4. doi : 10.1007/BFb0006203 (cf. p. 48).

Wiesel, T. N. et D. H. Hubel (oct. 1959). « Receptive fields of single neurones in the cattextquotesingles striate cortex ». Anglais. In : *The Journal of Physiology* 148.3, p. 574-591. doi : 10.1113/jphysiol.1959.sp006308 (cf. p. 48).

Wolf, Peter, Andreas Schmidt et Michael Klein (2008). « SOPRANO – An extensible, open AAL platform for elderly people based on semantical contracts ». Anglais. In : *3rd Workshop on Artificial Intelligence Techniques for Ambient Intelligence (AITAmI'08), 18th European Conference on Artificial Intelligence (ECAI 08)* (cf. p. 11, 15).

Yao, Andrew Chi-Chih (oct. 1985). « Separating the polynomial-time hierarchy by oracles ». Anglais. In : *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*. Institute of Electrical et Electronics Engineers (IEEE), p. 1-10. doi : 10.1109/sfcs.1985.49 (cf. p. 47).

Yau, Stephen S. et Junwei Liu (2006). « Hierarchical Situation Modeling and Reasoning for Pervasive Computing ». Anglais. In : *Proceedings of the The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA06)*. SEUS-WCCIA 06. Washington, DC, USA : IEEE Computer Society, p. 5-10. isbn : 0-7695-2560-1. doi : 10.1109/SEUS-WCCIA.2006.25 (cf. p. 19).

Yilmaz, Ozgun et Riza Cenk Erdur (fév. 2012). « iConAwa – An intelligent context-aware system ». Anglais. In : *Expert Systems with Applications* 39.3, p. 2907-2918. issn : 0957-4174. doi : 10.1016/j.eswa.2011.08.152. url : http ://www.sciencedirect.com/science/article/pii/S0957417411012838 (visité le 03/10/2012) (cf. p. 16).

Zaidenberg, Sofia et Patrick Reignier (2011). « Reinforcement Learning of User Preferences for a Ubiquitous Personal Assistant ». Anglais. In : *Advances in Reinforcement Learning*. Sous la dir. d'Abdelhamid Mellouk. Intech, p. 59-80 (cf. p. 25, 26, 28).

Zaidenberg, Sofia, Patrick Reignier et Nadine Mandran (2010). « Learning User Preferences in Ubiquitous Systems : A User Study and a Reinforcement Learning Approach ». Anglais. In : *Artificial Intelligence Applications and Innovations*. Sous la dir. d'Harris Papadopoulos, AndreasS. Andreou et Max Bramer. T. 339. IFIP Advances in Information and Communication Technology. Springer Berlin Heidel-

berg, p. 336-343. ISBN : 978-3-642-16238-1. DOI : 10.1007/978-3-642-16239-8_44 (cf. p. 56).

ZEILER, Matthew D. et Rob FERGUS (2014). « Visualizing and Understanding Convolutional Networks ». Anglais. In : *Computer Vision – ECCV 2014*. Springer Nature, p. 818-833. DOI : 10.1007/978-3-319-10590-1_53 (cf. p. 50).

ZHANG, Chiyuan, Samy BENGIO, Moritz HARDT, Benjamin RECHT et Oriol VINYALS (10 nov. 2016). « Understanding deep learning requires rethinking generalization ». Anglais. In : *Computer Research Repository (CoRR)*. arXiv : 1611.03530 [cs.LG] (cf. p. 93).

ZHANG, Chiyuan, Qianli LIAO, Alexander RAKHLIN, Brando MIRANDA, Noah GOLOWICH et Tomaso POGGIO (19 juil. 2017). *Theory of Deep Learning III : Generalization Properties of SGD*. Rapp. tech. Center for Brains, Minds, et Machines, McGovern Institute for Brain Research, Massachusetts Institute of Technology (cf. p. 93).

ZIMMERMANN, Andreas, Andreas LORENZ et Reinhard OPPERMANN (20-24 août 2007). « An Operational Definition of Context ». Anglais. In : *Modeling and Using Context : 6th International and Interdisciplinary Conference (CONTEXT 2007)*. (Roskilde, Denmark). Sous la dir. de Boicho KOKINOV, Daniel C. RICHARDSON, Thomas R. ROTH-BERGHOFER et Laure VIEU. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 558-571. ISBN : 978-3-540-74255-5. DOI : 10.1007/978-3-540-74255-5_42 (cf. p. 14).

ZUCKERBERG, Mark (déc. 2016). *Building Jarvis*. Anglais. URL : https://www.facebook.com/notes/mark-zuckerberg/building-jarvis/10154361492931634 (visité le 24/01/2017) (cf. p. 12).

Bibliographie personnelle

Alexis Brenon, François Portet et Michel Vacher (nov. 2015). « Using Statistico-Relational Model for Activity Recognition in Smart Home ». Anglais. In : *European Conference on Ambient Intelligence 2015 (AmI-15)*. T. Vol-1528. AmI-2015 Adjunct Proceedings. Athens, Greece. HAL : hal-01222792

Abstract : This paper presents the use of a model which mixes logical knowledge and statistical inference to recognize Activities of Daily Living (ADL) from sensors in a smart home. This model called Markov Logic Network (MLN) has different implementations and we propose to compare three of them, from the widely used Alchemy 1 to the new generic framework DeepDive 2. Finally, we discuss the interest these software products can have for real time activity recognition.

Alexis Brenon, François Portet et Michel Vacher (juin 2016a). « Étude préliminaire d'une méthode de prise de décision adaptative pour la commande vocale dans un habitat intelligent ». In : *RFIA-RJCIA*. Clermont Ferrand, France, p. 2-8. HAL : hal-01326986

Abstract : Dans les habitats intelligents, les prédictions et décisions qui sont souvent faites a priori nécessitent de la part de l'utilisateur une configuration qui peut être complexe et fastidieuse. Ces habitats ont pourtant des capacités de perception requises pour analyser le comportement de l'utilisateur et modifier ses décisions automatiquement. Nous présentons une étude préliminaire qui teste un système de décision à partir d'une commande vocale et de son contexte modifié par renforcement. Le système expérimenté sur un corpus réaliste montre le potentiel d'une telle adaptation.

Alexis Brenon, François Portet et Michel Vacher (sept. 2016b). « Preliminary Study of Adaptive Decision-Making System for Vocal Command in Smart Home ». Anglais.

In : *12th International Conference on Intelligent Environments*. London, United Kingdom, p. 218-221. HAL : hal-01345333

Abstract : In smart homes, prediction and decision are often defined a priori and require tuning from the user, which can be tedious, and complex. However, these smart homes have the ability to analyze the user behavior and to modify their decisions automatically. We present a preliminary study that tests a decision system from voice command and the user's context, which is modified by reinforcement learning. The system ran on a realistic corpus, which shows the interest of such an adaptation.

Annexes

Annexe A

Représentation graphique des capteurs

Les figures regroupées ici présentent les différentes icônes utilisées pour la représentation des données sous forme graphique.

La Figure A.1a liste les icônes représentant les capteurs à états. On y voit, de gauche à droite, les capteurs de contacts (portes et fenêtres), les rideaux, les volets, les prises de courant, les détecteurs de mouvement. L'état de la lumière est particulier puisque ce capteur est à la fois à état, allumé ou éteint, mais également continu puisque l'intensité d'éclairage peut varier.

Les icônes de la Figure A.1b permettent de représenter des capteurs avec des valeurs continues. On y retrouve la température de l'air, l'humidité ambiante, la qualité globale de l'air, le niveau de bruit et la luminosité. L'opacité de l'icône (définie sur une échelle de 0 – transparent – à 255 – totalement opaque –) permet de représenter la valeur actuelle sur une échelle définie. Par exemple, l'humidité est définie entre 1 et 100 (puisqu'il s'agit d'une humidité relative exprimée en pourcentage) tandis que la température peut évoluer de 10 à 50 °C, ce qui semble des valeurs limites acceptables pour une utilisation dans un habitat classique dans nos régions.

La Figure A.2 reprend les icônes permettant de représenter le résultat d'une inférence. Les commandes vocales reconnues sont regroupées dans la Figure A.2a. De gauche à droite, les commandes de contrôle de la lumière, des rideaux, des volets et de la radio.

(a) Icônes de capteurs à états

(b) Icônes de capteurs à valeurs continues

Fig. A.1 : Icônes de capteurs

(a) Icônes des commandes vocales

(b) Icônes des activités

FIG. A.2 : Icônes des résultats d'inférence

Il s'agit de commandes complémentaires, avec une commande « allumer, ouvrir » et une commande « éteindre, fermer ». Les icônes suivantes sont associées aux commandes et permettent d'appeler les urgences ou la famille ou de demander au système l'heure ou la température.

Les dernières icônes (Figure A.2b) ne sont utilisées que lors de la génération d'images à partir de données inférées pour lesquelles l'activité de l'utilisateur est connue. Les activités possibles sont :

- « aucune » si l'activité ne peut être classée ;
- « en train de manger » ;
- « en train de cuisiner » ;
- « en train de faire la vaisselle » ;
- « en train de nettoyer » ;
- « en train de ranger » ;
- « en train de se détendre (lire) » ;
- « en train de téléphoner » ;
- « en train de dormir ».

Annexe **B**

Les types de neurones

Un réseau neuronal est composé de différentes couches, reliées les unes aux-autres, chaque couche étant composée d'un ensemble de neurones (ou d'unités). Comme présenté Section IV.1.1, un neurone est une unité de calcul qui à partir d'un certain nombre de valeurs en entrées, réalise une opération et fourni une ou des valeurs en sortie (Figure B.1). Il existe différents types de neurones, en fonction de l'opération qu'ils réalisent. Nous présentons ici les différents types de neurones évoqués dans ce mémoire.

Les unités totalement connectées

Les unités totalement connectées sont l'archétype même du neurone artificiel tel que définit par McCulloch et Pitts (1943). Chaque neurone d'une couche totalement connectée est relié à l'ensemble des neurones de la couche qui le précède, comme le montre la Figure B.2. Chaque neurone possède alors un vecteur de poids $\mathbf{w}$, contenant autant de valeur que le nombre de valeurs d'entrée, ainsi qu'une valeur de biais b. Ces valeurs peuvent ensuite être modifiées lors d'une phase d'apprentissage. Initialement, les neurones possédaient également une fonction d'activation f. Le neurone calculait alors l'image par f du produit vectoriel $\mathbf{w} \cdot \mathbf{x} + b$ où $\mathbf{x}$ est le vecteur des valeurs d'entrée : $o = f(\sum w_n x_n + b)$.

Historiquement, la fonction f était une fonction de seuillage discontinue, retournant 0 si le produit vectoriel était inférieur au seuil, 1 sinon. Dans les implémentations actuelles, la

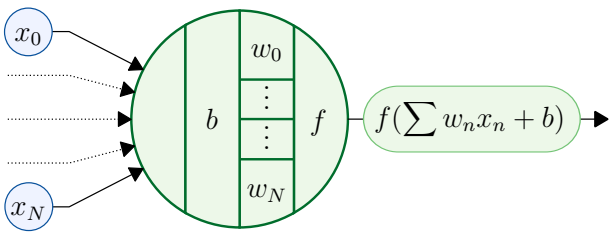


Fig. B.1 : Fonctionnement d'un neurone dans les systèmes neuronaux artificiels

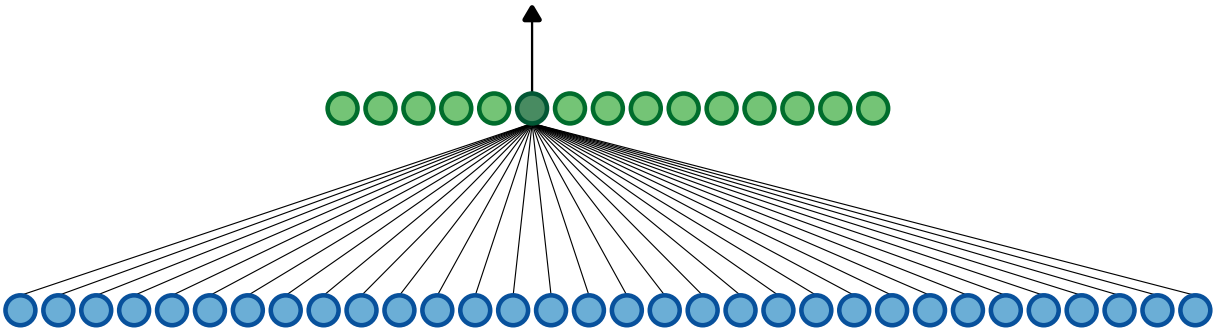


FIG. B.2 : Liens entre une unité totalement connectée et les unités de la couche précédente.

fonction d'activation est généralement extraite dans une unité à part, afin de tirer profit des capacités des cartes graphiques d'ordinateurs concernant les calculs matriciels. Une couche totalement connectée est alors uniquement définie par le nombre de neurones n qui la compose, on la note donc FC_n.

Les unités d'activation

Les unités d'activations suivent généralement chacune des couches d'un réseau de neurones. Elles permettent d'appliquer un seuil sur les valeurs retournées par la couche précédente afin de différencier les neurones actifs (qui réagisse à l'entrée fournie) des neurones inactifs. La fonction de seuil n'est cependant pas dérivable et les méthodes d'apprentissage basées sur le gradient (comme SGD) sont alors inefficaces. Pour répondre à ce problème, la fonction sigmoïde a été utilisée comme alternative.

Parmi les nombreuses fonctions utilisées comme fonction d'activation, on retrouve de nos jours : la fonction *tanh*, ReLU (*Rectified Linear Unit*), Maxout ou SoftMax (KRIZHEVSKY, SUTSKEVER et G. E. HINTON 2012). Ces unités ne possèdent pas de vecteurs de poids et leur comportement n'est donc pas modifié par les phases d'apprentissage.

Les unités convolutives

Les unités convolutives ont été conçues en supposant que les données d'entrées sont des images, et donc des matrices 2D et non des vecteurs. Ainsi, chaque neurone d'une couche convolutive possède un ensemble de n filtres (matrices) avec des poids pouvant être modifiés pendant l'apprentissage. Chaque filtre est de taille f minime par rapport à la taille de la matrice d'entrée. Lors du passage d'une entrée $\mathbf{X}$ dans une couche convolutive, chaque filtre est convolué sur l'image, comme le montre la Figure B.3, produisant autant de matrices (cartes d'activation) qu'il y a de filtres.

En plus du nombre de filtres, deux autres hyper-paramètres viennent définir une unité de convolution. Le pas (*stride* – s) permet de définir le déplacement du filtre sur l'image.

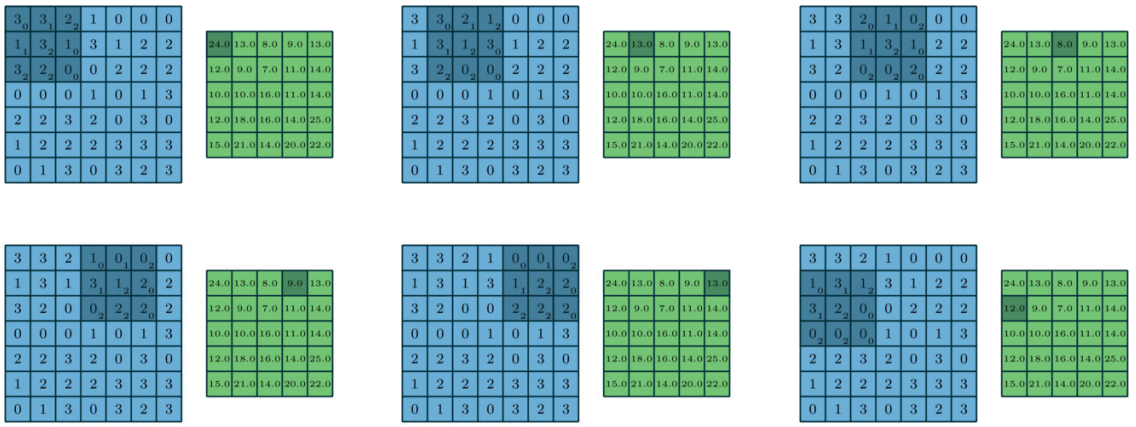


FIG. B.3 : Quelques étapes de la convolution d'un filtre sur une image (DUMOULIN 2016).

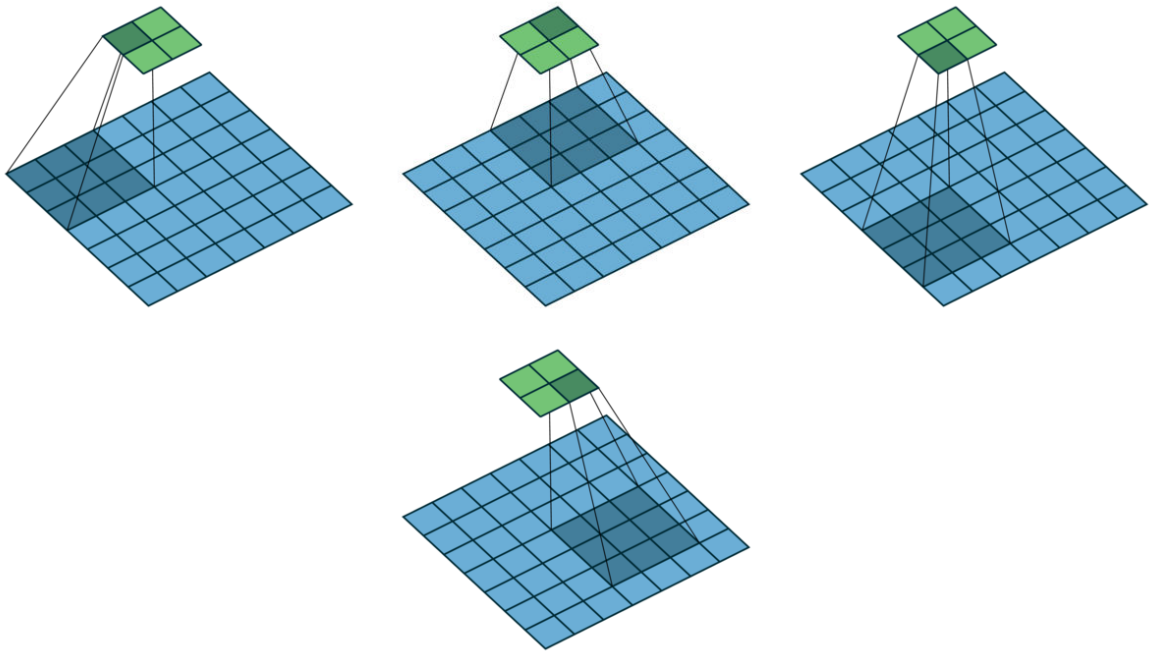


FIG. B.4 : Impact du pas de convolution sur la matrice de sortie (DUMOULIN 2016).

Dans la figure précédente, un pas de 1 est utilisé. L'utilisation d'un pas plus grand permet de réduire la taille de la sortie. Ainsi, à partir d'une entrée de taille 7 avec un filtre de taille 3 et un pas de 1, la sortie est de taille 5 (Figure B.3). En utilisant un pas de 3, la sortie est désormais de taille 2, comme le montre la Figure B.4.

L'augmentation du pas peut cependant poser problème, comme ici puisque le filtre ne peut alors pas convoluer de bout en bout de l'image. Pour parer ce défaut, l'utilisation d'un bourrage (*padding* – p) permet de compléter l'entrée avec une bordure n'influençant pas le résultat de la convolution. La Figure B.5 montre l'intérêt de l'utilisation d'un bourrage de taille 1 pour permettre au filtre de convoluer sur la totalité de l'image. Ces hyper-

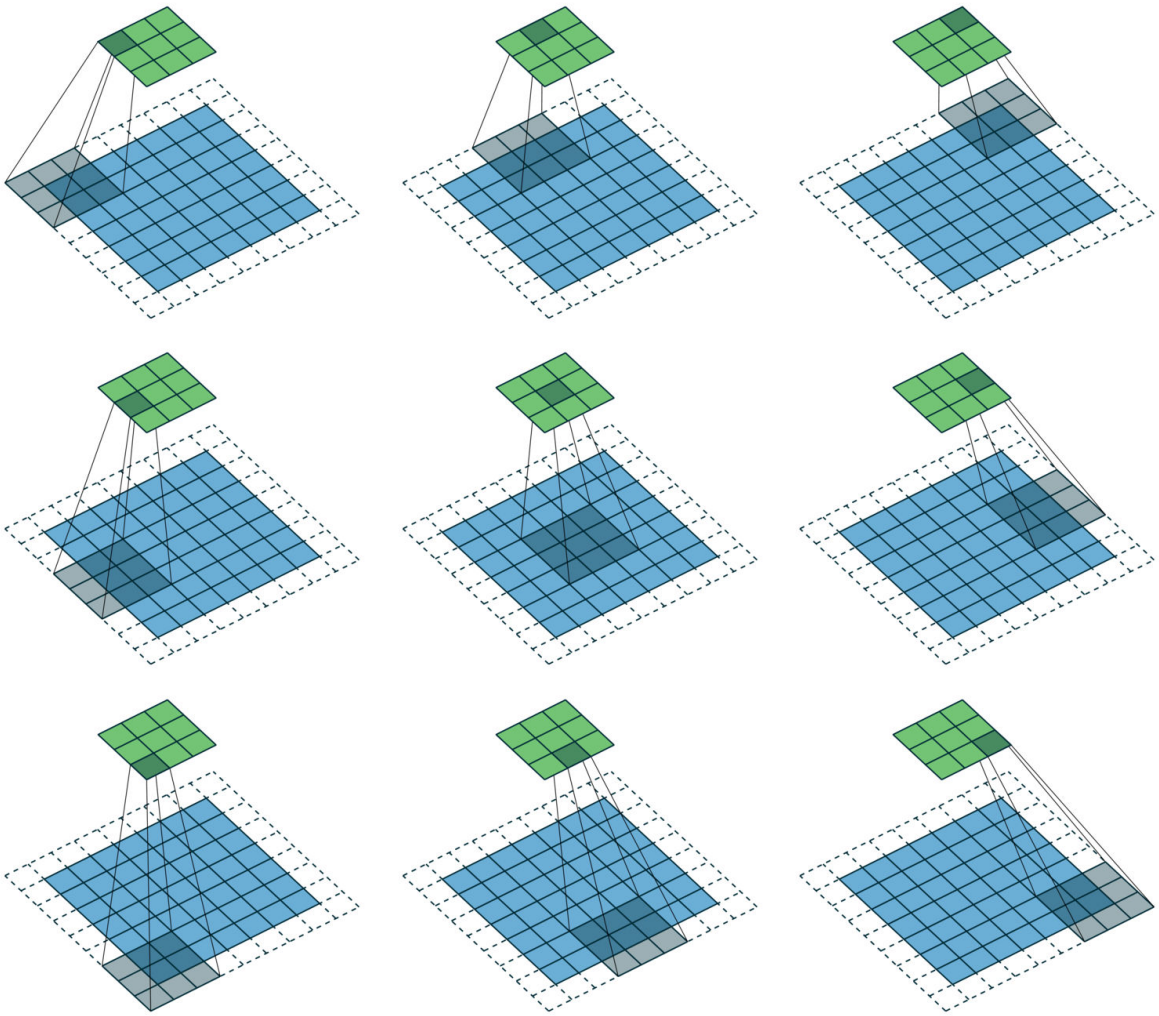


FIG. B.5 : Impact du bourrage lors de la convolution (DUMOULIN 2016).

paramètres permettent donc de définir une unité de convolution, que l'on note $C_n[f+p/s]$.

Il existe de nombreux autre type de couches ou d'unités pouvant être utilisées dans un réseau de neurones : sélection (*pooling*, SCHERER, MÜLLER et BEHNKE 2010), décrochage (*dropout*, G. E. HINTON, SRIVASTAVA et coll. 2012), normalisation par lot (*batch normalization*, IOFFE et SZEGEDY 2015), etc. Leur description n'est cependant pas pertinente dans ce mémoire et le lecteur pourra se référer aux ouvrages cités précédemment.

Index

P

Q

R

S

T

U